



Práctica P04- Filtrado de paquetes

Gabriel Lazovsky Igual

Alejandro Rodríguez Ferrer

INDICE

CORTAFUEGOS PERSONAL	5
1- Establecer distintas reglas de filtrado sin estado sobre las cadenas INPUT y OUTPUT. Filtraremos ambos sentidos de la conexión, con reglas complementarias. Fijar en primer lugar la política DROP para el tráfico entrante y saliente del equipo, y habilitar a continuación el intercambio ICMP con el resto de los equipos de nuestra subred. Permitir el tráfico saliente hacia los puertos 80/tcp, 443/tcp y 53/udp (nuestro equipo actúa como cliente). Habilitar el tráfico entrante al puerto 22/tcp y 80/tcp (nuestro equipo actúa como servidor)	5
2- Comprobar el efecto que tiene el orden de las reglas de filtrado. Partiendo de una configuración sin filtros, bloquear todo el tráfico TCP saliente salvo el dirigido a una dirección IP concreta de nuestra subred, viendo el efecto del orden en la entrada de las dos reglas necesarias.	8
3- Definir una regla que redirija todo el tráfico SSH saliente que va hacia una dirección IP concreta (por ejemplo, 1.2.3.4) para que se redirija a la IP de un servidor real que tenga el servicio SSH activo. Para probar su funcionamiento, abrir una sesión SSH contra 1.2.3.4.	9
4- Desde una máquina remota, utilizar nmap para identificar el sistema operativo de nuestra máquina virtual Linux. A continuación, aplicar una regla con estado para impedir conexiones TCP inválidas (las que no se inician con el segmento SYN). Volver a utilizar la herramienta nmap para identificar el SO y comentar los nuevos resultados obtenidos. Nota1. Para detectar el SO de un equipo es necesario que tenga abierto algún puerto TCP.	10
5- Partir nuevamente de una configuración sin filtros y bloquear todo el tráfico entrante y saliente. A continuación, definir reglas con estado que permitan iniciar conexiones TCP o UDP hacia el exterior, así como el tráfico ICMP siempre que sea iniciado por nosotros o relacionado con nuestras conexiones. Comprobarlo iniciando conexiones exteriores, intentando conectar desde el exterior (por ejemplo, al servidor SSH) y lanzando o recibiendo mensajes de alcanzabilidad (ping).	12
6- Considere la siguiente situación: Se están produciendo accesos continuos a nuestro servidor OpenSSH para intentar descubrir mediante fuerza bruta usuarios y contraseñas válidas del sistema. Nos gustaría limitar el número de accesos desde una única dirección IP origen (por ejemplo, un máximo de 3 conexiones cada 120 segundos desde cada IP origen), para minimizar el riesgo este tipo de intentos sin afectar a los usuarios legítimos. Busque una solución para este problema mediante iptables, e indique la(s) regla(s) que habría que aplicar para implantar esta política y la función que cumplen. Para acotar más aún estos intentos de acceso es posible limitar también el número máximo de reintentos de autenticación en cada sesión (por ejemplo, a dos). ¿Cómo podríamos configurar esta nueva restricción?	13
CORTAFUEGOS PERSONAL CON UFW	15

7- UFW (Uncomplicated Firewall) es una aplicación desarrollada por Canonical (Ubuntu) para activar reglas iptables y mantener un cortafuegos personal de manera sencilla. Para realizar este apartado se recomienda usar una máquina Ubuntu, que ya tienen instalado UFW. Llevar a cabo las siguientes tareas:	15
a. Verificar el estado del cortafuegos personal y si ya existen reglas iptables	15
b. Arrancar algún servicio en el equipo para realizar las pruebas posteriores (Apache2, OpenSSH Server, ...)	15
c. Arrancar UWF y configurar su arranque automático. Iniciar el cortafuegos (ufw enable)	15
d. Comprobar el estado de UFW (ufw status). También es posible ver información extendida (ufw status verbose) que incluye la política definida, y las reglas iptables generadas (iptables -L -n more). Con este último comando podemos ver las nuevas reglas y cadenas generadas por UFW.	16
e. Usar el comando ufw default allow deny incoming outgoing para definir la configuración por defecto del cortafuegos personal. Comprobar el efecto de estos comandos intentando conexiones hacia o desde el equipo.	16
f. Establecer una política por defecto que permita conexiones salientes e impida conexiones entrantes.	16
g. Sobre la configuración anterior permitir conexiones entrantes a los servicios arrancados en el apartado b) (ufw allow ...)	17
h. También es posible limitar las conexiones realizadas desde una IP origen (ufw limit ...), para mitigar los ataques de diccionario. Activar el acceso limitado al servidor SSH de nuestro equipo. ¿Qué límite se ha establecido para las conexiones SSH entrantes?	17
ROUTER CON FUNCIÓN DE CORTAFUEGOS Y NAT	18
8- Vamos a preparar una maqueta como la reflejada en el gráfico adjunto. Usaremos tres máquinas virtuales (FW, interna y externa) y asociaremos un segundo interface a la máquina que actúa como cortafuegos. Se recomienda usar el SO Ubuntu Server al menos en FW, ya que cuenta con los paquetes shorewall en el repositorio oficial. La red exterior tendrá un direccionamiento público (20.20.20.0/24) y la privada tendrá direccionamiento privado (10.10.10.0/24) (puede usar las redes predefinidas del entorno de virtualización para ambas redes). Las máquinas interna y externa tendrán como router por defecto al equipo que actúa como firewall (FW).	18
9- Instalar shorewall en el equipo FW. También se recomienda instalar la herramienta tcpdump y algún servicio (por ejemplo, Apache) en las tres máquinas.	22
10- Configurar los interfaces de red de los tres equipos y default GW en los hosts interno y externo. Reiniciar los servicios de red y comprobar que somos capaces de alcanzar las máquinas interna y externa desde el cortafuegos (FW). Habilitar el encaminamiento en el cortafuegos (net.ipv4.ip_forward=1 en el archivo /etc/sysctl.conf) y comprobar que es posible	

comunicar las máquinas interna y externa sin restricciones. Levantar algún servicio en los equipos y comprobar que es posible acceder remotamente.	24
11- Describir unas políticas básicas en el cortafuegos Shorewall: se acepta el tráfico desde la red interna al exterior (saliente), se impide tráfico entrante desde el exterior, se acepta tráfico saliente desde el cortafuegos hacia cualquier sitio. Se acepta tráfico entrante al cortafuegos desde la red interna. Activar las reglas (systemctl start shorewall) y verificar su correcto funcionamiento.	27
12- Habilitar el enmascaramiento de direcciones desde la red interior hacia la red exterior. Verificar que todo el tráfico saliente se enmascara (puede usar el sniffer tcpdump). Se recomienda desactivar el GW en la máquina Externa y comprobar que sigue siendo posible alcanzar sus servicios desde el interior.	30
13- Implantar una política restrictiva para el tráfico del FW: todo el tráfico entrante, saliente y que atraviesa el FW será rechazado. A continuación, se habilitará la salida a servicios concretos (http, https, SMTP, ...). Configurar algún servicio de port forwarding (como Apache) que será visible desde el exterior y ofrecido por la máquina de la red interna. Configurar un acceso al cortafuegos por SSH desde una dirección externa concreta. Establecer el redireccionamiento de todo el tráfico DNS saliente para que se reenvíe a un servidor de DNS externo concreto. Para realizar este apartado se recomienda estudiar los ejemplos de reglas shorewall presentes en el manual de esta aplicación (man shorewall-rules).	33
PARTES OPCIONALES	35
14- Vamos a trabajar con la sintaxis nativa del módulo Nftables (nft). Para ello realizaremos las siguientes tareas:	35
15- Instalar una máquina virtual con la distribución Linux IPFire_ (https://www.ipfire.org/). Estudiar sus funcionalidades y realizar algunas tareas de configuración a través de su panel de control (WUI). Indicar también algunos add-ons que pueden instalarse para incrementar sus funcionalidades.	40

Cortafuegos personal

1- Establecer distintas reglas de filtrado sin estado sobre las cadenas INPUT y OUTPUT. Filtraremos ambos sentidos de la conexión, con reglas complementarias. Fijar en primer lugar la política DROP para el tráfico entrante y saliente del equipo, y habilitar a continuación el intercambio ICMP con el resto de los equipos de nuestra subred. Permitir el tráfico saliente hacia los puertos 80/tcp, 443/tcp y 53/udp (nuestro equipo actúa como cliente). Habilitar el tráfico entrante al puerto 22/tcp y 80/tcp (nuestro equipo actúa como servidor)

Primero se verifica la IP de nuestra máquina que será útil para próximos apartados con el comando:

```
ifconfig
```

```
[root@server ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 172.22.0.138 netmask 255.255.255.0 broadcast 172.22.0.255
      inet6 fe80::20c:29ff:fea0:a014 prefixlen 64 scopeid 0x20<link>
        ether 00:0c:29:a0:a0:14 txqueuelen 1000 (Ethernet)
          RX packets 41 bytes 4842 (4.7 KiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 52 bytes 4816 (4.7 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
          RX packets 0 bytes 0 (0.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 0 bytes 0 (0.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

En el caso de Rocky instalamos y reiniciamos los siguientes servicios:

```
sudo dnf install -y iptables iptables-services
```

En Ubuntu:

```
sudo apt install iptables iptables-persistent -y
```

Para habilitar e iniciar iptables:

```
sudo systemctl enable iptables  
sudo systemctl start iptables
```

En Ubuntu:

```
sudo systemctl enable nftables.service  
sudo systemctl start nftables.service
```

*Este comando lo usaremos para guardar de forma persistente las reglas en /etc/sysconfig/iptables :

```
sudo service iptables save  
sudo netfilter-persistent save
```

Para cargar las reglas guardadas en Ubuntu, iptables-services lo hace automático:

```
sudo netfilter-persistent reload // Se guardan en /etc/iptables/rules.v4 y  
/etc/iptables/rules.v6
```

Rocky usa firewalld, por lo que debemos desabilitarlo:

```
sudo systemctl disable firewalld –now
```

Mientras que en el caso de Ubuntu instalamos y habilitamos estos servicios principalmente para la persistencia:

```
apt install iptables-persistent netfilter-persistent  
systemctl enable netfilter-persistent
```

Primero listamos las reglas con el siguiente comando:

```
iptables -L -n -v
```

```
root@p04:~# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Podemos observar que de primeras las políticas están por defecto todas a ACCEPT.

Ahora con los siguientes comandos vamos a cambiar la política de todo a default DENY:

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
```

Ahora con las siguientes reglas habilitamos el ICMP en la maquina:

```
# Permitir que entren pings desde la subred
iptables -A INPUT -p icmp -s 172.22.0.0/24 -j ACCEPT

# Permitir que salgan nuestras respuestas (o nuestros propios pings) hacia la
subred
iptables -A OUTPUT -p icmp -d 172.22.0.0/24 -j ACCEPT
```

Después permitimos tráfico saliente (Cliente Web y DNS)

```
# --- REGLAS DE SALIDA (Tu petición) ---
# Web (HTTP y HTTPS)
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 443 -j ACCEPT
# DNS (Resolución de nombres)
iptables -A OUTPUT -p udp --dport 53 -j ACCEPT

# --- REGLAS DE ENTRADA (La respuesta del servidor) ---
# En un filtro sin estado, debemos aceptar paquetes que vienen DE esos
puertos
iptables -A INPUT -p tcp --sport 80 -j ACCEPT
iptables -A INPUT -p tcp --sport 443 -j ACCEPT
iptables -A INPUT -p udp --sport 53 -j ACCEPT
```

Para el rol como servidor, habilitamos el puerto 22 y 80:

iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

Explicación, lógica **Sin Estado**:

Cliente:

OUTPUT --> --dport (al puerto del servicio)

INPUT --> --sport (del puerto del servicio remoto)

Servidor:

INPUT --> --dport (a tu puerto de servicio)

OUTPUT --> --sport (a tu puerto de servicio)

2- Comprobar el efecto que tiene el orden de las reglas de filtrado. Partiendo de una configuración sin filtros, bloquear todo el tráfico TCP saliente salvo el dirigido a una dirección IP concreta de nuestra subred, viendo el efecto del orden en la entrada de las dos reglas necesarias.

Esta vez comenzaremos con reglas ACCEPT para poder ver la diferencia:

sudo iptables -P INPUT ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -F

Regla 1, TCP saliente bloqueado:

sudo iptables -A OUTPUT -p tcp -j DROP
--

Regla 2, permitir IP 192.168.1.50:

sudo iptables -A OUTPUT -p tcp -d 192.168.1.50 -j ACCEPT
--

```
[root@server ~]# sudo iptables -L OUTPUT -v -n --line-numbers
Chain OUTPUT (policy ACCEPT 79 packets, 6348 bytes)
num  pkts bytes target     prot opt in     out     source               destination
1      0     0  DROP       tcp   --  *      *      0.0.0.0/0            0.0.0.0
/0
2      0     0           tcp   --  *      *      0.0.0.0/0            192.168
.2.60
```

De esta forma, a pesar de que la regla 2 permite explícitamente la ip 192.168.1.50, la regla 1 la rechaza antes. Para corregirlo hay que ejecutar estos comandos:

sudo iptables -F
sudo iptables -A OUTPUT -p tcp -d 192.168.1.50 -j ACCEPT
sudo iptables -A OUTPUT -p tcp -j DROP

