P04 – Filtrado de Paquetes

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 ponemos las políticas generales en la tabla filter (la predeterminada) para que haga drop de todo el tráfico entrante y saliente.

```
root@server:~# iptables -t filter -P OUTPUT DROP
target     prot opt source               de
root@server:~# iptables -P INPUT DROP
root@server:~# iptables -L -n
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
root@server:~#
```

Después permitimos el tráfico ICMP entrante y saliente de nuestra red. Podemos ver que nos deja hacer ping de nuestra maquina a otra de nuestra red, y de otra de nuestra red a nuestra máquina.

```
root@server:~# iptables -A OUTPUT -p icmp -d 192.168.217.0/24 -j ACCEPT
root@server:~# iptables -A INPUT -p icmp -s 192.168.217.0/24 -j ACCEPT
root@server:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  192.168.217.0/24     anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             192.168.217.0/24
root@server:~#
```

```
root@server:~# ping 192.168.217.130
PING 192.168.217.130 (192.168.217.130) 56(84) bytes of data.
64 bytes from 192.168.217.130: icmp_seq=1 ttl=64 time=0.205 ms
64 bytes from 192.168.217.130: icmp_seq=2 ttl=64 time=0.834 ms
^C
--- 192.168.217.130 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1035ms
rtt min/avg/max/mdev = 0.205/0.519/0.834/0.314 ms
```

```
C:\Users\kutit>ping 192.168.217.133

Pinging 192.168.217.133 with 32 bytes of data:
Reply from 192.168.217.133: bytes=32 time<1ms TTL=64
Reply from 192.168.217.133: bytes=32 time<1ms TTL=64
Reply from 192.168.217.133: bytes=32 time<1ms TTL=64
Reply from 192.168.217.133: bytes=32 time<1ms TTL=64
```

Ahora permitimos el tráfico saliente para los puertos 80/tcp, 443/tcp y 53/udp (equipo como cliente).

```
root@server:~# iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
root@server:~# iptables -A OUTPUT -p tcp --dport 443 -j ACCEPT
root@server:~# iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
```

Y para permitir el tráfico entrante a los puertos 22/tcp y 80/tcp (equipo como servidor).

```
root@server:~# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
root@server:~# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

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.

Partimos de la config sin filtros original

```
server@server:~$ sudo iptables -F
```

```
root@server:~# iptables -L --line-numbers -n
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
```

Aceptamos la conexion tcp desde una maquina 192.168.217.1 primero, y después bloqueamos todas las conexiones tcp salientes. Y podemos ver que desde la ip especifica si sigo pudiendo conectarme con ssh (que usa tcp), pero no desde otras maquinas en la red)

```
root@server:~# iptables -A OUTPUT -p tcp -d 192.168.217.1 -j ACCEPT
root@server:~# iptables -A OUTPUT -p tcp -j DROP
root@server:~# iptables -L --line-numbers -n
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     6    --  0.0.0.0/0            192.168.217.1
2    DROP       6    --  0.0.0.0/0            0.0.0.0/0
```

```
PS C:\Users\kutit> ssh root@192.168.217.133
root@192.168.217.133's password:
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-52-generic x86_64)
```

maq. 192.168.217.1

```
root@server:~# ssh root@192.168.217.133

^C
```

desde otra maquina

Y ahora cambiando el orden de las reglas, no deja conectarse con la maquina con la IP permitida, aunque esta puesta su regla. Mira primero la regla que tira todas las conexiones tcp y no mira la segunda que permite la conexión desde una IP concreta. EL ORDEN IMPORTA!!

```
root@server:~# iptables -A OUTPUT -p tcp -j DROP
root@server:~# iptables -A OUTPUT -p tcp -d 192.168.217.1 -j ACCEPT
root@server:~# iptables -L --line-numbers -n
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
1    DROP       6    --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     6    --  0.0.0.0/0            192.168.217.1
root@server:~#
```

```
C:\Users\kutit>ssh root@192.168.217.133
ssh: connect to host 192.168.217.133 port 22: Connection timed out
```

desde la maq 192.168.217.1

Las reglas más específicas deben situarse **antes** que las reglas generales, ya que iptables aplica el **primer criterio coincidente** y no evalúa el resto de la cadena.

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.

Queremos que desde la maquina en la que estamos alternando las iptables, para conetarnos a otra maquina con servidor ssh disponible (la maquina 192.168.217.130:22) lo hagamos con una ip "falsa", la ip 1.2.3.4. Hacemos esto alterando la tabla NAT, haciendo un DNAT.


