

# Trabajo Práctico

## TP7 - Filtrado de paquetes utilizando Netfilter

Fecha de entrega: 10/11/2021

Franco, Juan Martín 149.615

[juanmartin\\_franco@hotmail.com](mailto:juanmartin_franco@hotmail.com)

### Experiencia de Laboratorio

1. Teniendo en cuenta las actividades que ha realizado en las prácticas de esta asignatura y de la asignatura Teleinformática y Redes. De las cuatro políticas básicas vistas en la teoría (las “4 P”), ¿cuál considera que es la política de firewall (filtrado de paquetes) aplicada en los equipos Linux? ¿cómo llegó a esa conclusión?

Según las experiencias realizadas en los laboratorios de la asignatura y de TyR, puedo deducir que la política de firewall aplicada por defecto en los equipos Linux es la política promiscua (o a lo sumo permisiva).

En todos los laboratorios se permitían paquetes de todos los dispositivos implicados en los mismos y no creo que en los mismos haya una regla de ACCEPT solo para aquellos dispositivos (por lo que puedo descartar las políticas paranoica y prudente).

2. Valide su suposición ejecutando **iptables -L** Busque las líneas Chain INPUT, Chain FORWARD y Chain OUTPUT y tome nota de la política vigente.

Al ejecutar el comando:

```
iptables -L
```

Obtuve el siguiente resultado:

```
juanma@ubuntu:~$ sudo iptables -L
[sudo] password for juanma:
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
LIBVIRT_INP all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
LIBVIRT_FWX all  --  anywhere             anywhere
LIBVIRT_FWI all  --  anywhere             anywhere
LIBVIRT_FWO all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
LIBVIRT_OUT all  --  anywhere             anywhere
```

Por lo que pude confirmar que se trataba de una política promiscua.

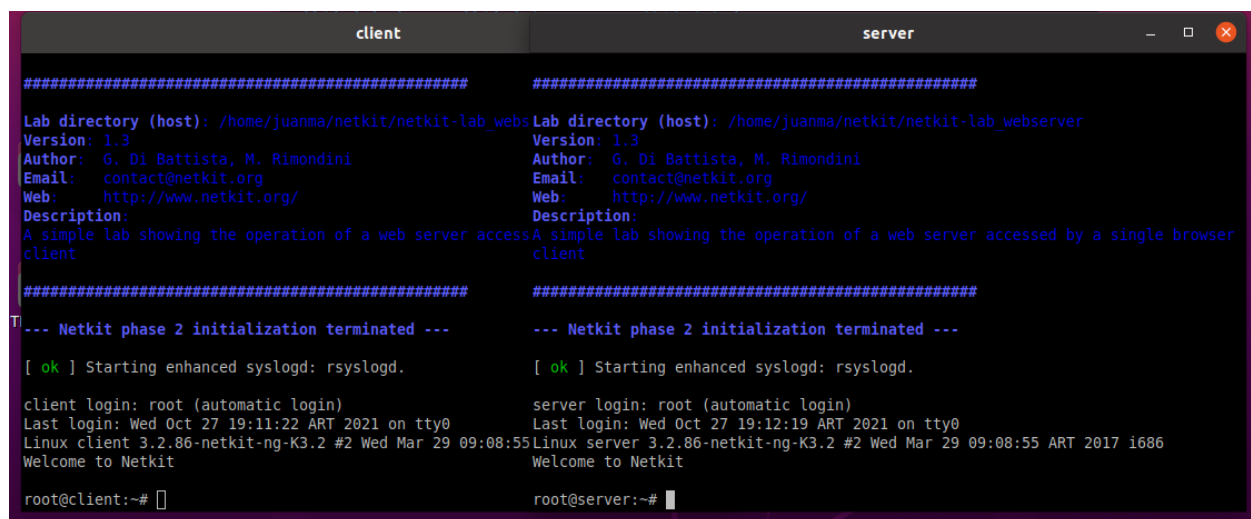
3. Para el resto de esta experiencia de laboratorio y para evitar modificar su configuración personal se utilizará una nueva versión del laboratorio netkit-lab\_webserver.tar.gz, por lo que recomendamos volver a descargarlo y reemplazar el laboratorio existente por el actual. Si no tiene instalado netkit siga las instrucciones provistas en: TP1 de TYR.