```
[root@server ~]# sudo iptables -A OUTPUT -p tcp -d 192.168.2.60 -j ACCEPT
[root@server ~]# sudo iptables -A OUTPUT -p tcp -j DROP
[root@server ~]# sudo iptables -L OUTPUT -v -n --line-numbers
Chain OUTPUT (policy ACCEPT 82 packets, 6576 bytes)
num  pkts bytes target     prot opt in     out     source               destination
1      0     0  ACCEPT    tcp   --  *      *      0.0.0.0/0            192.168
.2.60
2      0     0  DROP      tcp   --  *      *      0.0.0.0/0            0.0.0.0
/0
```

- 3- Definir una regla que redirija todo el tráfico SSH saliente que va hacia una dirección IP concreta (por ejemplo, 1.2.3.4) para que se redirija a la IP de un servidor real que tenga el servicio SSH activo. Para probar su funcionamiento, abrir una sesión SSH contra 1.2.3.4.

Para este apartado necesitaremos una máquina auxiliar (IP 172.22.0.139), donde levantaremos el servidor SSH. En la máquina principal ejecutaremos lo siguiente:
PD: DNAT en OUTPUT solo afecta al tráfico de origen local, no al que entra por las interfaces.

iptables -t nat -A OUTPUT -p tcp -d 1.2.3.4 --dport 22 -j DNAT --to-destination 172.22.0.139:22

Para comprobarlo, ejecutamos en la máquina principal:

ssh root@1.2.3.4

Y deberíamos de conectarnos a la máquina auxiliar

```
[root@server ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 172.22.0.139 netmask 255.255.255.0 broadcast 172.22.0.255
        inet6 fe80::20c:29ff:feb5:5c64 prefixlen 64 scopeid 0x20<link>
          ether 00:0c:29:b5:5c:64 txqueuelen 1000 (Ethernet)
            RX packets 90 bytes 11131 (10.8 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 81 bytes 10069 (9.8 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
          loop txqueuelen 1000 (Local Loopback)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 0 bytes 0 (0.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@server ~]# ^C
[root@server ~]# exit
cerrar sesión
Connection to 1.2.3.4 closed.
[root@server ~]# _
```

4- Desde una máquina remota, utilizar **nmap** para identificar el sistema operativo de nuestra máquina virtual Linux. A continuación, aplicar una regla con estado para impedir conexiones TCP inválidas (las que no se inician con el segmento SYN). Volver a utilizar la herramienta nmap para identificar el SO y comentar los nuevos resultados obtenidos. Nota1. Para detectar el SO de un equipo es necesario que tenga abierto algún puerto TCP.

Desde la máquina auxiliar escaneamos el sistema operativo de la siguiente manera:

```
nmap -O 172.22.0.138
```

```
[root@server ~]# nmap -O 172.22.0.138
Starting Nmap 7.92 ( https://nmap.org ) at 2025-12-21 21:08 CET
Nmap scan report for 172.22.0.138
Host is up (0.0011s latency).
Not shown: 989 filtered tcp ports (no-response), 10 filtered tcp ports (host-prohibited)
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 00:0C:29:A0:A0:14 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 4.X15.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6, Linux 5.0 - 5.4
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/
submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.73 seconds
```

La regla iptables que ejecutaremos para evitarlo, en la máquina anfitriona:

```
iptables -A INPUT -m state --state INVALID -j DROP
```

Tras ejecutar la nueva regla, se van a dropear las respuestas que espera nmap, limitando la información que obtiene.

```
[root@server ~]# nmap -O 172.22.0.138
Starting Nmap 7.92 ( https://nmap.org ) at 2025-12-22 18:24 CET
Nmap scan report for 172.22.0.138
Host is up (0.00079s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    closed http
MAC Address: 00:0C:29:A0:A0:14 (VMware)
Device type: general purpose
Running: Linux 4.X15.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/
submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.27 seconds
```

Fijarse en el número de paquetes filtrados, ya no salen 10 como host-prohibited, desaparece el warning.

Esto hace que nuestra máquina, envie menos respuestas, sea más difícil de identificar y de fingerprinting, y sea más sigiloso frente a scans de nmap.

5- Partir nuevamente de una configuración sin filtros y bloquear todo el tráfico entrante y saliente. A continuación, definir reglas con estado que permitan iniciar conexiones TCP o UDP hacia el exterior, así como el tráfico ICMP siempre que sea iniciado por nosotros o relacionado con nuestras conexiones. Comprobarlo iniciando conexiones exteriores, intentando conectar desde el exterior (por ejemplo, al servidor SSH) y lanzando o recibiendo mensajes de alcanzabilidad (ping).

Comenzamos estableciendo una política estricta:

iptables -F
iptables -P INPUT DROP
iptables -P OUTPUT DROP

Creamos la regla que permite tráfico si ya está establecido/iniciado, para la entrada y salida:

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

Reglas para permitir iniciar conexiones hacia el exterior. Permitiendo el estado “NEW” hacia fuera:

Para TCP/UDP:

iptables -A OUTPUT -p tcp -m state --state NEW -j ACCEPT
iptables -A OUTPUT -p udp -m state --state NEW -j ACCEPT

Para ICMP:

iptables -A OUTPUT -p icmp -m state --state NEW -j ACCEPT

Probamos a hacer ping desde nuestra máquina:

ping 172.22.0.139

```
[root@server ~]# ping 172.22.0.139
PING 172.22.0.139 (172.22.0.139) 56(84) bytes of data.
64 bytes from 172.22.0.139: icmp_seq=1 ttl=64 time=0.462 ms
64 bytes from 172.22.0.139: icmp_seq=2 ttl=64 time=1.12 ms
64 bytes from 172.22.0.139: icmp_seq=3 ttl=64 time=1.09 ms
64 bytes from 172.22.0.139: icmp_seq=4 ttl=64 time=1.38 ms
```

Desde la auxiliar:

```
ping 172.22.0.138
[root@server ~]# ping 172.22.0.138
PING 172.22.0.138 (172.22.0.138) 56(84) bytes of data.
^C
--- 172.22.0.138 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6176ms
```

6- Considere la siguiente situación: Se están produciendo accesos continuos a nuestro servidor OpenSSH para intentar descubrir mediante fuerza bruta usuarios y contraseñas válidas del sistema. Nos gustaría limitar el número de accesos desde una única dirección IP origen (por ejemplo, un máximo de 3 conexiones cada 120 segundos desde cada IP origen), para minimizar el riesgo este tipo de intentos sin afectar a los usuarios legítimos. Busque una solución para este problema mediante iptables, e indique la(s) regla(s) que habría que aplicar para implantar esta política y la función que cumplen. Para acotar más aún estos intentos de acceso es posible limitar también el número máximo de reintentos de autenticación en cada sesión (por ejemplo, a dos). ¿Cómo podríamos configurar esta nueva restricción?

Para solucionarlo, además de las reglas iptables, usaremos el módulo “recent”, que crea una lista dinámica de IPs atacantes. De forma que, si la IP ya está en la lista y ha intentado 3 veces acceder en los últimos 120 segundos, DROP. Si no está en la lista, se registra la IP.

Verificamos si la IP está en la lista y excede el límite:

```
iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --name SSH_BRUTE --update --seconds 120 --hitcount 3 --name SSH_BRUTE -j DROP
```

Si pasa el filtro anterior, añadir la IP a la lista (o actualizar su timestamp) y aceptar

```
iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --set --name SSH_BRUTE -j ACCEPT
```

```
[root@server ~]# ssh root@172.22.0.138
The authenticity of host '172.22.0.138 (172.22.0.138)' can't be established.
ED25519 key fingerprint is SHA256:1XqGcOzTF8uPYnSAZDnIuBXG5fMNFyhmfGadg5z3+7E.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.22.0.138' (ED25519) to the list of known hosts.
root@172.22.0.138's password:
Permission denied, please try again.
root@172.22.0.138's password:
Permission denied, please try again.
root@172.22.0.138's password:
root@172.22.0.138: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
[root@server ~]#
```

Respondiendo a la pregunta, desde iptables no podemos limitar el número de intentos cuando se está intentando logear, eso lo debemos hacer modificando /etc/ssh/sshd_config de la siguiente forma:

```
#LoginGraceTime 2m
#PermitRootLogin prohibit-password
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

```
#LoginGraceTime 2m
#PermitRootLogin prohibit-password
PermitRootLogin yes
#StrictModes yes
MaxAuthTries 2
#MaxSessions 10
```

Cortafuegos personal con UFW

7- UFW (Uncomplicated Firewall) es una aplicación desarrollada por Canonical (Ubuntu) para activar reglas iptables y mantener un cortafuegos personal de manera sencilla. Para realizar este apartado se recomienda usar una máquina Ubuntu, que ya tienen instalado UFW. Llevar a cabo las siguientes tareas:

- a. Verificar el estado del cortafuegos personal y si ya existen reglas iptables

```
sudo ufw status
root@server:~# sudo ufw status
Status: inactive
root@server:~# _
```

- b. Arrancar algún servicio en el equipo para realizar las pruebas posteriores (Apache2, OpenSSH Server, ...)

```
sudo systemctl start ssh
sudo systemctl enable ssh
```

- c. Arrancar UWF y configurar su arranque automático. Iniciar el cortafuegos (ufw enable)

```
sudo ufw enable
sudo systemctl enable ufw
root@server:~# sudo ufw enable
Firewall is active and enabled on system startup
root@server:~# sudo systemctl enable ufw
Synchronizing state of ufw.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable ufw
Created symlink /etc/systemd/system/multi-user.target.wants/ufw.service → /usr/lib/systemd/system/ufw.service.
```

d. Comprobar el estado de UFW (`ufw status`). También es posible ver información extendida (`ufw status verbose`) que incluye la política definida, y las reglas iptables generadas (`iptables -L -n | more`). Con este último comando podemos ver las nuevas reglas y cadenas generadas por UFW.

Para ver las políticas por defecto:

<code>sudo ufw status</code>
<code>sudo ufw status verbose</code>

```
root@server:~# sudo ufw status
Status: active
root@server:~# sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip
...
```

Para ver las reglas reales que UFW ha creado:

<code>sudo iptables -L -n more</code>

e. Usar el comando `ufw default allow|deny incoming|outgoing` para definir la configuración por defecto del cortafuegos personal. Comprobar el efecto de estos comandos intentando conexiones hacia o desde el equipo.

<code>sudo ufw enable</code>
<code>sudo ufw status</code>

<code>sudo ufw default deny incoming</code>
<code>sudo ufw default allow outgoing</code>

f. Establecer una política por defecto que permita conexiones salientes e impida conexiones entrantes.

Definir explícitamente que se cierra todo lo que entra, pero permitiendo todo lo que sale:

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
root@server:~# sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
root@server:~# sudo ufw default allow outgoing
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
```

g. Sobre la configuración anterior permitir conexiones entrantes a los servicios arrancados en el apartado b) (ufw allow ...)

```
sudo ufw allow ssh
root@server:~# sudo ufw allow ssh
Rule added
Rule added (v6)
```

h. También es posible limitar las conexiones realizadas desde una IP origen (ufw limit ...), para mitigar los ataques de diccionario. Activar el acceso limitado al servidor SSH de nuestro equipo. ¿Qué límite se ha establecido para las conexiones SSH entrantes?

Borrar la regla anterior:

```
sudo ufw delete allow ssh
root@server:~# sudo ufw delete allow ssh
Rule deleted
Rule deleted (v6)
```

Aplicar la regla con límite:

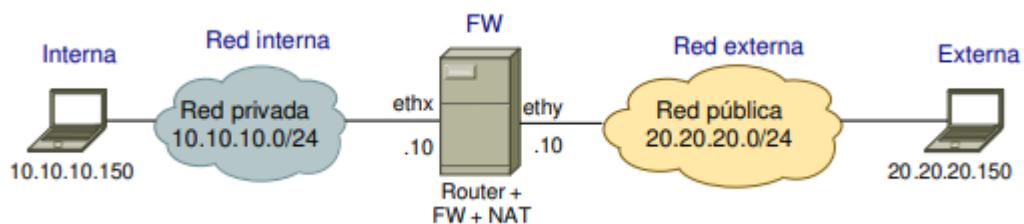
```
sudo ufw limit ssh
root@server:~# sudo ufw limit ssh
Rule added
Rule added (v6)
```

Esta regla nos sirve para limitar el número de intentos máximos (6 por default) en un intervalo de tiempo 30 segundos, desde una misma IP.

Router con función de cortafuegos y NAT

8- Vamos a preparar una maqueta como la reflejada en el gráfico adjunto. Usaremos tres máquinas virtuales (FW, interna y externa) y asociaremos un segundo interface a la máquina que actúa como cortafuegos. Se recomienda usar el SO Ubuntu Server al menos en FW, ya que cuenta con los paquetes shorewall en el repositorio oficial. La red exterior tendrá un direccionamiento público (20.20.20.0/24) y la privada tendrá direccionamiento privado (10.10.10.0/24) (puede usar las redes predefinidas del entorno de virtualización para ambas redes). Las máquinas interna y externa tendrán como router por defecto al equipo que actúa como firewall (FW).

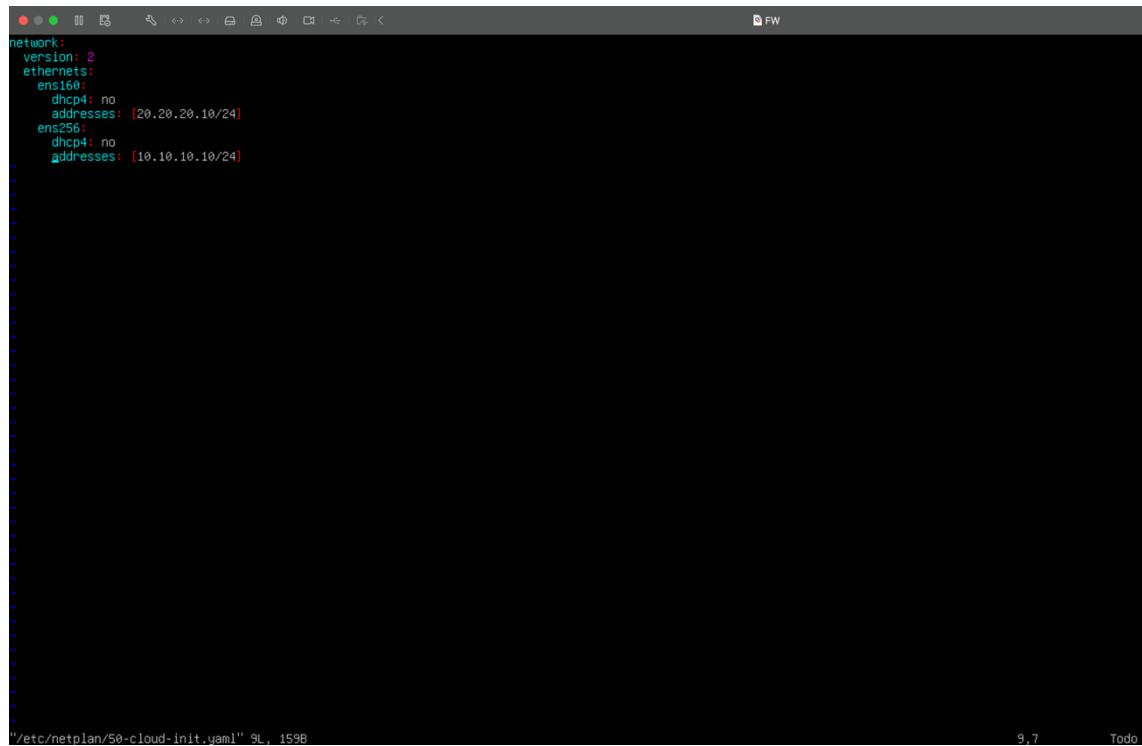
NOTA: Verificar el nombre de los interfaces asignados, y la red a la que están conectados.



En primer lugar, procedemos a configurar el direccionamiento de la máquina Ubuntu FW, para ello entramos con el siguiente comando en el documento donde se definen estas direcciones:

```
sudo su - #lo primero para tener permisos de root  
vi /etc/netplan/50-cloud-init.yaml #en el caso de esta distribución Ubuntu
```

Y una vez dentro se definen las rutas como en la imagen anterior:



The screenshot shows a terminal window with the following content:

```
network:  
  version: 2  
  ethernets:  
    ens160:  
      dhcp4: no  
      addresses: [20.20.20.10/24]  
    ens256:  
      dhcp4: no  
      addresses: [10.10.10.10/24]
```

The terminal window has a dark background and light-colored text. The title bar says "FW". At the bottom, it shows the file path "/etc/netplan/50-cloud-init.yaml" and its size "9L, 159B". On the right side, there are status indicators "9,7" and "Todo".

Aplicamos los cambios y podemos observar las interfaces del sistema con las nuevas direcciones.

```

System information as of mar 23 dic 2025 13:34:45 UTC
System load: 0.0           Swap usage: 0%          Users logged in: 0
Usage of /: 27.0% of 9.75GB   Temperature: 11758.9 C
Memory usage: 6%           Processes: 238

El mantenimiento de seguridad expandido para Applications está desactivado
Se pueden aplicar 74 actualizaciones de forma inmediata.
16 de estas son actualizaciones de seguridad estándares.
Para ver estas actualizaciones adicionales, ejecute: apt list --upgradable
Active ESM Apps para recibir futuras actualizaciones de seguridad adicionales.
Vea https://ubuntu.com/esm o ejecute «sudo pro status»
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

gabriel@fw:~$ ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 20.20.20.10 netmask 255.255.255.0 broadcast 20.20.20.255
                inet6 fe80::20c:29ff:fea1:7188 prefixlen 64 scopcid 0x20<link>
                    ether 00:0c:29:1a:71:88 txqueuelen 1000 (Ethernet)
                        RX packets 0 bytes 0 (0.0 B)
                        RX errors 0 dropped 0 overruns 0 frame 0
                        TX packets 8 bytes 656 (656.0 B)
                        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
                        device interrupt 45 memory 0x3fe00000-3fe20000

ens256: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.10.10.10 netmask 255.255.255.0 broadcast 10.10.10.255
                inet6 fe80::20c:29ff:fea1:7192 prefixlen 64 scopcid 0x20<link>
                    ether 00:0c:29:1a:71:92 txqueuelen 1000 (Ethernet)
                        RX packets 0 bytes 0 (0.0 B)
                        RX errors 0 dropped 0 overruns 0 frame 0
                        TX packets 8 bytes 656 (656.0 B)
                        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
                        device interrupt 47 memory 0x3e600000-3e620000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
                inet6 ::1 prefixlen 128 scopcid 0x10<host>
                    loop txqueuelen 1000 (Local Loopback)
                        RX packets 120 bytes 8840 (8.8 KB)
                        RX errors 0 dropped 0 overruns 0 frame 0
                        TX packets 120 bytes 8840 (8.8 KB)
                        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
                        device interrupt 47 memory 0x3e600000-3e620000

gabriel@fw:~$ 

```

Realizándolo de forma similar en la maquina externa la cual también es una Ubuntu de forma que se puede observar de la siguiente forma como quedan sus interfaces junto con su archivo netplan correspondiente.

```

root@externa:~# ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 20.20.20.150 netmask 255.255.255.0 broadcast 20.20.20.255
                inet6 fe80::20c:29ff:fea1:899d prefixlen 64 scopcid 0x20<link>
                    ether 00:0c:29:51:89:9d txqueuelen 1000 (Ethernet)
                        RX packets 222 bytes 24668 (24.6 KB)
                        RX errors 0 dropped 0 overruns 0 frame 0
                        TX packets 818 bytes 61468 (61.4 KB)
                        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
                        device interrupt 45 memory 0x3fe00000-3fe20000

ens256: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        inet6 fe80::20c:29ff:fea1:89a7 prefixlen 64 scopcid 0x20<link>
                    ether 00:0c:29:51:89:a7 txqueuelen 1000 (Ethernet)
                        RX packets 1373 bytes 1625710 (1.6 MB)
                        RX errors 0 dropped 0 overruns 0 frame 0
                        TX packets 787 bytes 66933 (66.0 KB)
                        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
                        device interrupt 47 memory 0x3e600000-3e620000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
                inet6 ::1 prefixlen 128 scopcid 0x10<host>
                    loop txqueuelen 1000 (Local Loopback)
                        RX packets 340 bytes 87637 (87.6 KB)
                        RX errors 0 dropped 0 overruns 0 frame 0
                        TX packets 340 bytes 87637 (87.6 KB)
                        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@externa:~#

```

```
network:
  version: 2
  ethernets:
    ens168:
      dhcp4: no
      addresses: [20.20.20.150/24]
      routes:
        - to: 0.0.0.0/0
          via: 20.20.20.10
    ens256:
      dhcp4: yes
```

"/etc/netplan/50-cloud-init.yaml" 11L, 194B 11,16 Todo

Y por último nuestra maquina interna la cual es una Rocky Linux que utiliza NetworkManager en lugar de netplan que tiene el siguiente archivo de configuración de la interfaz:

```
[connection]
id=ens168
uuid=1259e1cb-1e59-43c8-a69e-61f87f5831e9
type=ethernet
interface-name=ens168

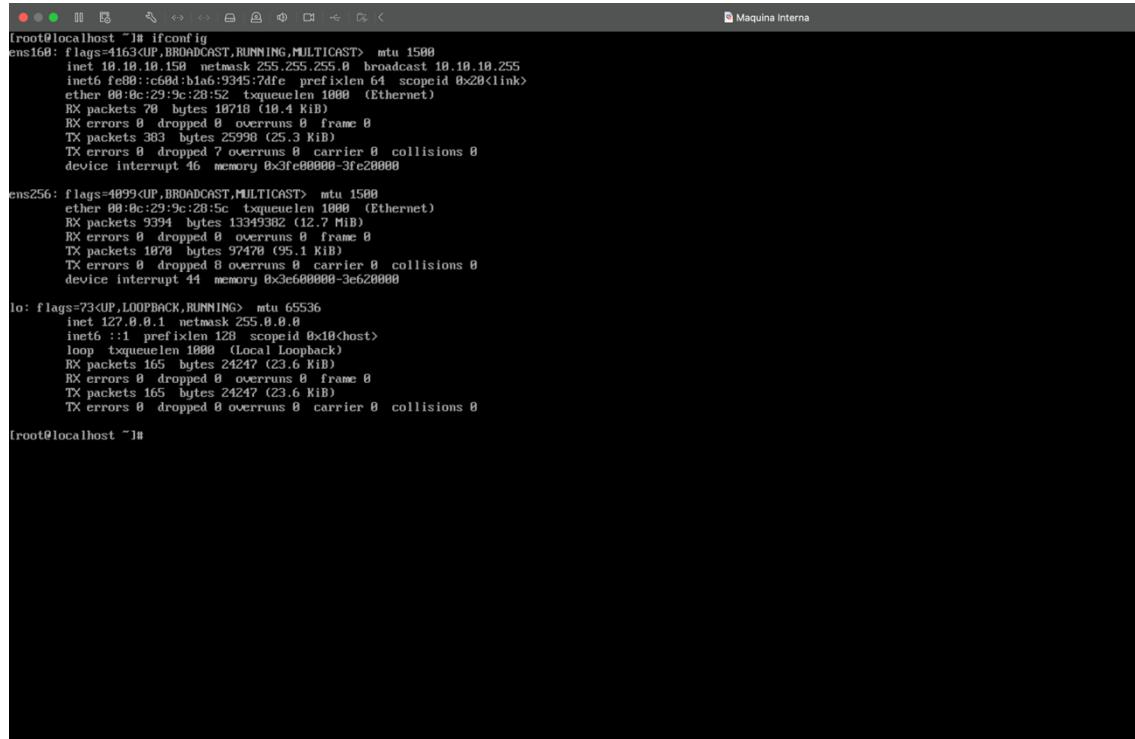
[ethernet]
[ipv4]
address=10.10.10.150/24
dns=8.8.8.8
gateway=10.10.10.10
method=manual

[ipv6]
addr-gen-mode=default
method=auto

[proxy]
```

"/etc/NetworkManager/system-connections/ens168.nmconnection" 19L, 244B 1,1 Todo

Y posteriormente, podemos comprobar que efectivamente se ha aplicado el cambio correctamente:



A screenshot of a terminal window titled "Maquina Interna". The window displays the output of the "ifconfig" command. The output shows three network interfaces: ens160, ens256, and lo. The "ens160" interface has an IP address of 10.10.10.158 and is connected to an Ethernet card. The "ens256" interface is a loopback interface with an IP address of 127.0.0.1. The "lo" interface is also a loopback interface. The terminal prompt at the bottom is "[root@localhost ~]#".

```
[root@localhost ~]# ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.10.10.158 brd 255.255.255.0 broadcast 10.10.10.255
      inet6 fe80::c66d:b1a6:9345:7dfe brd ff02::1 prefixlen 64 scopeid 0x20<link>
        ether 00:0c:29:9c:28:52 txqueuelen 1000  (Ethernet)
          RX packets 78 bytes 18718 (18.4 Kib)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 363 bytes 25998 (25.3 Kib)
          TX errors 0 dropped 7 overruns 0 carrier 0 collisions 0
        device interrupt 46 memory 0xfed00000-0xfed20000

ens256: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
      ether 00:0c:29:9c:28:5c txqueuelen 1000  (Ethernet)
        RX packets 9394 bytes 13349382 (12.7 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 1078 bytes 97478 (95.1 KiB)
        TX errors 0 dropped 8 overruns 0 carrier 0 collisions 0
      device interrupt 44 memory 0xfed00000-0xfed20000

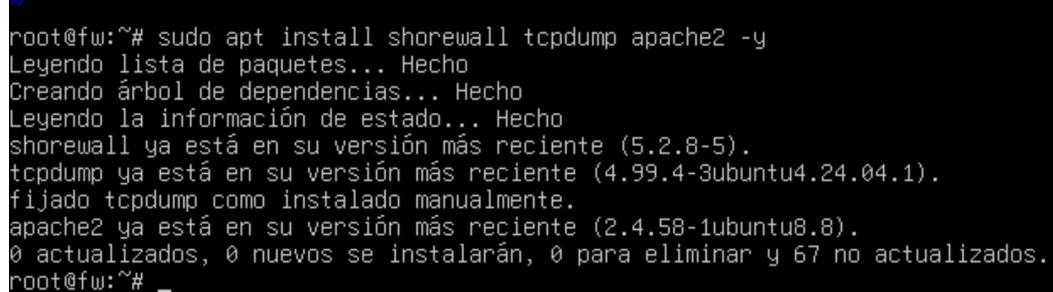
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 brd 255.0.0.0 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000  (Local Loopback)
        RX packets 165 bytes 24247 (23.6 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 165 bytes 24247 (23.6 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@localhost ~]#
```

9- Instalar shorewall en el equipo FW. También se recomienda instalar la herramienta tcpdump y algún servicio (por ejemplo, Apache) en las tres máquinas.

Una vez realizada esa configuración inicial se procede a instalar shorewall, en la maquina fw junto con el tcpdum y el apache. Para eso se escriben los siguientes comandos:

```
apt-get update
apt install shorewall tcpdump apache2 -y
```



A screenshot of a terminal window showing the output of the "sudo apt install shorewall tcpdump apache2 -y" command. The output indicates that shorewall, tcpdump, and apache2 are already installed in their latest versions. The terminal prompt at the bottom is "root@fw:~#".