```
root@server:~# ssh root@1.2.3.4
The authenticity of host '1.2.3.4 (1.2.3.4)' can't be established.
ED25519 key fingerprint is SHA256:7x2vGCBBE0bKKcEditZYI9SX2Cvp1HVQeF1YJG5RCCc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '1.2.3.4' (ED25519) to the list of known hosts.
root@1.2.3.4's password:
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-86-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of mié 24 dic 2025 14:43:55 CET

  System load:  0.05              Processes:             252
  Usage of /:   45.2% of 18.53GB  Users logged in:       1
  Memory usage: 33%               IPv4 address for eth0: 192.168.217.130
  Swap usage:   0%
```

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.

Hacemos un primer scan con nmap para sacar el OS de la maquina:

```
┌──(kali㉿kali)-[~]
└─$ nmap -O 192.168.217.133
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-24 08:51 EST
Nmap scan report for 192.168.217.133
Host is up (0.00049s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 00:0C:29:6B:8F:EB (VMware)
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:
linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
```

Ponemos la regla para que obligue a que una conexión se inicie con segmento SYN:

```
root@server:~# iptables -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
root@server:~#
```

Y ya no nos sale información del sistema operativo cuando hacemos un nmap:

```
└─$ nmap -O 192.168.217.133
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-24 09:05 EST
Nmap scan report for 192.168.217.133
Host is up (0.00056s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 00:0C:29:6B:8F:EB (VMware)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.95%E=4%D=12/24%OT=22%CT=1%CU=37747%PV=Y%DS=1%DC=D%G=Y%M=000C29%
OS:TM=694BF34F%P=x86_64-pc-linux-gnu)SEQ(SP=100%GCD=1%ISR=10A%TI=Z%CI=Z%II=
OS:I%TS=A)SEQ(SP=106%GCD=1%ISR=109%TI=Z%CI=Z%TS=A)SEQ(SP=106%GCD=1%ISR=10C%
OS:TI=Z%CI=Z%TS=A)SEQ(SP=106%GCD=1%ISR=10C%TI=Z%CI=Z%II=I%TS=A)SEQ(SP=FF%GC
OS:D=1%ISR=104%TI=Z%CI=Z%II=I%TS=A)OPS(O1=M5B4ST11NW7%O2=M5B4ST11NW7%O3=M5B
OS:4NNT11NW7%O4=M5B4ST11NW7%O5=M5B4ST11NW7%O6=M5B4ST11)WIN(W1=FE88%W2=FE88%
OS:W3=FE88%W4=FE88%W5=FE88%W6=FE88)ECN(R=Y%DF=Y%T=40%W=FAF0%O=M5B4NNSNW7%CC
OS:=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=N)T5(R=Y
OS:%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=N)T7(R=Y%DF=Y%T=40%W=0%S=Z%
OS:A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%
OS:RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)
```

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).

Ponemos las politicas generales en DROP:

```
server@server:~$ sudo iptables -P OUTPUT DROP
server@server:~$ sudo iptables -P FORWARD DROP
server@server:~$ sudo iptables -P INPUT DROP
```

Y ponemos las reglas para permitir solo conexiones salientes para tcp y udp:

```
root@server:~# iptables -A INPUT -p udp -m state --state ESTABLISHED -j ACCEPT
root@server:~# iptables -A OUTPUT -p udp -m state --state NEW,ESTABLISHED -j ACC
EPT
root@server:~# iptables -A INPUT -p tcp -m state --state ESTABLISHED -j ACCEPT
root@server:~# iptables -A OUTPUT -p tcp -m state --state NEW,ESTABLISHED -j ACC
EPT
root@server:~#
```

```
server@server:~$ sudo iptables -A OUTPUT -p tcp -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
server@server:~$ sudo iptables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
server@server:~$ sudo iptables -A OUTPUT -p udp -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
server@server:~$ sudo iptables -A INPUT -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT
server@server:~$ _
```

Y lo mismo para ICMP pero también permitiendo conexiones RELATED (no hace falta NEW EN INPUT):

```
root@server:~# iptables -A OUTPUT -p icmp -m state --state NEW,ESTABLISHED,RELAT
ED -j ACCEPT
```

```
root@server:~# iptables -A INPUT -p icmp -m state --state ESTABLISHED,RELATED -j
 ACCEPT
```

```
root@server:~# iptables -L --line-number
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     udp  --  anywhere             anywhere             state ESTABLI
SHED
2    ACCEPT     tcp  --  anywhere             anywhere             state ESTABLI
SHED
3    ACCEPT     icmp --  anywhere             anywhere             state RELATED
,ESTABLISHED