Para iniciar el laboratorio **cd ~/netkit/netkit-lab\_webserver/** y luego **lstart**.

Este es un escenario muy sencillo que tiene un host que funciona como webserver llamado “server” con ip **10.0.0.1** y un host “client” con ip **10.0.0.2** que puede realizarle peticiones.

Para iniciar el servidor, una vez posicionado en la carpeta del mismo, utilizo el comando:

`lstart`



```
client                                     server
#####                                     #####
Lab directory (host): /home/juanma/netkit/netkit-lab_webserver
Version: 1.3
Author: G. Di Battista, M. Rimondini
Email: contact@netkit.org
Web: http://www.netkit.org/
Description:
A simple lab showing the operation of a web server accessed by a single browser client
#####                                     #####
--- Netkit phase 2 initialization terminated ---
[ ok ] Starting enhanced syslogd: rsyslogd.
client login: root (automatic login)
Last login: Wed Oct 27 19:11:22 ART 2021 on tty0
Linux client 3.2.86-netkit-ng-K3.2 #2 Wed Mar 29 09:08:55 ART 2017 i686
Welcome to Netkit
root@client:~#

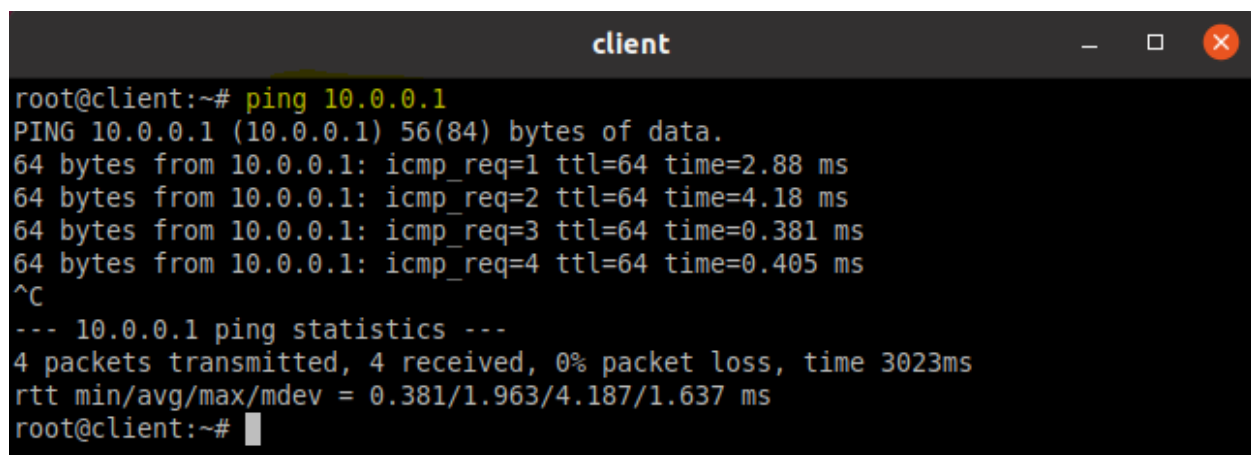
server login: root (automatic login)
Last login: Wed Oct 27 19:12:19 ART 2021 on tty0
Linux server 3.2.86-netkit-ng-K3.2 #2 Wed Mar 29 09:08:55 ART 2017 i686
Welcome to Netkit
root@server:~#
```

4. Haga un ping desde “client” hacia “server” y verifique la conectividad. Luego realice un escaneo al servidor mediante **nmap** . Finalmente, verifique las reglas existentes hasta el momento en el firewall de cada equipo con **iptables -L**.

Para realizar esto, ejecuto el comando:

`ping 10.0.0.1`

desde el cliente.



```
client
root@client:~# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_req=1 ttl=64 time=2.88 ms
64 bytes from 10.0.0.1: icmp_req=2 ttl=64 time=4.18 ms
64 bytes from 10.0.0.1: icmp_req=3 ttl=64 time=0.381 ms
64 bytes from 10.0.0.1: icmp_req=4 ttl=64 time=0.405 ms
^C
--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3023ms
rtt min/avg/max/mdev = 0.381/1.963/4.187/1.637 ms
root@client:~#
```

Como se aprecia en la imagen, hay conectividad entre cliente y servidor.