```
root@fw:~# sudo apt install shorewall tcpdump apache2 -y
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
shorewall ya está en su versión más reciente (5.2.8-5).
tcpdump ya está en su versión más reciente (4.99.4-3ubuntu4.24.04.1).
fijado tcpdump como instalado manualmente.
apache2 ya está en su versión más reciente (2.4.58-1ubuntu8.8).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 67 no actualizados.
root@fw:~#
```

También, se instala en las otras maquinas tcpdump y apache. En la Ubuntu de forma similar con:

```
apt-get update  
apt install tcpdump apache2 -y
```

Y en la Rocky Linux se instala con:

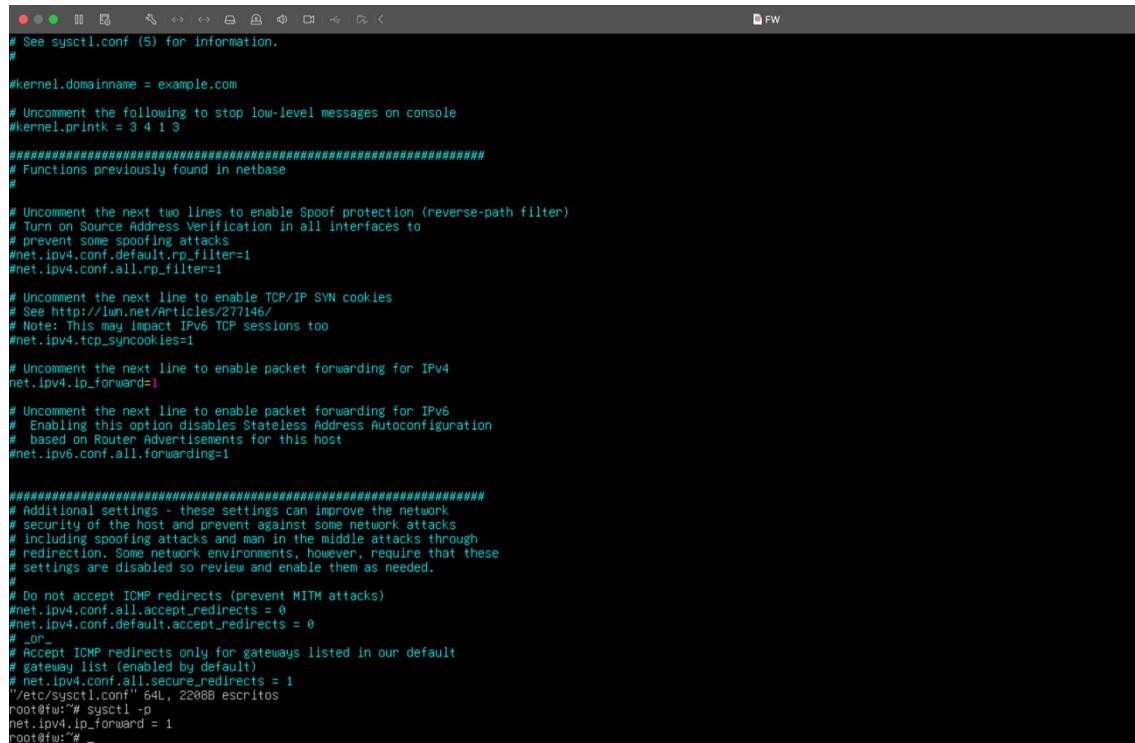
```
dnf install httpd tcpdump
```

Y lo arrancamos para que este disponible para futuros apartados

```
root@localhost ~# dnf install httpd tcpdump  
Última comprobación de caducidad de metadatos hecha hace 3:46:18, el mar 23 dic 2025 14:28:56.  
El paquete httpd-2.4.63-4.el18.1.2.aarch64 ya está instalado.  
El paquete tcpdump-14:4.99.4-10.el18.aarch64 ya está instalado.  
Dependencias resueltas.  
Nada por hacer.  
[listo]  
root@localhost ~# systemctl status httpd  
● httpd.service - The Apache HTTP Server  
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)  
  Active: active (running) since Tue 2025-12-23 14:31:46 CET; 3h 43min ago  
  Invocation: 47f0fa917669486bc8a995a1d40423b  
    Docs: man:httpd.service(8)  
 Main PID: 1982 (httpd)  
   Status: "Total requests: 1; Idle/Busy workers 100/0; Requests/sec: 7.46e-05; Bytes served/sec: 0 B/sec"  
 Tasks: 177 (limit: 18685)  
 Memory: 13.2M (peak: 13.7M)  
 CPU: 28.81%  
 CGroup: /system.slice/httpd.service  
         └─1982 /usr/sbin/httpd -DFOREGROUND  
             ├─1983 /usr/sbin/httpd -DFOREGROUND  
             ├─1984 /usr/sbin/httpd -DFOREGROUND  
             ├─1985 /usr/sbin/httpd -DFOREGROUND  
             └─1986 /usr/sbin/httpd -DFOREGROUND  
  
dic 23 14:31:46 localhost.localdomain systemd[1]: Starting httpd.service - The Apache HTTP Server...  
dic 23 14:31:46 localhost.localdomain (httpd)[1982]: httpd.service: Referenced but unset environment variable evaluates to an empty string: OPTIONS  
dic 23 14:31:46 localhost.localdomain httpd[1982]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using localhost.localdomain for ServerName  
dic 23 14:31:46 localhost.localdomain httpd[1982]: Server configured, listening on: port 80  
dic 23 14:31:46 localhost.localdomain systemd[1]: Started httpd.service - The Apache HTTP Server.  
Lines 1-22/22 (END)
```

10- Configurar los interfaces de red de los tres equipos y default GW en los hosts interno y externo. Reiniciar los servicios de red y comprobar que somos capaces de alcanzar las máquinas interna y externa desde el cortafuegos (FW). Habilitar el encaminamiento en el cortafuegos (net.ipv4.ip_forward=1 en el archivo /etc/sysctl.conf) y comprobar que es posible comunicar las máquinas interna y externa sin restricciones. Levantar algún servicio en los equipos y comprobar que es posible acceder remotamente.

Se habilita el ipforward y se comprueba la comunicación entre la maquina interna y externa.



```
# See sysctl.conf (5) for information.
#
#kernel.domainname = example.com
# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3
#####
# Functions previously found in netbase
#
# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

#####
# Additional settings - these settings can improve the network
# security of the host and prevent against some network attacks
# including spoofing attacks and man in the middle attacks through
# redirection. Some network environments, however, require that these
# settings are disabled so review and enable them as needed.
#
# Do not accept ICMP redirects (prevent MITM attacks)
#net.ipv4.conf.all.accept_redirects = 0
#net.ipv4.conf.default.accept_redirects = 0
#_OR_
# Accept ICMP redirects only for gateways listed in our default
# gateway list (enabled by default)
# net.ipv4.conf.all.secure_redirects = 1
#'/etc/sysctl.conf' 64L, 2208B escritos
root@fw:~# sysctl -p
net.ipv4.ip_forward = 1
root@fw:~#
```

```

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
#   Enabling this option disables Stateless Address Autoconfiguration
#   based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

#####
# Additional settings - these settings can improve the network
# security of the host and prevent against some network attacks
# including spoofing attacks and man in the middle attacks through
# redirection. Some network environments, however, require that these
# settings are disabled so review and enable them as needed.
#
# Do not accept ICMP redirects (prevent MITM attacks)
#net.ipv4.conf.all.accept_redirects = 0
#net.ipv4.conf.default.accept_redirects = 0
#_OR_
# Accept ICMP redirects only for gateways listed in our default
# gateway list (enabled by default)
# net.ipv4.conf.all.secure_redirects = 1
/etc/sysctl.conf" 64L, 22088 escritos
root@fw:~$ sysctl -p
net.ipv4.ip_forward = 1
root@fw:~# ping 10.10.10.150
PING 10.10.10.150 (10.10.10.150) 56(84) bytes of data.
64 bytes from 10.10.10.150: icmp_seq=1 ttl=64 time=1.43 ms
64 bytes from 10.10.10.150: icmp_seq=2 ttl=64 time=0.774 ms
^C
--- 10.10.10.150 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.774/1.103/1.433/0.329 ms
root@fw:~# ping 20.20.20.150
PING 20.20.20.150 (20.20.20.150) 56(84) bytes of data.
64 bytes from 20.20.20.150: icmp_seq=1 ttl=64 time=1.31 ms
64 bytes from 20.20.20.150: icmp_seq=2 ttl=64 time=1.00 ms
64 bytes from 20.20.20.150: icmp_seq=3 ttl=64 time=0.752 ms
^C
--- 20.20.20.150 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.752/1.021/1.311/0.228 ms
root@fw:~#"

root@localhost ~# ping 28.28.28.158
PING 28.28.28.158 (28.28.28.158) 56(84) bytes of data.
64 bytes from 28.28.28.158: icmp_seq=1 ttl=63 time=1.28 ms
64 bytes from 28.28.28.158: icmp_seq=2 ttl=63 time=1.42 ms
64 bytes from 28.28.28.158: icmp_seq=3 ttl=63 time=1.48 ms
64 bytes from 28.28.28.158: icmp_seq=4 ttl=63 time=1.38 ms
^C
--- 28.28.28.158 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 1.199/1.369/1.479/0.184 ms
root@localhost ~# ifconfig
ens168: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.10.10.150 netmask 255.255.255.0 broadcast 10.10.255
                inet6 fe80::c60d:bfa6:9345:7fe prefixlen 64 scopeid 0x20<link>
                  ether 00:0c:29:9c:28:52 txqueuelen 1000 (Ethernet)
                    RX packets 228 bytes 24424 (23.8 kB)
                    RX errors 0 dropped 0 overruns 0 frame 0
                    TX packets 672 bytes 49174 (46.8 kB)
                    TX errors 0 dropped 7 overruns 0 carrier 0 collisions 0
                    device interrupt 46 memory 0xb800000-3fe20000

ens256: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        ether 00:0c:29:9c:28:5c txqueuelen 1000 (Ethernet)
          RX packets 9394 bytes 13349382 (12.7 MB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 1878 bytes 97478 (95.1 kB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
          device interrupt 44 memory 0xb800000-3fe20000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
                inet6 ::1 prefixlen 128 scopeid 0x10<host>
                  loop txqueuelen 1000 (Local Loopback)
                    RX packets 165 bytes 24247 (23.6 kB)
                    RX errors 0 dropped 0 overruns 0 frame 0
                    TX packets 165 bytes 24247 (23.6 kB)
                    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@externa:~# ping 10.10.10.150
PING 10.10.10.150 (10.10.10.150) 56(84) bytes of data.
64 bytes from 10.10.10.150: icmp_seq=1 ttl=63 time=1.61 ms
64 bytes from 10.10.10.150: icmp_seq=2 ttl=63 time=1.41 ms
64 bytes from 10.10.10.150: icmp_seq=3 ttl=63 time=2.35 ms
^C
--- 10.10.10.150 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 1.407/1.789/2.348/0.403 ms
root@externa:~# "

```

Una vez levantados los servicios se comprueban que funcionan al accederse entre la maquina interna y externa con el servicio de apache.

The image shows two terminal windows side-by-side. The left window, titled 'Maquina Externa', displays the Apache test page for a Rocky Linux system. It includes instructions for adding content to the webroot directory and mentions the use of Apache and Nginx. The right window, titled 'Maquina Interna', displays the Apache test page for an Ubuntu system, which provides information on document roots and reporting bugs. Both pages include footer links for the Apache Software Foundation and F5 Networks.