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     udp  --  anywhere             anywhere             state NEW,EST
ABLISHED
2    ACCEPT     tcp  --  anywhere             anywhere             state NEW,EST
ABLISHED
3    ACCEPT     icmp --  anywhere             anywhere             state NEW,REL
ATED,ESTABLISHED
```

Si intentamos hacer un ping a la maquina desde otra maquina externa o una conexion ssh, no vamos a recibir respuesta:

```
root@server:~# ping 192.168.217.133
PING 192.168.217.133 (192.168.217.133) 56(84) bytes of data.
^C
--- 192.168.217.133 ping statistics ---
68 packets transmitted, 0 received, 100% packet loss, time 68640ms
```

```
root@server:~# ssh root@192.168.217.133
ssh: connect to host 192.168.217.133 port 22: Connection timed out
root@server:~#
```

Pero desde nuestra maquina a otra si hay respuesta:

```
root@server:~# ping 192.168.217.130
PING 192.168.217.130 (192.168.217.130) 56(84) bytes of data.
64 bytes from 192.168.217.130: icmp_seq=1 ttl=64 time=0.277 ms
64 bytes from 192.168.217.130: icmp_seq=2 ttl=64 time=0.284 ms
64 bytes from 192.168.217.130: icmp_seq=3 ttl=64 time=0.566 ms
^C
--- 192.168.217.130 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2083ms
```

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 autentificación en cada sesión (por ejemplo, a dos). ¿Cómo podríamos configurar esta nueva restricción?

Primero creamos una regla para registrar las direcciones IPs de las nuevas conexiones:

```
server@server:~$ sudo iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --set --name LISTASSH --rsource
server@server:~$
```

-m recent: sirve para poder utilizar el módulo recent para contadores de tiempo

--set –name (nombre lista): para crear y nombrar la lista

--rsource: guardar IPs origen dentro de la lista creada

```
server@server:~$ sudo iptables -A INPUT  -p tcp --dport 22 -m state --state NEW -m recent --update --seconds 120 --hitcount 4 --name LISTASSH --rsource -j DROP
server@server:~$
```

Luego, hay que acotar las conexiones para una misma IP a 3 conexiones

```
server@server:~$ sudo iptables -A INPUT -p tcp --dport 22 -mstate --state NEW -j ACCEPT
server@server:~$
```

Luego, si no se cumple la regla anterior (porque una misma dir. IP no ha rebasado el límite permitido, entonces se evaluará esta política para permitir que entre).

```
└─$ ssh servers@192.168.126.131
The authenticity of host '192.168.126.131 (192.168.126.131)' can't be established.
ED25519 key fingerprint is SHA256:L4Tq5/e0GyTzzGW4e3dDONrlk4oaPjKTc8LXv798TVE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.126.131' (ED25519) to the list of known hosts.
servers@192.168.126.131's password:
Permission denied, please try again.
servers@192.168.126.131's password:
Permission denied, please try again.
servers@192.168.126.131's password:
```

Como iptables no tiene capacidad de restringir los límites de autentificación a 2 porque no puede porque trabaja a nivel de red y no puede tener en cuenta los usuarios o sesiones. Por tanto, para acotar la autentificación hay que realizarlo a nivel de aplicación de OpenSSH.

```
"/etc/ssh/sshd_config" 131L, 3516B written
server@server:~$ sudo vi /etc/ssh/sshd_config
```

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

#PubkeyAuthentication yes
```

```
server@server:~$ sudo systemctl restart ssh
server@server:~$ _
```

```
server@server:~$ ssh server@192.168.126.131
server@192.168.126.131's password:
Permission denied, please try again.
server@192.168.126.131's password:
Received disconnect from 192.168.126.131 port 22:2: Too many authentication failures
Disconnected from 192.168.126.131 port 22
server@server:~$
```

CORTAFUEGOS PERSONAL CON UFW

7.

a) Verificar el estado del cortafuegos personal y si ya existen reglas iptables:

```
server@server:~$ sudo ufw status
Status: inactive
server@server:~$ sudo 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
server@server:~$
```