Ahora, ejecuto el comando `iptables -L` en ambos hosts:

`iptables -L` (cliente)

```
client
root@client:~# iptables -L
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
root@client:~#
```

`iptables -L` (servidor)

```
server
root@server:~# iptables -L
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
root@server:~#
```

Como se ve en las imágenes, ambos tienen una política promiscua.

5. Vamos a proceder ahora a configurar el firewall en el equipo “server”. Una buena práctica, cuando se quiere configurar una política de seguridad prudente, es denegar primero todo aquello que no está expresamente permitido. Para lograrlo se deben definir para cada cadena (INPUT, OUTPUT, FORWARD) una política por defecto. La acción recomendada para ello es “DROP”. Con esto lo que lograremos es descartar cualquier paquete recibido y tampoco podremos enviar ni reenviar nada.

**`iptables -P INPUT DROP`**

**`iptables -P FORWARD DROP`**

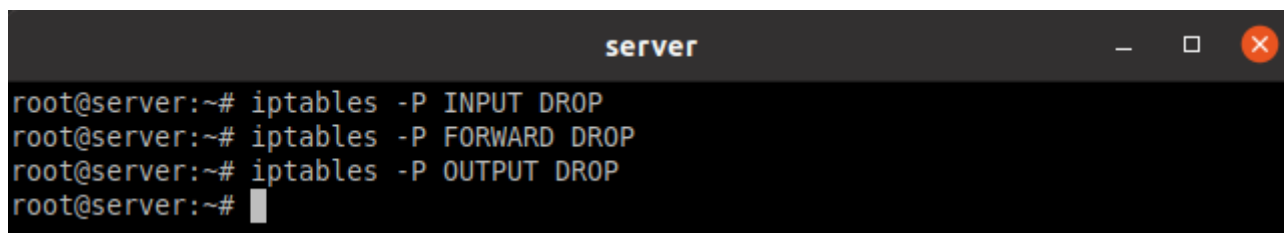
**`iptables -P OUTPUT DROP`**

Ahora, paso a configurar una política de seguridad prudente mediante la ejecución de los 3 comandos nombrados anteriormente:

`iptables -P INPUT DROP`

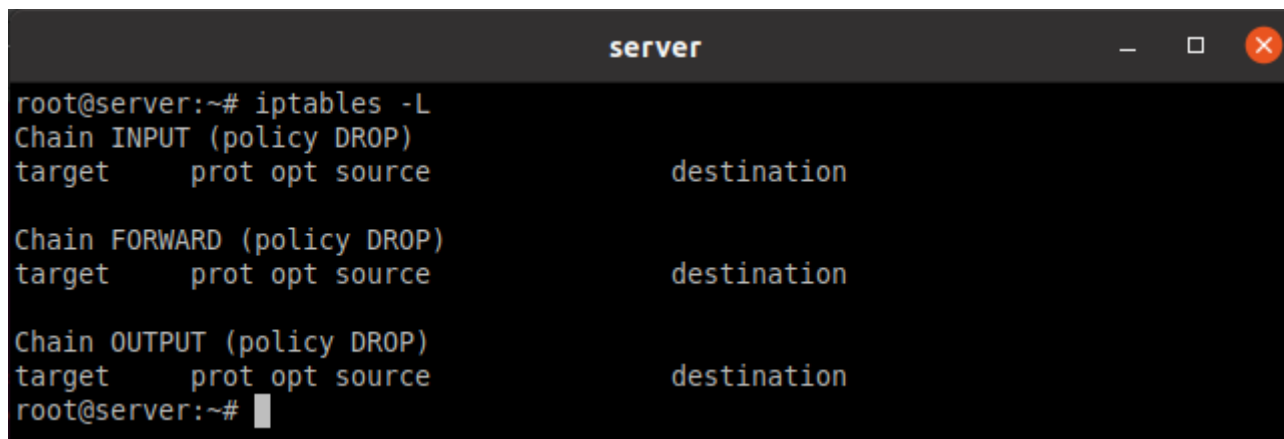
`iptables -P FORWARD DROP`

```
iptables -P OUTPUT DROP
```



```
server
root@server:~# iptables -P INPUT DROP
root@server:~# iptables -P FORWARD DROP
root@server:~# iptables -P OUTPUT DROP
root@server:~#
```