```
<li>Neither the <strong>Rocky Linux Project</strong> nor the <strong>Rocky Enterprise Software Foundation</strong> have anything to do with this website or its content.</li>
<li>The Rocky Linux Project nor the <strong>RESF</strong> have "hacked" this webserver: This test page is included with the distribution.</li>
</ul>
<p><a href="https://rockylinux.org/"><strong>Rocky Linux website</strong></a>.</p>
</div>
<div class='col-sm-12 col-md-6 col-md-6 col-md-offset-12'>
<div class='section'>
<h2>I am the admin, what do I do?</h2>
<p>You may now add content to the webroot directory for your software.</p>
<p><strong>For systems using the <a href="https://httpd.apache.org/">Apache Webserver</strong></a>: You can add content to the directory <code>/var/www/html</code>. Until you do so, people visiting your website will see this page. If you would like this page to not be shown, follow the instructions in <code>/etc/httpd/conf.d/welcome.conf</code>.</p>
<p><strong>For systems using <a href="https://nginx.org">Nginx</strong></a>; You can add your content in a location of your choice and edit the <code>root</code> configuration directive in <code>/etc/nginx/nginx.conf</code>.</p>
<div id="logos">
<a href="https://rockylinux.org/" id="rocky-poweredby"></a> <!-- Rocky -->
 <!-- webserver -->
</div>
</div>
</div>

<footer class="col-sm-12">
<a href="https://apache.org">Apache®</a> is a registered trademark of <a href="https://apache.org">the Apache Software Foundation</a> in the United States and/or other countries.<br />
<a href="https://nginx.org">NGINX®</a> is a registered trademark of <a href="https://f5.com">F5 Networks, Inc.</a>.
</footer>
</body>
</html>
root@externa:~# curl 10.10.10.158
```



```
default configuration.
</li>
</div>

<div class="section_header">
<div id="docroot"></div>
Document Roots
</div>

<div class="content_section_text">
<p>By default, Ubuntu does not allow access through the web browser to <em>any</em> file outside of those located in <tt>/var/www</tt>, <a href="http://httpd.apache.org/docs/2.4/mod/mod_userdir.html" rel="nofollow">public_html</a> directories (when enabled) and <tt>/usr/share</tt> (for web applications). If your site is using a web document root located elsewhere (such as in <tt>/srv</tt>) you may need to whitelist your document root directory in <tt>/etc/apache2/apache2.conf</tt>.
</p>
<p>The default Ubuntu document root is <tt>/var/www/html</tt>. You can make your own virtual hosts under <tt>/var/www</tt>.
</p>
<div class="section_header">
<div id="bugs"></div>
Reporting Problems
</div>
<div class="content_section_text">
<p>Please use the <tt>ubuntu-bug</tt> tool to report bugs in the Apache2 package with Ubuntu. However, check <a href="https://bugs.launchpad.net/ubuntu/+source/apache2" rel="nofollow">existing bug reports</a> before reporting a new bug.
</p>
<p>Please report bugs specific to modules (such as PHP and others) to their respective packages, not to the web server itself.
</p>
</div>
</div>
<div class="validator">
</div>
</body>
</html>
root@localhost ~# curl 20.20.20.158
```

11- Describir unas políticas básicas en el cortafuegos

Shorewall: se acepta el tráfico desde la red interna al exterior (saliente), se impide tráfico entrante desde el exterior, se acepta tráfico saliente desde el cortafuegos hacia cualquier sitio. Se acepta tráfico entrante al cortafuegos desde la red interna. Activar las reglas (systemctl start shorewall) y verificar su correcto funcionamiento.

Primero editaremos desde la maquina FW los siguientes archivos, definimos las zonas con el siguiente comando:

```
vim /etc/shorewall/zones
```

De forma que quede de la siguiente forma

```
fw      firewall
net    ipv4
loc    ipv4
~
```

Después asignamos las interfaces las cuales se encuentran en la siguiente ruta:

```
vim /etc/shorewall/interfaces
```

#ZONE	INTERFACE	BROADCAST	OPTIONS
net	ens160	detect	dhcp
loc	ens256	detect	dhcp

Y definimos la política tal y como la describe el enunciado editando el archivo correspondiente:

```
vim /etc/shorewall/policy
```

#	SOURCE	DEST	POLICY	LOG	LIMIT:BURST	CONNLIMIT
1	loc	net	ACCEPT			
2	loc	fw	ACCEPT			
3	fw	all	ACCEPT			
4	net	all	DROP	info		
5	all	all	DROP	info		
6	~					
7						

Una vez declarados los tres archivos se procede a validar y arrancar el servicio con los siguientes comandos.

```
shorewall check  
systemctl enable shorewall  
systemctl start shorewall
```

```
root@fw:~# shorewall check  
Checking using Shorewall 5.2.8...  
Processing /etc/shorewall/params ...  
Processing /etc/shorewall/shorewall.conf...  
Loading Modules...  
Compiling /etc/shorewall/zones...  
Compiling /etc/shorewall/interfaces...  
Determining Hosts in Zones...  
Locating Action Files...  
Compiling /etc/shorewall/policy...  
Adding rules for DHCP  
Compiling TCP Flags filtering...  
Compiling Kernel Route Filtering...  
Compiling Martian Logging...  
Compiling MAC Filtration -- Phase 1...  
Compiling MAC Filtration -- Phase 2...  
Applying Policies...  
Shorewall configuration verified  
root@fw:~# systemctl enable shorewall.service  
Synchronizing state of shorewall.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.  
Executing: /usr/lib/lib/systemd/systemd-sysv-install enable shorewall  
Created symlink /etc/systemd/system/basic.target.wants/shorewall.service → /usr/lib/systemd/system/shorewall.service.  
root@fw:~#
```

Y los cambios se han debido aplicar, lo que podemos observar con los siguientes pings.

Desde la Interna -> Exterior (debería funcionar)

```
[root@localhost ~]# ping 20.20.20.150
PING 20.20.20.150 (20.20.20.150) 56(84) bytes of data.
64 bytes from 20.20.20.150: icmp_seq=1 ttl=63 time=1.21 ms
64 bytes from 20.20.20.150: icmp_seq=2 ttl=63 time=1.23 ms
64 bytes from 20.20.20.150: icmp_seq=3 ttl=63 time=0.495 ms
^C
--- 20.20.20.150 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.495/0.979/1.232/0.342 ms
```

Desde la Interna -> FW (debería funcionar)

```
[root@localhost ~]# ping 10.10.10.10
PING 10.10.10.10 (10.10.10.10) 56(84) bytes of data.
64 bytes from 10.10.10.10: icmp_seq=1 ttl=64 time=0.592 ms
64 bytes from 10.10.10.10: icmp_seq=2 ttl=64 time=1.26 ms
64 bytes from 10.10.10.10: icmp_seq=3 ttl=64 time=0.640 ms
^C
--- 10.10.10.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2078ms
rtt min/avg/max/mdev = 0.592/0.830/1.260/0.304 ms
[root@localhost ~]#
```

Desde la Externa -> FW (debería fallar)

```
root@externa:~# ping 20.20.20.10
PING 20.20.20.10 (20.20.20.10) 56(84) bytes of data.
^C
--- 20.20.20.10 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2083ms
root@externa:~#
```

Desde la FW -> Exterior (debería funcionar)

```
root@fw:~# ping 20.20.20.150
PING 20.20.20.150 (20.20.20.150) 56(84) bytes of data.
64 bytes from 20.20.20.150: icmp_seq=1 ttl=64 time=0.757 ms
64 bytes from 20.20.20.150: icmp_seq=2 ttl=64 time=0.628 ms
^C
--- 20.20.20.150 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1021ms
rtt min/avg/max/mdev = 0.628/0.692/0.757/0.064 ms
root@fw:~#
```

12- Habilitar el enmascaramiento de direcciones desde la red interior hacia la red exterior. Verificar que todo el tráfico saliente se enmascara (puede usar el sniffer tcpdump). Se recomienda desactivar el GW en la máquina Externa y comprobar que sigue siendo posible alcanzar sus servicios desde el interior.

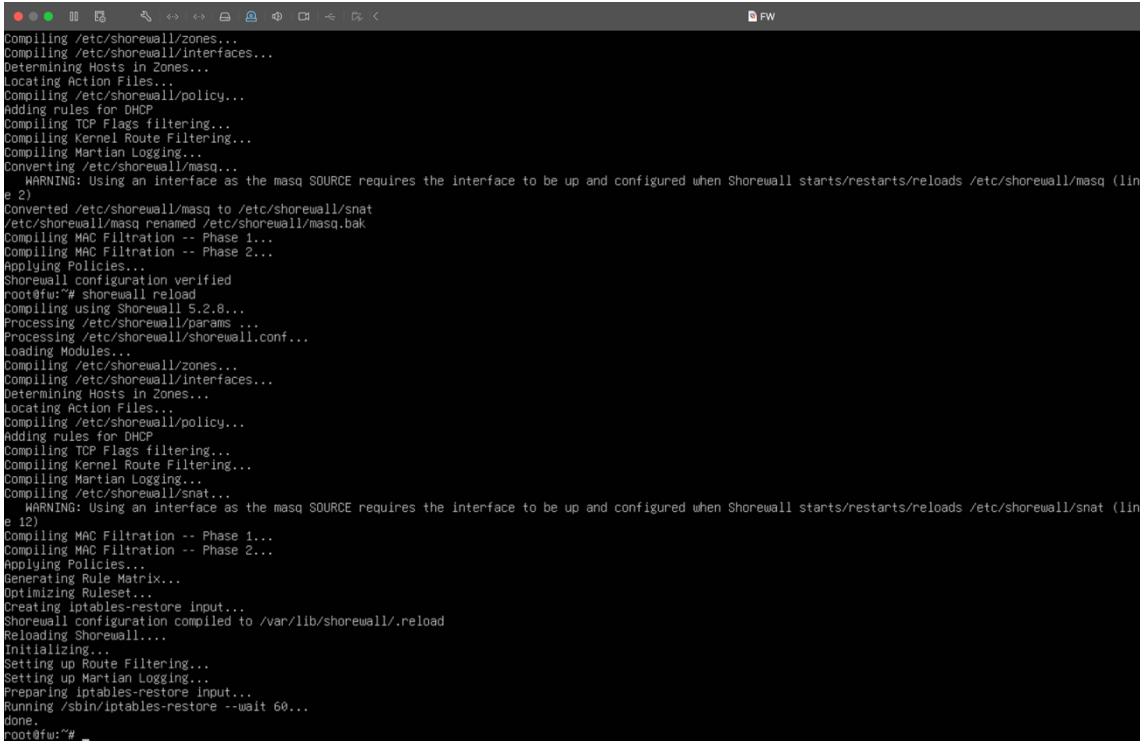
Se edita el archivo de configuración de enmascaramiento:

```
vim /etc/shorewall/masq
```

```
#INTERFACE      SOURCE
ens160         ens256
~
```

Se aplican y se guardan

```
shorewall check
shorewall reload
```



```
Compiling /etc/shorewall/zones...
Compiling /etc/shorewall/interfaces...
Determining Hosts in Zones...
Locating Action Files...
Compiling /etc/shorewall/policy...
Adding rules for DHCP...
Compiling TCP Flags filtering...
Compiling Kernel Route Filtering...
Compiling Martian Logging...
Compiling /etc/shorewall/masq...
, WARNING: Using an interface as the masq SOURCE requires the interface to be up and configured when Shorewall starts/restarts/reloads /etc/shorewall/masq (line 2)
Converted /etc/shorewall/masq to /etc/shorewall/snat
/etc/shorewall/masq renamed /etc/shorewall/masq.bak
Compiling MAC Filtration -- Phase 1...
Compiling MAC Filtration -- Phase 2...
Applying Policies...
Shorewall configuration verified
root@fw:~# Shorewall reload
Compiling using Shorewall 5.2.8...
Processing /etc/shorewall/params ...
Processing /etc/shorewall/shorewall.conf...
Loading Modules...
Compiling /etc/shorewall/zones...
Compiling /etc/shorewall/interfaces...
Determining Hosts in Zones...
Locating Action Files...
Compiling /etc/shorewall/policy...
Adding rules for DHCP...
Compiling TCP Flags filtering...
Compiling Kernel Route Filtering...
Compiling Martian Logging...
Compiling /etc/shorewall/snat...
, WARNING: Using an interface as the masq SOURCE requires the interface to be up and configured when Shorewall starts/restarts/reloads /etc/shorewall/snat (line 12)
Compiling MAC Filtration -- Phase 1...
Compiling MAC Filtration -- Phase 2...
Applying Policies...
Generating Rule Matrix...
Optimizing RuleSet...
Creating iptables-restore input...
Shorewall configuration compiled to /var/lib/shorewall/.reload
Reloading Shorewall...
Initializing...
Setting up Route Filtering...
Setting up Martian Logging...
Preparing iptables-restore input...
Running /sbin/iptables-restore --wait 60...
done.
root@fw:~#
```

Por último se verifica con tcpdump, en el Firewall, se ejecuta este comando para escuchar el tráfico en la interfaz externa:

```
tcpdump -i ens160 -n icmp
```

Y en el pc interno hacemos un ping al externo y vemos que en el echo request sale con la ip del firewall no con la de la maquina externa.