b) Arrancar servicios para pruebas

```
server@server:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Thu 2025-12-25 18:33:16 UTC; 23s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 1889 (apache2)
      Tasks: 55 (limit: 4548)
     Memory: 5.2M (peak: 5.7M)
        CPU: 45ms
     CGroup: /system.slice/apache2.service
             ├─1889 /usr/sbin/apache2 -k start
             ├─1891 /usr/sbin/apache2 -k start
             └─1892 /usr/sbin/apache2 -k start

Dec 25 18:33:16 server systemd[1]: Starting apache2.service - The Apache HTTP Server...
Dec 25 18:33:16 server apachectl[1888]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to
Dec 25 18:33:16 server systemd[1]: Started apache2.service - The Apache HTTP Server.
lines 1-16/16 (END)
```

```
server@server:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/usr/lib/systemd/system/ssh.service; disabled; preset: enabled)
     Active: active (running) since Thu 2025-12-25 18:32:16 UTC; 2min 7s ago
TriggeredBy: ● ssh.socket
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 1197 (sshd)
      Tasks: 1 (limit: 4548)
     Memory: 4.0M (peak: 5.0M)
        CPU: 47ms
     CGroup: /system.slice/ssh.service
             └─1197 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Dec 25 18:32:15 server systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
Dec 25 18:32:16 server sshd[1197]: Server listening on 0.0.0.0 port 22.
Dec 25 18:32:16 server systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
Dec 25 18:32:16 server sshd[1197]: Server listening on :: port 22.
Dec 25 18:32:17 server sshd[1199]: Accepted password for server from 192.168.126.1 port 65129 ssh2
Dec 25 18:32:17 server sshd[1199]: pam_unix(sshd:session): session opened for user server(uid=1000) by server(uid=0)
server@server:~$
```

c) Arrancar UFW y configurar su arranque automático

```
server@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
server@server:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
server@server:~$
```

d) Comprobar el estado de UFW y las reglas generadas

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

```
server@server:~$
server@server:~$ sudo iptables -L -n | more
Chain INPUT (policy DROP)
target     prot opt source               destination
ufw-before-logging-input  0    --  0.0.0.0/0            0.0.0.0/0
ufw-before-input  0    --  0.0.0.0/0            0.0.0.0/0
ufw-after-input  0    --  0.0.0.0/0            0.0.0.0/0
ufw-after-logging-input  0    --  0.0.0.0/0            0.0.0.0/0
ufw-reject-input  0    --  0.0.0.0/0            0.0.0.0/0
ufw-track-input  0    --  0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source               destination
ufw-before-logging-forward  0    --  0.0.0.0/0            0.0.0.0/0
ufw-before-forward  0    --  0.0.0.0/0            0.0.0.0/0
ufw-after-forward  0    --  0.0.0.0/0            0.0.0.0/0
ufw-after-logging-forward  0    --  0.0.0.0/0            0.0.0.0/0
ufw-reject-forward  0    --  0.0.0.0/0            0.0.0.0/0
ufw-track-forward  0    --  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ufw-before-logging-output  0    --  0.0.0.0/0            0.0.0.0/0
ufw-before-output  0    --  0.0.0.0/0            0.0.0.0/0
ufw-after-output  0    --  0.0.0.0/0            0.0.0.0/0
ufw-after-logging-output  0    --  0.0.0.0/0            0.0.0.0/0
ufw-reject-output  0    --  0.0.0.0/0            0.0.0.0/0
ufw-track-output  0    --  0.0.0.0/0            0.0.0.0/0
```

UFW es capaz de traducir automáticamente sus reglas a reglas de iptables.

e) Definir la configuración por defecto del cortafuegos

Restringimos conexiones salientes y entrantes

```
server@server:~$ sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
server@server:~$ sudo ufw default deny outgoing
Default outgoing policy changed to 'deny'
(be sure to update your rules accordingly)
server@server:~$ curl -I google.com
^C
server@server:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), deny (outgoing), disabled (routed)
New profiles: skip
server@server:~$
```

Volvemos a permitir la salida y entrada de tráfico, obviamente se puede jugar con las combinaciones que queramos

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

f) Política por defecto para permitir conexiones salientes e impedir conexiones entrantes

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

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

g) Permitir conexiones entrantes a los servicios arrancados

```
server@server:~$ sudo ufw allow ssh
Rule added
Rule added (v6)
server@server:~$ sudo ufw allow http
Rule added
Rule added (v6)
server@server:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)

server@server:~$
```

h) Limitar conexiones desde una IP origen (mitigar ataques de diccionario)

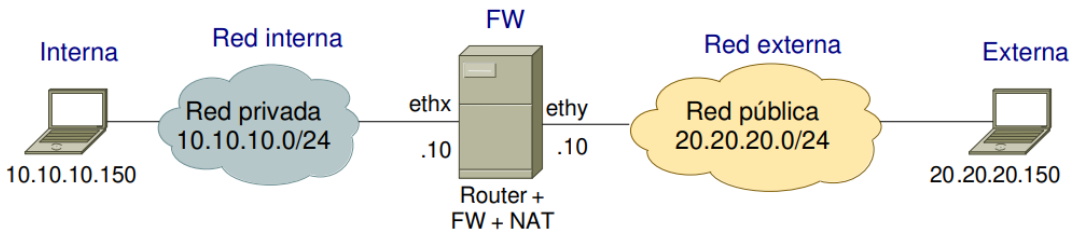
```
server@server:~$ sudo ufw limit ssh
Rule updated
Rule updated (v6)
server@server:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     LIMIT       Anywhere
80/tcp                     ALLOW       Anywhere
22/tcp (v6)                LIMIT       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)

server@server:~$
```

Se ha aplicado un límite de 6 conexiones cada 30 segundos para las conexiones entrantes SSH.