Por último, verifico que se haya configurado correctamente:



```
server
root@server:~# iptables -L
Chain INPUT (policy DROP)
target      prot opt source                destination

Chain FORWARD (policy DROP)
target      prot opt source                destination

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

**¿Ha llegado usted a la configuración más segura para un servidor web? ¿Qué servicio de seguridad no se estaría cumpliendo?**

Se podría decir que en parte es segura ya que no se recibe ningún paquete, aunque tampoco se egresa ni se reenvían, por lo que el servidor no estaría funcionando correctamente. El servicio de seguridad que no se estaría cumpliendo es el de disponibilidad.

6. Si en algún momento quiere deshacer todos los cambios efectuados en las reglas de iptables ejecute **iptables --flush** . De todas formas, las reglas de iptables se borran al reiniciar los equipos (la política por defecto debe ser cambiada manualmente con iptables -P como se vio en el punto 5).

Para deshacer los cambios efectuados, ejecuto el comando:

```
iptables --flush
```



```
server
root@server:~# iptables --flush
root@server:~# iptables -L
Chain INPUT (policy DROP)
target      prot opt source                destination

Chain FORWARD (policy DROP)
target      prot opt source                destination

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

En este caso, no se cambió nada porque las políticas por defecto solo se modifican con los comandos utilizados en el ejercicio 5.

7. Realice una captura desde su terminal en su equipo real. Para ello y mientras está situado en el directorio del laboratorio, escriba el comando **vdump A > captura.pcap**. Verifique la configuración realizando un ping desde “client” hacia “server”. ¿Llegan?. Finalice en su terminal la captura con **CTRL + C** y abra la misma con **wireshark captura.pncap**. ¿Qué puede observar?

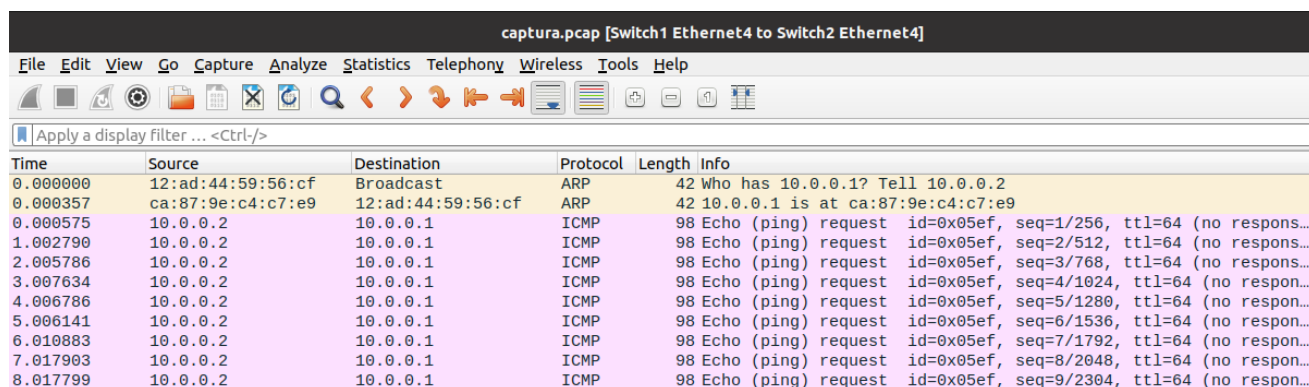
Para realizar una captura, me posiciono sobre la terminal del Host (es decir, de la máquina donde se inició el laboratorio) y ejecuto el comando:

```
vdump A > captura.pncap
```

```
juanma@ubuntu:~/Desktop/TP7$ vdump A > captura.pcap
Running ==> uml_dump A
^C caught signal 2, cleaning up and exiting
```