```

root@localhost ~# ping 28.28.28.159
From 18.18.18.10 icmp_seq=4 Destination Net Unreachable
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3068ms

PING 28.28.28.159 (28.28.28.159) 56(84) bytes of data.
64 bytes from 28.28.28.159: icmp_seq=1 ttl=63 time=4.88 ms
64 bytes from 28.28.28.159: icmp_seq=2 ttl=63 time=1.77 ms
64 bytes from 28.28.28.159: icmp_seq=3 ttl=63 time=0.728 ms
64 bytes from 28.28.28.159: icmp_seq=4 ttl=63 time=2.16 ms
64 bytes from 28.28.28.159: icmp_seq=5 ttl=63 time=1.75 ms
64 bytes from 28.28.28.159: icmp_seq=6 ttl=63 time=1.88 ms
64 bytes from 28.28.28.159: icmp_seq=7 ttl=63 time=0.938 ms
64 bytes from 28.28.28.159: icmp_seq=8 ttl=63 time=1.11 ms
64 bytes from 28.28.28.159: icmp_seq=9 ttl=63 time=1.27 ms
64 bytes from 28.28.28.159: icmp_seq=10 ttl=63 time=1.15 ms
...
--- 28.28.28.159 ping statistics ---
10 packets transmitted, 0 received, 0% packet loss, time 9896ms
rtt min/avg/max/mdev = 0.728/1.735/4.795/1.110 ms
[root@localhost ~]# ifconfig
ens160: flags=4163 mtu 1500
    inet 18.18.18.158 brd 255.255.255.0 bcast 18.18.18.255
        netmask 0xffffffff broadcast 18.18.18.255
        inetb 1880::c880:16a8:9345:7dfe brd 1880::ffff:ffff:ffff:ffff
        scopeid 0x20<link>
        ether 00:0c:29:9c:28:52 txqueuelen 1000 (Ethernet)
        RX packets 1249 bytes 158766 (155.8 KIB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 1577 bytes 128695 (125.6 KIB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
        device interrupt 46 memory 0xf0000000-3fc20000

ens256: flags=4199 mtu 1500
    ether 00:0c:29:9c:28:5c txqueuelen 1000 (Ethernet)
    RX packets 9394 bytes 13349382 (12.7 MIB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1878 bytes 97478 (95.1 KIB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 44 memory 0xe6000000-3c620000

lo: flags=73 mtu 65536
    inet 127.0.0.1 brd 127.0.0.1
        netmask 0xffffffff broadcast 127.0.0.1
        inet6 ::1 brd ::1
        scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 175 bytes 32933 (32.1 KIB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 175 bytes 32933 (32.1 KIB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
        device interrupt 0 memory 0xf0000000-3fc20000

[root@localhost ~]#
done.
root@fw:~# tcpdump -i ens160 -n icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens160, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:20:13.913721 IP 20.20.20.10 > 20.20.20.150: ICMP echo request, id 6, seq 1, length 64
14:20:13.915041 IP 20.20.20.150 > 20.20.20.10: ICMP echo reply, id 6, seq 1, length 64
14:20:14.913396 IP 20.20.20.10 > 20.20.20.150: ICMP echo request, id 6, seq 2, length 64
14:20:14.914225 IP 20.20.20.150 > 20.20.20.10: ICMP echo reply, id 6, seq 2, length 64
14:20:15.915326 IP 20.20.20.10 > 20.20.20.150: ICMP echo request, id 6, seq 3, length 64
14:20:15.915645 IP 20.20.20.150 > 20.20.20.10: ICMP echo reply, id 6, seq 3, length 64
14:20:16.922960 IP 20.20.20.10 > 20.20.20.150: ICMP echo request, id 6, seq 4, length 64
14:20:16.923827 IP 20.20.20.150 > 20.20.20.10: ICMP echo reply, id 6, seq 4, length 64
14:20:17.924778 IP 20.20.20.10 > 20.20.20.150: ICMP echo request, id 6, seq 5, length 64
14:20:17.926160 IP 20.20.20.150 > 20.20.20.10: ICMP echo reply, id 6, seq 5, length 64
14:20:18.927721 IP 20.20.20.10 > 20.20.20.150: ICMP echo request, id 6, seq 6, length 64
14:20:18.928484 IP 20.20.20.150 > 20.20.20.10: ICMP echo reply, id 6, seq 6, length 64
14:20:19.928553 IP 20.20.20.10 > 20.20.20.150: ICMP echo request, id 6, seq 7, length 64
14:20:19.928851 IP 20.20.20.150 > 20.20.20.10: ICMP echo reply, id 6, seq 7, length 64
14:20:21.002586 IP 20.20.20.10 > 20.20.20.150: ICMP echo request, id 6, seq 8, length 64
14:20:21.003000 IP 20.20.20.150 > 20.20.20.10: ICMP echo reply, id 6, seq 8, length 64
14:20:22.004139 IP 20.20.20.10 > 20.20.20.150: ICMP echo request, id 6, seq 9, length 64
14:20:22.004578 IP 20.20.20.150 > 20.20.20.10: ICMP echo reply, id 6, seq 9, length 64
14:20:23.006674 IP 20.20.20.10 > 20.20.20.150: ICMP echo request, id 6, seq 10, length 64
14:20:23.007123 IP 20.20.20.150 > 20.20.20.10: ICMP echo reply, id 6, seq 10, length 64

```

Y tal y como se propone en el enunciado se realiza la prueba del “GW desactivado”, se elimina en la maquina externa el Gateway con el comando:

ip route del default

Y desde la maquina interna se le hace el ping a la externa y se observa que funciona porque la Máquina Externa ve llegar un paquete desde la IP del Firewall (que está en su misma red). No necesita un Gateway para responderle a la otra maquina.

```
64 bytes from 20.20.20.150: icmp_seq=1491 ttl=63 time=1.33 ms
64 bytes from 20.20.20.150: icmp_seq=1492 ttl=63 time=1.35 ms
64 bytes from 20.20.20.150: icmp_seq=1493 ttl=63 time=1.34 ms
64 bytes from 20.20.20.150: icmp_seq=1494 ttl=63 time=1.93 ms
64 bytes from 20.20.20.150: icmp_seq=1495 ttl=63 time=1.43 ms
64 bytes from 20.20.20.150: icmp_seq=1496 ttl=63 time=1.00 ms
64 bytes from 20.20.20.150: icmp_seq=1497 ttl=63 time=1.45 ms
64 bytes from 20.20.20.150: icmp_seq=1498 ttl=63 time=1.23 ms
64 bytes from 20.20.20.150: icmp_seq=1499 ttl=63 time=1.48 ms
64 bytes from 20.20.20.150: icmp_seq=1500 ttl=63 time=1.23 ms
64 bytes from 20.20.20.150: icmp_seq=1501 ttl=63 time=1.64 ms
64 bytes from 20.20.20.150: icmp_seq=1502 ttl=63 time=1.58 ms
64 bytes from 20.20.20.150: icmp_seq=1503 ttl=63 time=2.74 ms
64 bytes from 20.20.20.150: icmp_seq=1504 ttl=63 time=1.12 ms
64 bytes from 20.20.20.150: icmp_seq=1505 ttl=63 time=1.18 ms
64 bytes from 20.20.20.150: icmp_seq=1506 ttl=63 time=1.13 ms
64 bytes from 20.20.20.150: icmp_seq=1507 ttl=63 time=1.36 ms
64 bytes from 20.20.20.150: icmp_seq=1508 ttl=63 time=0.438 ms
64 bytes from 20.20.20.150: icmp_seq=1509 ttl=63 time=1.25 ms
64 bytes from 20.20.20.150: icmp_seq=1510 ttl=63 time=1.64 ms
64 bytes from 20.20.20.150: icmp_seq=1511 ttl=63 time=1.33 ms
64 bytes from 20.20.20.150: icmp_seq=1512 ttl=63 time=1.12 ms
64 bytes from 20.20.20.150: icmp_seq=1513 ttl=63 time=0.982 ms
64 bytes from 20.20.20.150: icmp_seq=1514 ttl=63 time=0.818 ms
64 bytes from 20.20.20.150: icmp_seq=1515 ttl=63 time=0.891 ms
64 bytes from 20.20.20.150: icmp_seq=1516 ttl=63 time=2.49 ms
64 bytes from 20.20.20.150: icmp_seq=1517 ttl=63 time=0.997 ms
64 bytes from 20.20.20.150: icmp_seq=1518 ttl=63 time=1.56 ms
64 bytes from 20.20.20.150: icmp_seq=1519 ttl=63 time=1.43 ms
64 bytes from 20.20.20.150: icmp_seq=1520 ttl=63 time=0.994 ms
64 bytes from 20.20.20.150: icmp_seq=1521 ttl=63 time=0.722 ms
64 bytes from 20.20.20.150: icmp_seq=1522 ttl=63 time=1.32 ms
64 bytes from 20.20.20.150: icmp_seq=1523 ttl=63 time=1.02 ms
64 bytes from 20.20.20.150: icmp_seq=1524 ttl=63 time=1.26 ms
64 bytes from 20.20.20.150: icmp_seq=1525 ttl=63 time=0.982 ms
64 bytes from 20.20.20.150: icmp_seq=1526 ttl=63 time=1.34 ms
64 bytes from 20.20.20.150: icmp_seq=1527 ttl=63 time=1.48 ms
64 bytes from 20.20.20.150: icmp_seq=1528 ttl=63 time=1.44 ms
^C
--- 20.20.20.150 ping statistics ---
1528 packets transmitted, 1528 received, 0% packet loss, time 1538146ms
rtt min/avg/max/mdev = 0.438/1.549/159.243/4.124 ms
root@localhost ~#
```

- 13- Implantar una política restrictiva para el tráfico del FW:
todo el tráfico entrante, saliente y que atraviesa el FW será
rechazado. A continuación, se habilitará la salida a servicios
concretos (http, https, SMTP, ...). Configurar algún servicio
de port forwarding (como Apache) que será visible desde el
exterior y ofrecido por la máquina de la red interna.
Configurar un acceso al cortafuegos por SSH desde una
dirección externa concreta. Establecer el
redireccionamiento de todo el tráfico DNS saliente para que
se reenvíe a un servidor de DNS externo concreto. Para
realizar este apartado se recomienda estudiar los ejemplos
de reglas shorewall presentes en el manual de esta
aplicación (man shorewall-rules).

Para conseguir lo que se nos plantea en el apartado debemos en primer lugar, cambiar las políticas en el siguiente archivo:

```
vim /etc/shorewall/policy
```

De tal forma que queden así:

```
#SOURCE DEST  POLICY LOG    LIMIT:BURST      CONNLIMIT
fw    all    REJECT  Info
net   all    DROP    Info
all   all    REJECT  Info
```

También hay que editar el archivo rules, el cual se edita con el siguiente comando:

```
vim /etc/shorewall/rules
```

```
#ACTION SOURCE  DEST      PROTO    DEST_PORT
ACCEPT   loc     net      tcp      80,443
ACCEPT   loc     net      tcp      25
~
```

Acceptando tanto los puertos HTTP, como HTTPS y SMTP. Asimismo habilitamos el port-forwarding de la siguiente forma:

```

#ACTION SOURCE DEST PROTO DEST_PORT
ACCEPT loc net tcp 80,443
ACCEPT loc net tcp 25
DNAT net loc:10.10.10.150      tcp 80
~
```

Y por ultimo le añadimos las reglas para que acepte el SSH y acate la redireccion del DNS.

```

#ACTION SOURCE DEST PROTO DEST_PORT
ACCEPT loc net tcp 80,443
ACCEPT loc net tcp 25
DNAT net loc:10.10.10.150      tcp 80
ACCEPT net:20.20.20.150 fw tcp 22
DNAT loc net:8.8.8.8      tcp 53
DNAT loc net:8.8.8.8      udp 53_
~
```

Y lo comprobamos y aplicamos con:

```

shorewall check
shorewall restart
```

```

root@fw:~# shorewall check
Checking using Shorewall 5.2.8...
Processing /etc/shorewall/params ...
Processing /etc/shorewall/shorewall.conf...
Loading Modules...
Compiling /etc/shorewall/zones...
Compiling /etc/shorewall/interfaces...
Determining Hosts in Zones...
Locating Action Files...
Compiling /etc/shorewall/policy...
Adding rules for DHCP
Compiling TCP Flags filtering...
Compiling Kernel Route Filtering...
Compiling Martian Logging...
Compiling /etc/shorewall/snat...
... WARNING: Using an interface as the masq SOURCE requires the interface to be up and configured when Shorewall starts/restarts/reloads /etc/shorewall/snat (line 12)
Compiling MAC Filtration -- Phase 1...
Compiling /etc/shorewall/rules...
Compiling MAC Filtration -- Phase 2...
Applying Policies...
Shorewall configuration verified
root@fw:~#
```

Partes OPCIONALES

14- Vamos a trabajar con la sintaxis nativa del módulo Nftables (nft). Para ello realizaremos las siguientes tareas:

a) Habilitar el uso de nftables. Normalmente ya contamos con el módulo y el interface de comandos (este último, de no existir, se instala con el paquete nftables). La activación de nftables se realiza como si fuera un nuevo servicio (`systemctl enable nftables` && `systemctl start nftables`). Ojo: si previamente están activos otros servicios de filtrado (`ufw`, `firewalld`, ...), deben desactivarse.

Antes de comenzar, debemos de desactivar iptables, firewalld o scripts iptables:

```
sudo systemctl stop ufw  
sudo systemctl disable ufw
```

```
sudo systemctl stop firewalld  
sudo systemctl disable firewalld
```

```
iptables -F
```

Instalamos sino viene instalado el paquete:

```
sudo apt update  
sudo apt install nftables -y
```

Inicializamos el servicio:

```
systemctl enable nftables  
systemctl start nftables
```

b) Identificar la tabla predefinida (inet filter), con sus cadenas, políticas y posibles reglas (nft list ruleset). Identificar también dónde se encuentra el archivo de esta tabla nftables (el que se activa al iniciar el servicio).

Para visualizar las reglas actuales:

```
nft list ruleset
```

```
root@server:~# nft list ruleset
table inet filter {
    chain input {
        type filter hook input priority filter; policy accept;
    }

    chain forward {
        type filter hook forward priority filter; policy accept;
    }

    chain output {
        type filter hook output priority filter; policy accept;
    }
}
```

La ruta se encuentra en /etc/nftables.conf:

```
root@server:~# cat /etc/nftables.conf
#!/usr/sbin/nft -f

flush ruleset

table inet filter {
    chain input {
        type filter hook input priority filter;
    }
    chain forward {
        type filter hook forward priority filter;
    }
    chain output {
        type filter hook output priority filter;
    }
}
```

c) Establecer una política de drop para todo el tráfico entrante y saliente.

Creamos la tabla *filter* (a diferencia que con iptables, es necesario crear las tablas y cadenas antes de asignar políticas), con valor *inet*, que vale para IPv4 o IPv6:

```
nft add table inet filter
```

Creamos las cadenas con políticas *DROP*:

```
nft add chain inet filter input { type filter hook input priority 0 \;
policy drop \; }
nft add chain inet filter output { type filter hook output priority 0 \;
policy drop \; }
nft add chain inet filter forward { type filter hook forward priority
0 \; policy drop \; }
```

Para comprobar visualmente:

```
nft -a list ruleset
root@server:~# nft -a list ruleset
table inet filter { # handle 3
    chain input { # handle 1
        type filter hook input priority filter; policy drop;
    }

    chain forward { # handle 2
        type filter hook forward priority filter; policy drop;
    }

    chain output { # handle 3
        type filter hook output priority filter; policy drop;
    }
}
```

d) Añadir reglas para permitir la entrada y salida al interface de loopback (lo).