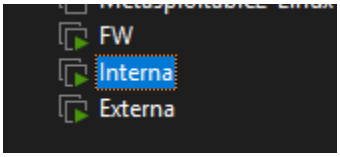
Router con función de cortafuegos y NAT

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 entono 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.



Creamos 3 máquinas virtuales Ubuntu server para realizar la maqueta:



Procedemos con el direccionamiento de cada máquina

FW:

- **vi /etc/netplan/00-installer-config.yaml**
- Guardar cambios con -- > **sudo netplan apply**
- Agregar un network interface en ajustes de la VM para eth1

```
#
#
network:
  version: 2
  ethernets:
   eth0:
    addresses:
        - 20.20.20.10/24
   eth1:
    addresses:
        - 10.10.10.10/24
```

Interna:

```
#
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 10.10.10.150/24
      routes:
      - to: default
        via: 10.10.10.10
```

Externa:

```
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 20.20.20.150/24

      routes:
      - to: default
        via: 20.20.20.10
```

Comprobamos la conectividad desde el FW al resto de máquinas:

```
root@server:~# ping -c 3 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=0.568 ms
64 bytes from 10.10.10.150: icmp_seq=2 ttl=64 time=0.688 ms
64 bytes from 10.10.10.150: icmp_seq=3 ttl=64 time=0.740 ms

--- 10.10.10.150 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.568/0.665/0.740/0.072 ms
root@server:~#
```

```
root@server:~# ping -c 3 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=0.612 ms
64 bytes from 20.20.20.150: icmp_seq=2 ttl=64 time=0.614 ms
64 bytes from 20.20.20.150: icmp_seq=3 ttl=64 time=0.743 ms

--- 20.20.20.150 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.612/0.656/0.743/0.061 ms
root@server:~#
```

También activamos el FORWARD IP en el firewall:

```
root@server:~# sudo vi /etc/sysctl.conf_
```

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

```
root@server:~# sudo sysctl -p
net.ipv4.ip_forward = 1
root@server:~# _
```

Ahora podemos hacer ping desde la máquina interna a la externa y vice versa:

```
root@server:~# ping -c 3 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=1.44 ms
64 bytes from 20.20.20.150: icmp_seq=2 ttl=63 time=1.19 ms
64 bytes from 20.20.20.150: icmp_seq=3 ttl=63 time=1.52 ms

--- 20.20.20.150 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.189/1.383/1.517/0.140 ms
root@server:~#
```

```
root@server:~# ping -c 3 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=1.12 ms
64 bytes from 10.10.10.150: icmp_seq=2 ttl=63 time=1.34 ms
64 bytes from 10.10.10.150: icmp_seq=3 ttl=63 time=1.66 ms

--- 10.10.10.150 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.117/1.370/1.658/0.222 ms
root@server:~# _
```

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

En el FW, Interno y Externo:

```
root@server:~# sudo apt install apache2 -y_
```

```
root@server:~# sudo apt install shorewall -y
```

```
root@server:~# sudo apt install tcpdump -y
```

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.

Lo primero que pide el enunciado lo hemos realizado en el apartado 8) a la hora de crear la maqueta. Luego, ahora levantaremos algún servicio y veremos que se puede acceder:

Externa --> Interna

```
root@server:~# ssh root@20.20.20.150_
```

```
root@server:~# ssh root@20.20.20.150
root@20.20.20.150's password: _
```

```
Last login: Fri Dec 26 14:30:17 2025 from 20.20.20.150
root@server:~# _
```

Interna--> Externa

```
root@server:~# ssh root@20.20.20.150_
```

```
root@server:~# ssh root@20.20.20.150
root@20.20.20.150's password: _
```

```
Last login: Fri Dec 26 14:32:16 2025 from 10.10.10.150
root@server:~#
```

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.

```
root@server:~# ls /etc/shorewall/
conntrack  interfaces  params  policy  rules  shorewall.conf  zones
root@server:~#
```

Archivo **zones**:

```
#ZONE   TYPE    OPTIONS                 IN                      OUT
#                                       OPTIONS                 OPTIONS
fw      firewall
net     ipv4
loc     ipv4
```