```
nft add rule inet filter input iifname "lo" accept
nft add rule inet filter output oifname "lo" accept
```

e) Añadir reglas para permitir conexiones HTTP y HTTPS salientes (iniciadas desde el equipo hacia cualquier IP externa).

1# Reglas Statefull (queremos aceptar conexiones ya aceptadas o relacionadas:

```
nft add rule inet filter input ct state established,related accept  
nft add rule inet filter output ct state established,related accept
```

2# Permitir iniciar conexiones Web hacia el exterior:

```
nft add rule inet filter output tcp dport { 80, 443 } ct state new  
accept
```

f) Añadir reglas para permitir la conexión a nuestro servidor SSH desde cualquier equipo que se encuentre en la misma red el nuestro.

Para ver la subred usar ifconfig o ip addr y luego ejecutar esto:

```
inet 172.22.0.140/24 metric 100 brd 172.22.0.255
```

```
nft add rule inet filter input ip saddr 172.22.0.0/24 tcp dport 22  
ct state new accept
```

g) Verificar la política definida y salvar la nueva política para que se active en el arranque.

Para verificar la política usaremos el comando:

```
nft list ruleset
```

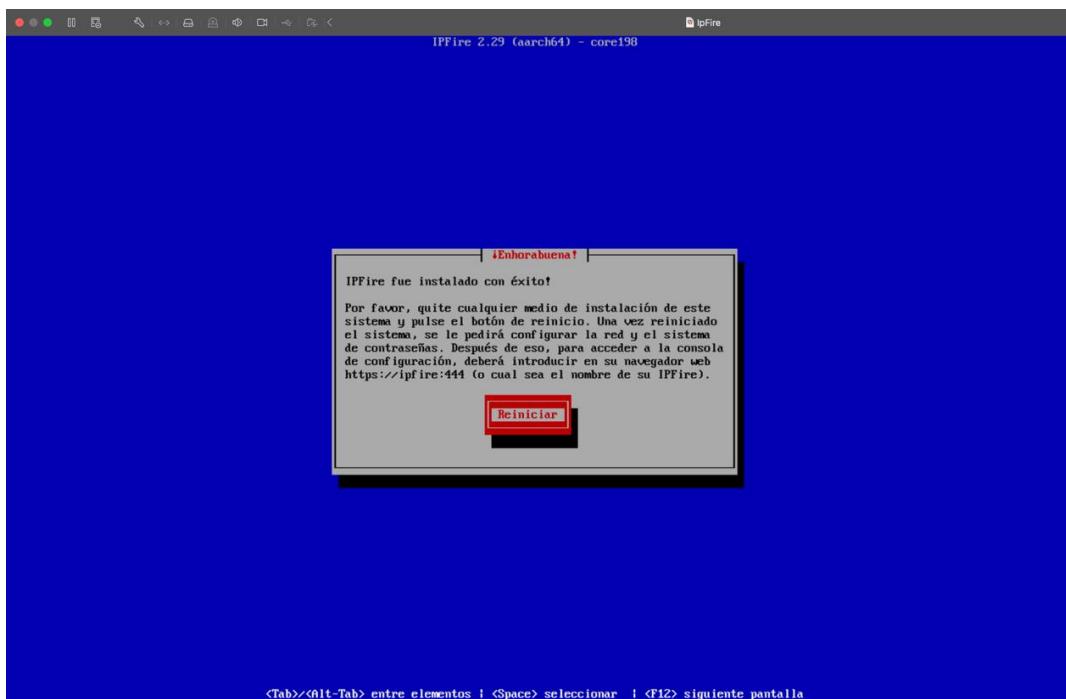
Para guardar “sobrescribiremos” la ruta mencionada en el apartado b:

```
nft list ruleset > /etc/nftables.conf
```

Ahora si se reinicia el servicio o el equipo, las reglas permanecen.

15. Instalar una máquina virtual con la distribución Linux IPFire_ (<https://www.ipfire.org/>). Estudiar sus funcionalidades y realizar algunas tareas de configuración a través de su panel de control (WUI). Indicar también algunos add-ons que pueden instalarse para incrementar sus funcionalidades.

Una vez descargada la imagen de la página oficial se procede a la instalación con su posterior configuración.



```

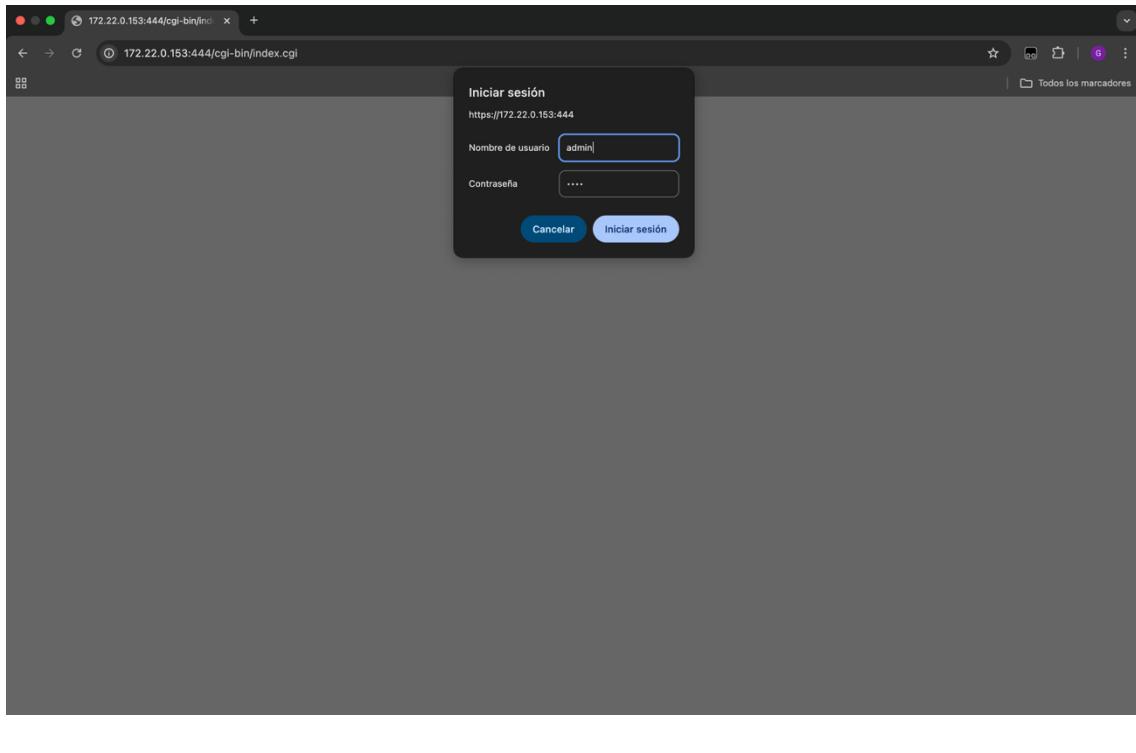
Stopping kernel log daemon... [ OK ]
Stopping system log daemon... [ OK ]
Bringing up the loopback interface... [ OK ]
Setting hostname to ipfire.localdomain... [ OK ]
Setting up firewall... [ OK ]
INIT: Entering runlevel: 3
Starting vnsstat...
No interfaces found in database, adding available interfaces...
Interface "green0" added with 1000 Mbit bandwidth limit.
Interface "red0" added with 1000 Mbit bandwidth limit.
-> 2 new interfaces found.
Limits can be modified using the configuration file. See "man vnstat.conf".
Unwanted interfaces can be removed from monitoring with "vnstat --remove".
Starting kernel log daemon... [ OK ]
Starting system log daemon... [ OK ]
Saving Bootlog...
Starting Unbound DNS Proxy...
Starting ACPI daemon...
Enabling S.M.A.R.T.:
Bringing up the green0 interface...
Adding IPv4 address 172.22.0.153 to the green0 interface...
Bringing up the red0 interface...
Starting dhcpcd on the red0 interface... [ OK ]

Stopping system log daemon... [ OK ]
Bringing up the loopback interface... [ OK ]
Setting hostname to ipfire.localdomain... [ OK ]
Setting up firewall... [ OK ]
INIT: Entering runlevel: 3
Starting vnsstat...
No interfaces found in database, adding available interfaces...
Interface "green0" added with 1000 Mbit bandwidth limit.
Interface "red0" added with 1000 Mbit bandwidth limit.
-> 2 new interfaces found.
Limits can be modified using the configuration file. See "man vnstat.conf".
Unwanted interfaces can be removed from monitoring with "vnstat --remove".
Starting kernel log daemon... [ OK ]
Starting system log daemon... [ OK ]
Saving Bootlog...
Starting Unbound DNS Proxy...
Starting ACPI daemon...
Enabling S.M.A.R.T.:
Bringing up the green0 interface...
Adding IPv4 address 172.22.0.153 to the green0 interface...
Bringing up the red0 interface...
Starting dhcpcd on the red0 interface... [ OK ]
    DHCP Assigned Settings for red0:
    IP Address: 172.22.0.154
    Hostname: ipfire
    Subnet Mask: 255.255.255.0
    Default Gateway: 172.22.0.2
    DNS Server: 172.22.0.2
Adding static routes...
Adding static routes...
Mounting network file systems...
Starting the Cyrus SASL Server...
Setting time on boot...
Error resolving @.ipfire.pool.ntp.org: Name or service not known (-2)
Error resolving 1.ipfire.pool.ntp.org: Name or service not known (-2)
Starting ntpd...
Generating SSH key (ecdsa)...
Generating SSH key (ed25519)...
Generating HTTPS ECDSA server key...
Signing ECDSA certificate...
Starting Apache daemon...
Starting cron...
IPFire v2.29 - www.ipfire.org
=====
ipfire.localdomain running on Linux 6.12.41-ipfire aarch64
ipfire login: root
Password:
No mail.
[root@ipfire ~]#