Archivo **interfaces**:

```
#############################################################################
#ZONE   INTERFACE       OPTIONS
net     NET_IF          dhcp,tcpflags,nosmurfs,routefilter,logmartians,sourceroute=0,physical=eth0
loc     LOC_IF          tcpflags,nosmurfs,routefilter,logmartians,physical=eth1
~
~
```

En el archivo policy ponemos una regla para que pueda salir la red interna al exterior, otra para que pueda no pueda entrar la red externa ni a la interior ni al FW, otra para el FW a cualquier parte y otra para que la interna pueda al FW.

```
#############################################################################
#SOURCE DEST            POLICY          LOGLEVEL        RATE    CONNLIMIT

loc     net             ACCEPT
net     all             DROP            $LOG_LEVEL
fw      all             ACCEPT
loc     fw              ACCEPT
# THE FOLOWING POLICY MUST BE LAST
all     all             REJECT          $LOG_LEVEL

~
```

Ping de la externa a la interna ya no llega :)

```
root@server:~# ping 10.10.10.150
PING 10.10.10.150 (10.10.10.150) 56(84) bytes of data.
^C
--- 10.10.10.150 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2050ms
```

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.

Usando el archivo de snat en examples de shorewall ya nos da la configuracion necesaria para hacer el enmascaramiento

```
root@server:/etc/shorewall# ls
conntrack  interfaces  params  policy  rules  shorewall.conf  snat  zones
```

```
#############################################################################
#ACTION                  SOURCE                  DEST            PROTO   DPORT
SPORT   IPSEC   MARK    USER    SWITCH  ORIGDEST        PROBABILITY
#
# Rules generated from masq file /home/teastep/shorewall/trunk/Shorewall/Samples
/two-interfaces/masq by Shorewall 5.0.13-RC1 - Sat Oct 15 11:41:40 PDT 2016
#
MASQUERADE              10.10.10.0/24           NET_IF
"/etc/shorewall/snat" 21L, 1115B                                21,1          Todo
```

Todos estos ejemplos están bajo el directorio **/usr/share/doc/shorewall/examples/**

Ahora realizamos la prueba cargándonos el GW para la externa:

```
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 20.20.20.150/24
_
```

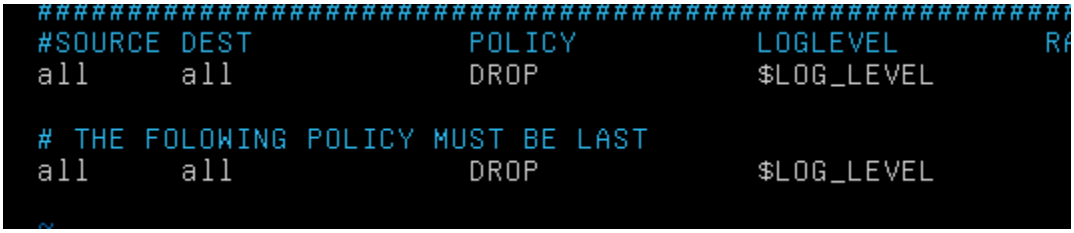
```
root@server:~# ping 20.20.20.150_
```

```
28
17:15:52.616272 ARP, Request who-has _gateway tell server, length 28
17:15:52.616452 ARP, Reply _gateway is-at 00:50:56:f0:72:73 (oui Unknown), lengt
h 46
17:15:53.107838 IP 20.20.20.10 > server: ICMP echo request, id 5195, seq 7, leng
th 64
17:15:53.107857 IP server > 20.20.20.10: ICMP echo reply, id 5195, seq 7, length
 64
17:15:54.109256 IP 20.20.20.10 > server: ICMP echo request, id 5195, seq 8, leng
th 64
17:15:54.109275 IP server > 20.20.20.10: ICMP echo reply, id 5195, seq 8, length
 64
17:15:55.110614 IP 20.20.20.10 > server: ICMP echo request, id 5195, seq 9, leng
th 64
17:15:55.110632 IP server > 20.20.20.10: ICMP echo reply, id 5195, seq 9, length
 64
```