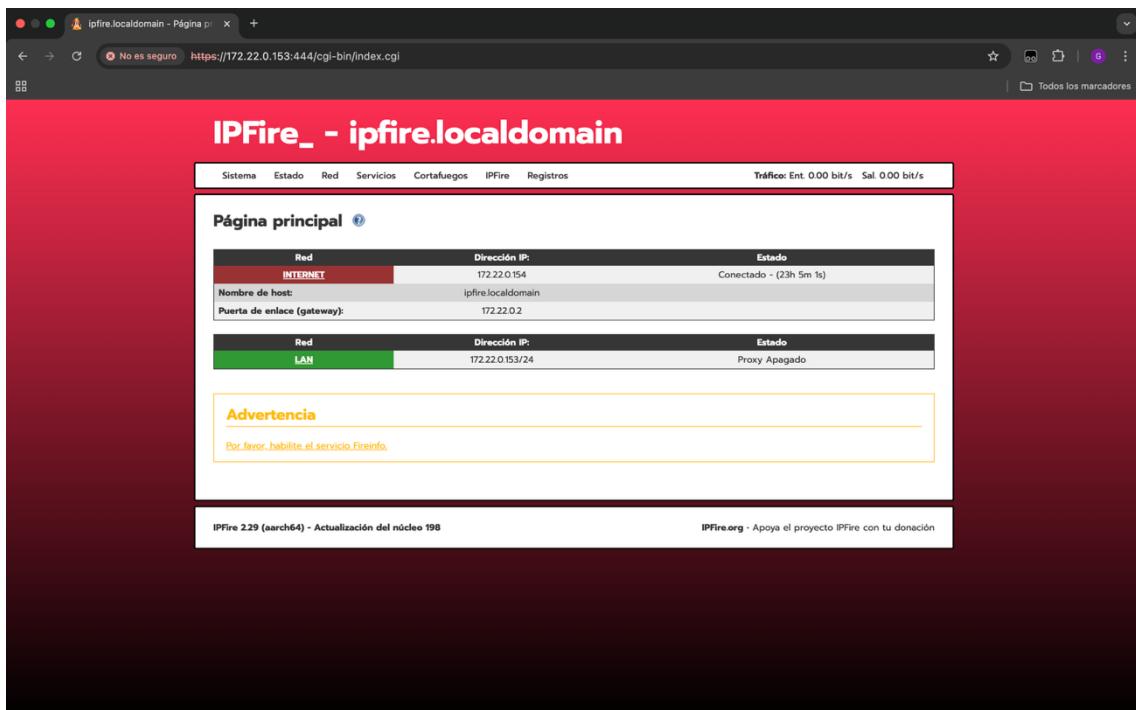
```

Una vez configurado podemos entrar en la maquina poniendo nuestro usuario y contraseña que hemos registrado previamente en la configuración.

Una vez dentro podemos acceder a la consola de administración web accediendo al puerto 444 de la maquina virtual en cuestión con el usuario de administración y la contraseña registrada.



Una vez dentro y después de ignorar el certificado que no esta auto firmado podemos observar y gestionar la maquina virtual desde el navegador



Como ejemplo de dos funcionalidades diferentes la configuración de DHCP y la posibilidad de configurar un proxy que bloquee algunas URL específicas.

The image contains two screenshots of the IPFire web interface, both titled "ipfire.localdomain - Configuración".

Screenshot 1: Configuración DHCP

This screenshot shows the "Configuración DHCP" page. It includes sections for "DHCP", "Actualización DNS", and "Opciones DHCP adicionales".

- DHCP:**
 - Interfaz Green: Activado (checkbox checked)
 - Dirección de inicio: 172.22.0.155
 - Dirección IP: 172.22.0.153
 - Máscara de red: 255.255.255.0
 - Dirección final: 172.22.0.200
 - Tiempo de concesión por defecto (mins): 60
 - Tiempo máximo de concesión (mins): 120
 - Sufijo de nombre de dominio: localdomain
 - DNS Primario: 172.22.0.153
 - Servidor NTP Primario: (empty)
 - Dirección de servidor WINS Primario: (empty)
 - next-server: (empty)
 - filename: (empty)
- Actualización DNS:**
 - Habilitar la actualización de DNS (RFC2136): (checkbox)
 - localdomain: Nombre clave: (empty), Secreto: (empty), Algoritmo: HMAC-MD5
- Opciones DHCP adicionales:**
 - Añadir opción DHCP: (checkbox)
 - Nombre de opción: (empty)
 - Valor de la opción: (empty)
 - Activado: (checkbox)
 - Alcance de la opción: Green (checkbox)

Screenshot 2: Configuración de Proxy Web Avanzado

This screenshot shows the "Configuración de Proxy Web Avanzado" page. It includes sections for "Proxy Web Avanzado", "proxy de subida", "Configuraciones de Log", and "Administración del caché".

- Proxy Web Avanzado:**
 - Configuraciones comunes:
 - Activado en Green: (checkbox checked)
 - Transparente en Green: (checkbox checked)
 - Puerto del proxy: 800
 - Puerto transparente: 3128
 - Nombre de host visible: (empty)
 - Idioma de Mensajes de error: en
 - Diseño de mensajes de error: IPFire
- proxy de subida:**
 - Reenvío de dirección Proxy: (checkbox)
 - Reenvío de dirección IP de cliente: (checkbox)
 - Reenvío de nombre de usuario: (checkbox)
 - No redireccionar configuraciones con autenticación: (checkbox)
 - proxy de subida(host:puerto): (empty)
 - Nombre de usuario de subida: (empty)
 - Contraseña de subida: (empty)
- Configuraciones de Log:**
 - Log activado: (checkbox)
 - Registrar términos de petición: (checkbox)
 - Registrar useragents: (checkbox)
- Administración del caché:**
 - Activar Cachemanager: (checkbox)
 - Cantidad de descriptores de archivos: 16384
 - Tamaño de memoria caché: 128
 - Tamaño Mínimo del objeto(KB): 0
 - Número de subdirectorios Nivel: 16
 - Correo del Administrador del caché: (empty)
 - Contraseña del Administrador del Caché: (empty)
 - Tamaño del caché en disco duro (MB): 0
 - Tamaño máximo del objeto(KB): 4096
 - No meter a cache los siguientes dominios (uno por linea): (empty)

Configuración de Filtro de URL

Configuraciones de Filtro de URL

Categorías bloqueadas.

ads:	<input type="checkbox"/>	aggressive:	<input type="checkbox"/>	audio-video:	<input type="checkbox"/>	drugs:	<input type="checkbox"/>
gambling:	<input type="checkbox"/>	hacking:	<input type="checkbox"/>	mail:	<input type="checkbox"/>	porn:	<input type="checkbox"/>
proxy:	<input type="checkbox"/>	violence:	<input type="checkbox"/>	warez:	<input type="checkbox"/>		

Lista Negra personalizada

Dominios bloqueados (uno por linea)
Ejemplo: www.domain.com

URLs bloqueada (una por linea)
Ejemplo: www.domain.com/ads/

`www.google.es`

Activar Lista Negra personalizada:

Lista Blanca personalizada

Dominios permitidos (uno por linea)
Ejemplo: www.domain.com

URLs permitidos (uno por linea)
Ejemplo: www.domain.com/ads/

Activar Lista Blanca personalizada:

Lista de frases personalizadas

Frases bloqueadas (usar expresiones reguladas)

curl https://www.google.es

[root@ipfire ~]# curl https://www.google.es

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">

<TITLE>301 Moved</TITLE><BODY>

<H1>301 Moved</H1>

The document has moved

Please click [here](https://www.google.es/).

[root@ipfire ~]# curl https://google.es

Podemos observar que antes de meter www.google.es en la lista negra podíamos obtener respuesta dde la web con curl y después ya no. Y por ultimo intentamos instalar paquetes con Pakfire , descargamos tanto samba como el servicio de guardian.

IPFire_ - ipfire.localdomain

Sistema Estado Red Servicios Cortafuegos IPFire Registros Tráfico: Ent: 3915 kbit/s Sal: 10 03 kbit/s

Configuración de Pakfire

Pakfire

pAkfire ha terminado! Volviendo...

[Regresar a Pakfire](#)

```

Dec 27 20:22:32 ipfire pakfire: PANFIRE INFO: IPFire Pakfire 2.29-aarch64 started!
Dec 27 20:22:32 ipfire pakfire: DOWNLOAD STARTED: 2.29-aarch64/lists/server-list.db
Dec 27 20:22:32 ipfire pakfire: DOWNLOAD INFO: Host: pakfire.ipfire.org (HTTPS) - File: 2.29-aarch64/lists/server-list.db
Dec 27 20:22:33 ipfire pakfire: DOWNLOAD INFO: 2.29-aarch64/lists/server-list.db has size of 2285 bytes
Dec 27 20:22:33 ipfire pakfire: DOWNLOAD INFO: HTTP Status Code: 200 - 200 OK
Dec 27 20:22:33 ipfire pakfire: DOWNLOAD INFO: file received. Start checksum signature...
Dec 27 20:22:33 ipfire pakfire: DOWNLOAD INFO: Signature of server-list.db is fine.
Dec 27 20:22:33 ipfire pakfire: DOWNLOAD FINISHED: 2.29-aarch64/lists/server-list.db
Dec 27 20:22:33 ipfire pakfire: DOWNLOAD STARTED: lists/packages_list.db
Dec 27 20:22:33 ipfire pakfire: DOWNLOAD INFO: File: pub/linux/ipfire/pakfire2/2.29-aarch64/lists/packages_list.db
Dec 27 20:22:33 ipfire pakfire: DOWNLOAD INFO: Host: ftp.qwdg.de (HTTPS) - File: pub/linux/ipfire/pakfire2/2.29-aarch64/lists/packages_list.db
Dec 27 20:22:33 ipfire pakfire: DOWNLOAD INFO: pub/linux/ipfire/pakfire2/2.29-aarch64/lists/packages_list.db has size of 4762 by
Dec 27 20:22:34 ipfire pakfire: DOWNLOAD INFO: HTTP Status Code: 200 - 200 OK
Dec 27 20:22:34 ipfire pakfire: DOWNLOAD INFO: file received. Start checksum signature...
Dec 27 20:22:34 ipfire pakfire: DOWNLOAD INFO: Signature of packages_list.db is fine.
Dec 27 20:22:34 ipfire pakfire: DOWNLOAD FINISHED: pub/linux/ipfire/pakfire2/2.29-aarch64/lists/packages_list.db
Dec 27 20:22:34 ipfire pakfire: DOWNLOAD STARTED: lists/core-list.db
Dec 27 20:22:34 ipfire pakfire: DOWNLOAD INFO: File: pub/network/security/ipfire/pakfire2/2.29-aarch64/lists/core-list.db
Dec 27 20:22:36 ipfire pakfire: DOWNLOAD INFO: Host: ftp.yz.yamagata-u.ac.jp (HTTPS) - File: pub/network/security/ipfire/pakfire2/2.29-aarch64/lists/core-list.db
Dec 27 20:22:36 ipfire pakfire: DOWNLOAD INFO: pub/network/security/ipfire/pakfire2/2.29-aarch64/lists/core-list.db has size of
Dec 27 20:22:36 ipfire pakfire: DOWNLOAD INFO: HTTP Status Code: 200 - 200 OK
Dec 27 20:22:36 ipfire pakfire: DOWNLOAD INFO: file received. Start checksum signature...
Dec 27 20:22:38 ipfire pakfire: DOWNLOAD INFO: Signature of core-list.db is fine.
Dec 27 20:22:38 ipfire pakfire: DOWNLOAD FINISHED: pub/network/security/ipfire/pakfire2/2.29-aarch64/lists/core-list.db
Dec 27 20:22:38 ipfire pakfire: PANFIRE INFO: Pakfire has finished. Closing.

```

IPFire 2.29 (aarch64) - Actualización del núcleo 198 IPFire.org - Apoya el proyecto IPFire con tu donación

IPFire_ - ipfire.localdomain

Sistema Estado Red Servicios Cortafuegos IPFire Registros Tráfico: Ent: 0.00 bit/s Sal: 0.00 bit/s

Configuración de Pakfire

Instalar

Paquetes a instalar: **guardian**

Comprobando dependencias...

Dependencias del paquete:

- Paquete: guardian
 - guardian -> perl-Net-IP
 - guardian -> perl-inotify2
 - guardian -> perl-inotify2 -> perl-common-sense

Paquetes a instalar:

- guardian
 - perl-Net-IP
 - perl-common-sense
 - perl-inotify2

[Instalar](#) [Cancelar](#)

IPFire 2.29 (aarch64) - Actualización del núcleo 198 IPFire.org - Apoya el proyecto IPFire con tu donación

ipfire.localdomain - Configuración

No es seguro https://172.22.0.153:444/cgi-bin/pakfire.cgi

Todos los marcadores

IPFire_ - ipfire.localdomain

Sistema Estado Red Servicios Cortafuegos IPFire Registros Tráfico: Ent. 0.00 bit/s Sal. 0.00 bit/s

Configuración de Pakfire

Instalar

Paquetes a instalar: **samba**

Comprobando dependencias...

Dependencias del paquete:

```
Paquete: samba
samba > avahi
samba > avahi > dbus
samba > avahi > libdaemon
samba > libtalloc
samba > perl-Parse-Yapp
samba > wsdd
```

Paquetes a instalar:

```
avahi
dbus
libdaemon
libtalloc
perl-Parse-Yapp
samba
wsdd
```

Instalar Cancelar

IPFire 2.29 (aarch64) - Actualización del núcleo 198 IPFire.org · Apoya el proyecto IPFire con tu donación

ipfire.localdomain - Configuración

No es seguro https://172.22.0.153:444/cgi-bin/pakfire.cgi

Todos los marcadores

IPFire_ - ipfire.localdomain

Sistema Estado Red Servicios Cortafuegos IPFire Registros Tráfico: Ent. 935.76 bit/s Sal. 0.00 bit/s

Configuración de Pakfire

Pakfire

Estado del sistema: Nivel de actualización del núcleo: 198

última actualización realizada 63d 6h 30m 12s
última lista de actualización de servidor hecha 8m 4s
última lista de actualización de núcleo hecha 7m 59s
última lista de actualización de paquetes hecha 8m 3s

Refrescar Lista Actualizar

Complementos disponibles: Complementos instalados:

Seleccione uno o más complementos para instalar.

7zip-17.06-10
alac-0.0.7-1
alsa-1.2.14-23
amazon-ssm-agent-3.3.1345.0-9
apcupsd-3.14.14-2
arpwatch-3.8-1
aws-dl-1.37.4-11
bacula-15.0.2-14
bird-2.15.1-14
borgbackup-1.4.1-19
bwm-ng-0.6.3-3
c-ares-1.34.3-4

Seleccione uno o más complementos para eliminar.

avahi-0.9.14
dbus-1.16.2-13
guardian-2.0.2-27
libdaemon-0.14-1
libtalloc-2.4-3-5
perl-Net-IP-1.26-4
perl-Parse-Yapp-1.21-3
perl-common-sense-3.74-4
perl-inotify2-1.22-4
samba-4.22.4-115
wsdd-0.9-4

Instalar Remover

Configuraciones

Repositorio Estable