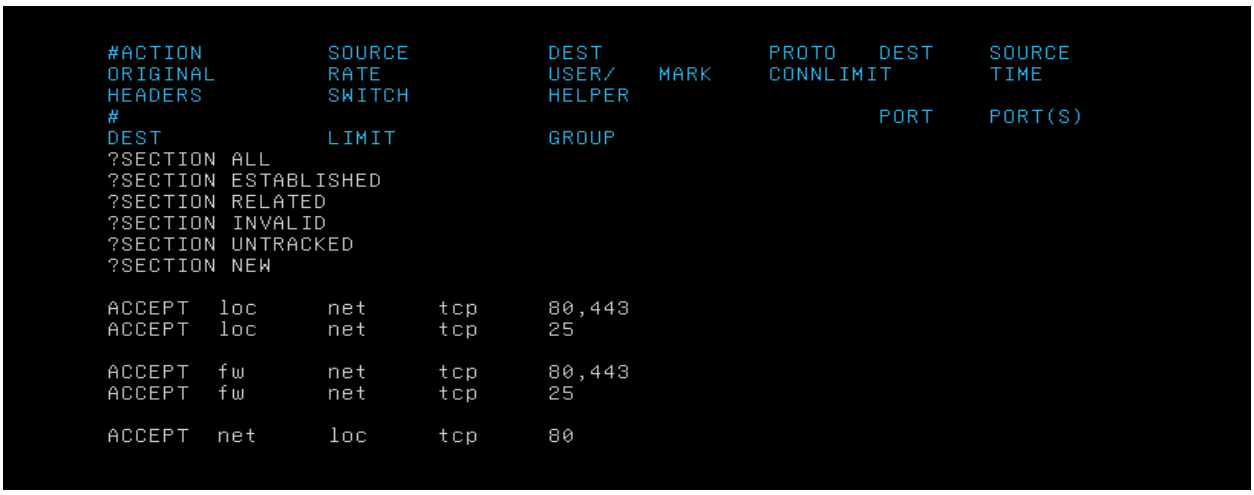
13. Implantar una política restrictiva para el tráfico del FW: todo el tráfico entrante, saliente y que atraviesa el FW será rechazado. A continuación, se habilitará la salida a

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).

Entramos dentro de la política de shorewall de la siguiente manera:
sudo vi /etc/shorewall/policy y metemos la siguiente regla para que el firewall restringa todo el tráfico entrante, saliente o que atraviese el firewall para que lo rechace:



Luego, habilitamos la salida del tráfico HTTP, HTTPS, SMTP como se pide el enunciado, para ello editamos el fichero sudo vi /etc/shorewall/rules y añadimos reglas de salida desde la red interna a la externa y desde el firewall a la red externa:



Luego, nos piden un **port-forwarding** (con un **servicio apache**) para que sea visible desde el exterior y el servicio apache será ofrecida por la máquina de la red interna. Para ello, en primer lugar, editamos el fichero sudo vi /etc/shorewall/dnat (en nuestro caso no existía de manera predeterminada, por tanto, lo creamos y añadimos lo siguiente:

```
#ACTION SOURCE  DEST            PROTO   DEST    PORT    REDIRECT TO

DNAT    net     fw:20.20.20.10  tcp     80              10.10.10.150:80
~
~
~
```

Y luego, en **/etc/shorewall/rules**, añadimos lo que sale en la última línea:

```
#ACTION         SOURCE          DEST            PROTO   DEST    SOURCE
ORIGINAL        RATE            USER/   MARK    CONNLIMIT       TIME
HEADERS         SWITCH          HELPER
#                                                       PORT    PORT(S)
DEST            LIMIT           GROUP
?SECTION ALL
?SECTION ESTABLISHED
?SECTION RELATED
?SECTION INVALID
?SECTION UNTRACKED
?SECTION NEW

ACCEPT  loc     net     tcp     80,443
ACCEPT  loc     net     tcp     25

ACCEPT  fw      net     tcp     80,443
ACCEPT  fw      net     tcp     25

ACCEPT  net     loc     tcp     80
```

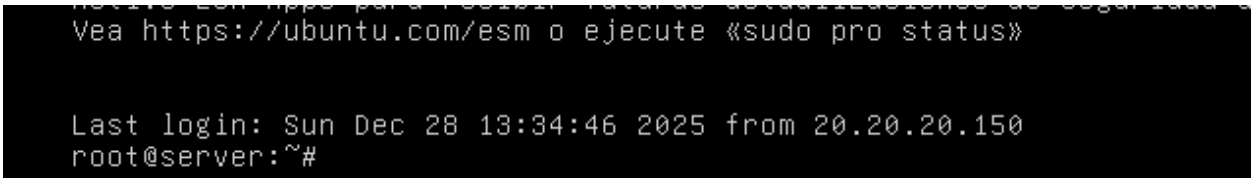
Ahora desde la máquina externa, se ve como es posible mediante el port-forwarding, hacer un curl a la 20.20.20.10 (interfaz del firewall que está en el lado de la red externa) y nos devuelve la página apache hosteada en la máquina interna:

```
        <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 b
ug.
          </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>
    <div class="validator">
    </div>
  </body>
</html>
root@server:/etc/shorewall# _
```

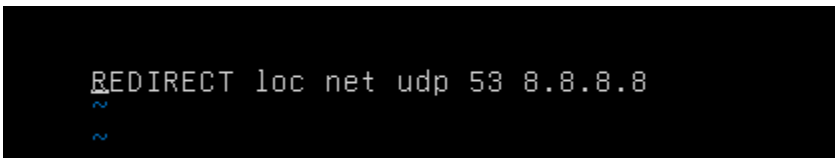
Ahora para permitir la conexión SSH al firewall desde la red externa solamente para la máquina externa 20.20.20.150 metemos la siguiente regla en **/etc/shorewall/rules** que sale a continuación en la captura de abajo**:**

```
#                                                                       PORT    PORT(S)
DEST            LIMIT           GROUP
?SECTION ALL
?SECTION ESTABLISHED
?SECTION RELATED
?SECTION INVALID
?SECTION UNTRACKED
?SECTION NEW

ACCEPT  loc     net     tcp     80,443
ACCEPT  loc     net     tcp     25

ACCEPT  fw      net     tcp     80,443
ACCEPT  fw      net     tcp     25

ACCEPT  net     loc     tcp     80

ACCEPT  net:20.20.20.150        fw      tcp     22
```

Ahora nos conectamos remotamente desde la máquina externa (20.20.20.150) al cortafuegos (su interfaz externa 20.20.20.10) y veremos que se realiza dicha conexión con éxito:

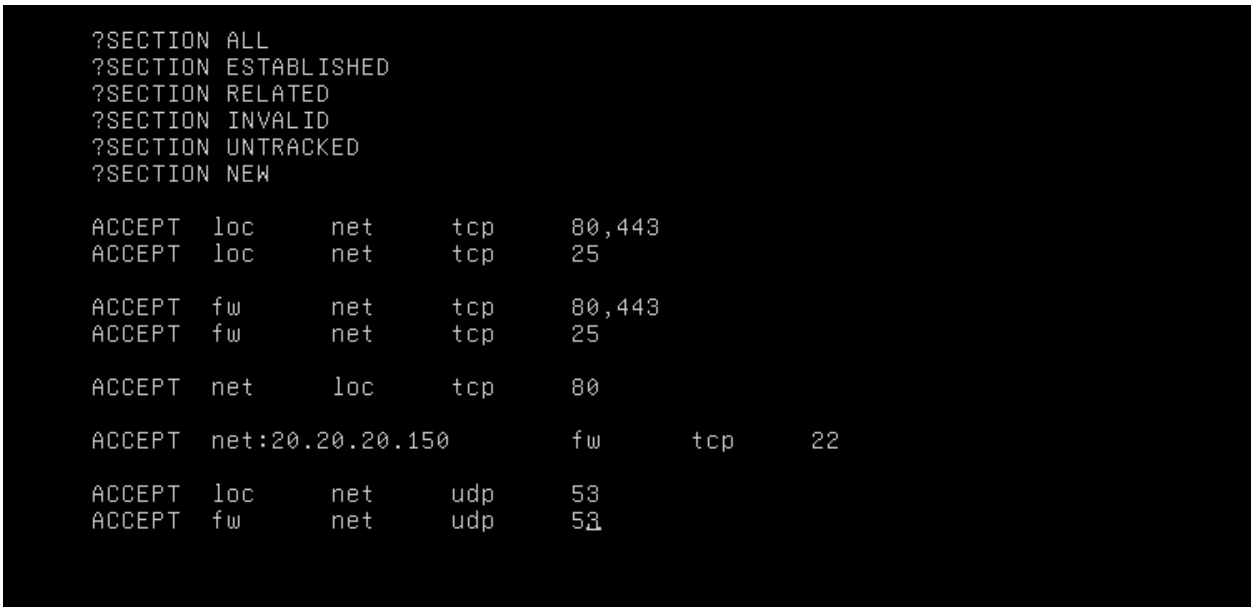


Para redirigir todo el tráfico saliente de DNS a un servidor DNS externo concreto creamos un fichero dentro de /etc/shorewall/redirect. Dentro del fichero /etc/shorewall/redirect metemos el siguiente contenido:



A partir de ahora todo el tráfico saliente de DNS será redirigido al servidor 8.8.8.8:

Añadir reglas para permitir tráfico saliente DNS (interna --> externa):



Comprobamos que se añadan bien los cambios con los siguientes comandos:

- sudo shorewall check
- sudo systemctl restart shorewall