Por último, abro la captura con el comando:

```
wireshark captura.pncap
```



Time	Source	Destination	Protocol	Length	Info
0.000000	12:ad:44:59:56:cf	Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
0.000357	ca:87:9e:c4:c7:e9	12:ad:44:59:56:cf	ARP	42	10.0.0.1 is at ca:87:9e:c4:c7:e9
0.000575	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x05ef, seq=1/256, ttl=64 (no response received)
1.002790	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x05ef, seq=2/512, ttl=64 (no response received)
2.005786	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x05ef, seq=3/768, ttl=64 (no response received)
3.007634	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x05ef, seq=4/1024, ttl=64 (no response received)
4.006786	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x05ef, seq=5/1280, ttl=64 (no response received)
5.006141	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x05ef, seq=6/1536, ttl=64 (no response received)
6.010883	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x05ef, seq=7/1792, ttl=64 (no response received)
7.017903	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x05ef, seq=8/2048, ttl=64 (no response received)
8.017799	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x05ef, seq=9/2304, ttl=64 (no response received)

Como se aprecia en la imagen, no se obtiene respuesta los mensajes ICMP ya que, como configuramos que las políticas por defecto sean DROP, el paquete nunca llega al servidor (se descartan en la cadena INPUT).

8. El “server” es después de todo un webserver ¿No?. Vamos a permitir que entonces pueda recibir peticiones HTTP de puerto 80 de cualquier IP.

```
iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
```

Con esta línea le decimos al firewall que vamos a agregar una regla (**-A**) para permitir (**-j ACCEPT**) el ingreso (cadena **INPUT**) de los paquetes que lleguen a la interfaz “eth0” (**-i eth0**), con protocolo “tcp” (**-p tcp**) y al puerto destino 80 (**--dport 80**).

Para permitir el ingreso de paquetes por la interfaz eth0 que sean pertenecientes al protocolo TCP y al puerto 80, utilizo el comando:

```
iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
```

```
server
root@server:~# iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
root@server:~#
```

Ahora, verifico que se haya aplicado correctamente:

```
server
root@server:~# iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
root@server:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source                destination            tcp dpt:http
ACCEPT     tcp  --  anywhere              anywhere               tcp dpt:http

Chain FORWARD (policy DROP)
target     prot opt source                destination

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

Efectivamente, cualquier paquete cuyo protocolo de capa 4 sea TCP y esté destinado al puerto 80 pasará por la cadena INPUT.

9. Con el comando “wget” desde el host “client” solicite al “server” la página index. Para ello ejecute **wget http://10.0.0.1/** ¿Funcionó? Realice nuevamente una captura con **vdump A > captura.pcap** y verifique que sucede con wireshark.

Ejecutando el comando:

```
wget http://10.0.0.1/
```

no se obtiene respuesta.

```
client
root@client:~# wget http://10.0.0.1/
--2021-10-29 04:20:50-- http://10.0.0.1/
Connecting to 10.0.0.1:80... ^C
root@client:~#
```

Ahora, para diagnosticar el problema, realizo una captura de Wireshark:

```
juanma@ubuntu: ~/Desktop/TP7
juanma@ubuntu:~/Desktop/TP7$ vdump A > captura2.pcap
Running ==> uml_dump A
```

Luego, ejecuto el mismo comando:

```
client
root@client:~# wget http://10.0.0.1/
--2021-10-29 04:22:52-- http://10.0.0.1/
Connecting to 10.0.0.1:80... ^C
root@client:~#
```

Por último, abro la captura de Wireshark para ver cuál es el problema:

captura2.pcap [Switch1 Ethernet4 to Switch2 Ethernet4]					
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help					
Apply a display filter ... <Ctrl-/>					
Time	Source	Destination	Protocol	Length	Info
0.000000	10.0.0.2	10.0.0.1	TCP	74	32803 → 80 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 T...
0.926724	10.0.0.2	10.0.0.1	TCP	74	[TCP Retransmission] 32803 → 80 [SYN] Seq=0 Win=14600 Len=0 M...
2.922333	10.0.0.2	10.0.0.1	TCP	74	[TCP Retransmission] 32803 → 80 [SYN] Seq=0 Win=14600 Len=0 M...
4.930341	12:ad:44:59:56:cf	ca:87:9e:c4:c7:e9	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
4.930422	ca:87:9e:c4:c7:e9	12:ad:44:59:56:cf	ARP	42	10.0.0.1 is at ca:87:9e:c4:c7:e9

El problema radica en que los paquetes pasan por la cadena INPUT (ya que creamos una regla para eso), pero la respuesta del servidor nunca llega ya que la cadena OUTPUT sigue con su política de DROP (sin ninguna regla creada) por lo que no podrá enviar ningún paquete.

**10. Claramente olvidamos permitir que el servidor pueda enviar paquetes! Para ello agregue la siguiente línea.**

**iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT**

**Vuelva a verificar con wget. ¿La página llegó? ¿Si probamos hacer ping desde el cliente al servidor? ¿Llegan? ¿Deberían hacerlo?**

Ahora, configuro la cadena OUTPUT para que se puedan enviar los paquetes:

```
server
root@server:~# iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT
root@server:~#
```

Verifico que se haya aplicado todo correctamente:

```
server
root@server:~# iptables -L
Chain INPUT (policy DROP)
target    prot opt source                destination            tcp dpt:http
ACCEPT    tcp  --  anywhere              anywhere

Chain FORWARD (policy DROP)
target    prot opt source                destination

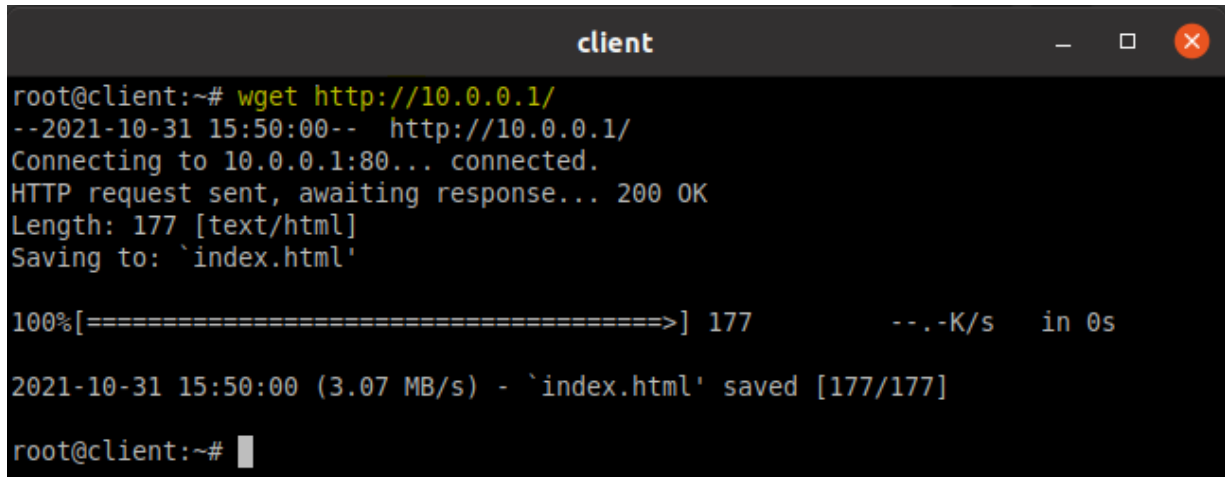
Chain OUTPUT (policy DROP)
target    prot opt source                destination            tcp spt:http
ACCEPT    tcp  --  anywhere              anywhere
root@server:~#
```



Efectivamente, los cambios se aplicaron correctamente.  
Ahora, realizo el mismo procedimiento:

```
wget http://10.0.0.1/
```

Si todo funciona correctamente, debería llegar el wget y debería obtenerse la respuesta del servidor.



```
client
root@client:~# wget http://10.0.0.1/
--2021-10-31 15:50:00-- http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 177 [text/html]
Saving to: `index.html'

100%[=====>] 177          --.-K/s   in 0s

2021-10-31 15:50:00 (3.07 MB/s) - `index.html' saved [177/177]

root@client:~#
```

Como se aprecia en la imagen, el paquete llega correctamente y el servidor envía la respuesta satisfactoriamente.

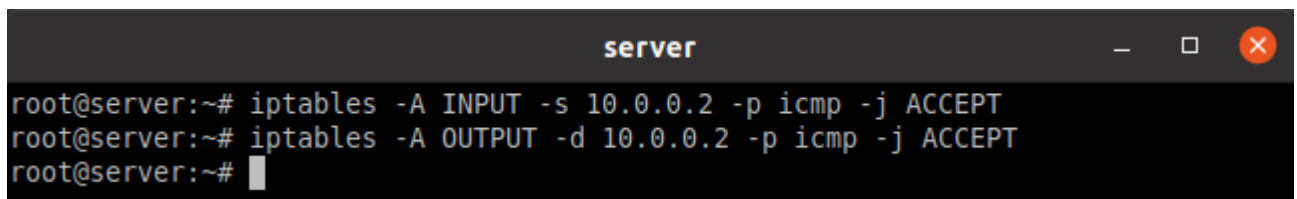
11. Vamos a ser piadosos con el host “client” y vamos a permitir todos los mensajes del protocolo ICMP con este. Para ello en el “server” escribiremos:

```
iptables -A INPUT -s 10.0.0.2 -p icmp -j ACCEPT
iptables -A OUTPUT -d 10.0.0.2 -p icmp -j ACCEPT
```

Con esta línea lo que logramos es agregar una regla en la cadena INPUT para que se acepte todo paquete ICMP de la IP origen del cliente (-s 10.0.0.2) y que también pueda enviarle a esa IP (-d 10.0.0.2) paquetes de este protocolo (-p icmp). Verifique haciendo ping desde el cliente y desde el servidor.

Ahora, procedo a permitir los mensajes del protocolo ICMP entre cliente y servidor, mediante el uso de los siguientes comandos:

```
iptables -A INPUT -s 10.0.0.2 -p icmp -j ACCEPT (para poder recibirlos)
iptables -A OUTPUT -d 10.0.0.2 -p icmp -j ACCEPT (para poder responderlos)
```



```
server
root@server:~# iptables -A INPUT -s 10.0.0.2 -p icmp -j ACCEPT
root@server:~# iptables -A OUTPUT -d 10.0.0.2 -p icmp -j ACCEPT
root@server:~#
```

Ahora, verifico que todo se configuró correctamente mediante el comando:

```
iptables -L
```



```

server
root@server:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source                destination            tcp dpt:http
ACCEPT     tcp  --  anywhere               anywhere               tcp dpt:http
ACCEPT     icmp --  10.0.0.2               anywhere

Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination            tcp spt:http
ACCEPT     tcp  --  anywhere               anywhere               tcp spt:http
ACCEPT     icmp --  anywhere               10.0.0.2
root@server:~#

```

12. Suponga que instalamos un nuevo servicio en el host “server” que escuchará en el puerto TCP 8000, pero no queremos que el host “client” se conecte a él. Así como tenemos configurado hasta ahora el firewall del servidor ¿Podría conectarse el host client al puerto 8000 del server? Vamos a verificar. Realice una captura con **vdump A > obvioQueNoLlega.pcap** . Luego desde el host “client” ejecute **nc 10.0.0.1 8000** para intentar conectarse al server al puerto 8000. Termine la captura. Obsérvela con wireshark.

Primero, realizo una captura con el comando:

`vdump A > obvioQueNoLlega.pcap`

```

juanma@ubuntu: ~/Desktop/TP7
juanma@ubuntu:~/Desktop/TP7$ vdump A > obvioQueNoLlega.pcap
Running ==> uml_dump A

```

Ahora, desde la terminal del cliente, ejecuto el comando:

`nc 10.0.0.1 8000`

```

client
root@client:~# nc 10.0.0.1 8000
^C
root@client:~#

```

Efectivamente, no llega el paquete y no se obtiene respuesta.

Ahora, analizo la captura con Wireshark:

0.000000	f6:ef:6d:d4:49:fe	Broadcast	ARP	42 Who has 10.0.0.1? Tell 10.0.0.2
0.000136	8a:22:5a:0b:f8:f5	f6:ef:6d:d4:49:fe	ARP	42 10.0.0.1 is at 8a:22:5a:0b:f8:f5
0.000555	10.0.0.2	10.0.0.1	TCP	74 60508 → 8000 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=
0.976668	10.0.0.2	10.0.0.1	TCP	74 [TCP Retransmission] 60508 → 8000 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=
2.980625	10.0.0.2	10.0.0.1	TCP	74 [TCP Retransmission] 60508 → 8000 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=
6.985509	10.0.0.2	10.0.0.1	TCP	74 [TCP Retransmission] 60508 → 8000 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=

Efectivamente, los paquetes no llegan, pero tampoco se envía una respuesta al cliente, por lo que TCP intenta establecer nuevamente la conexión una vez expirado el timer de cada mensaje SYN enviado.

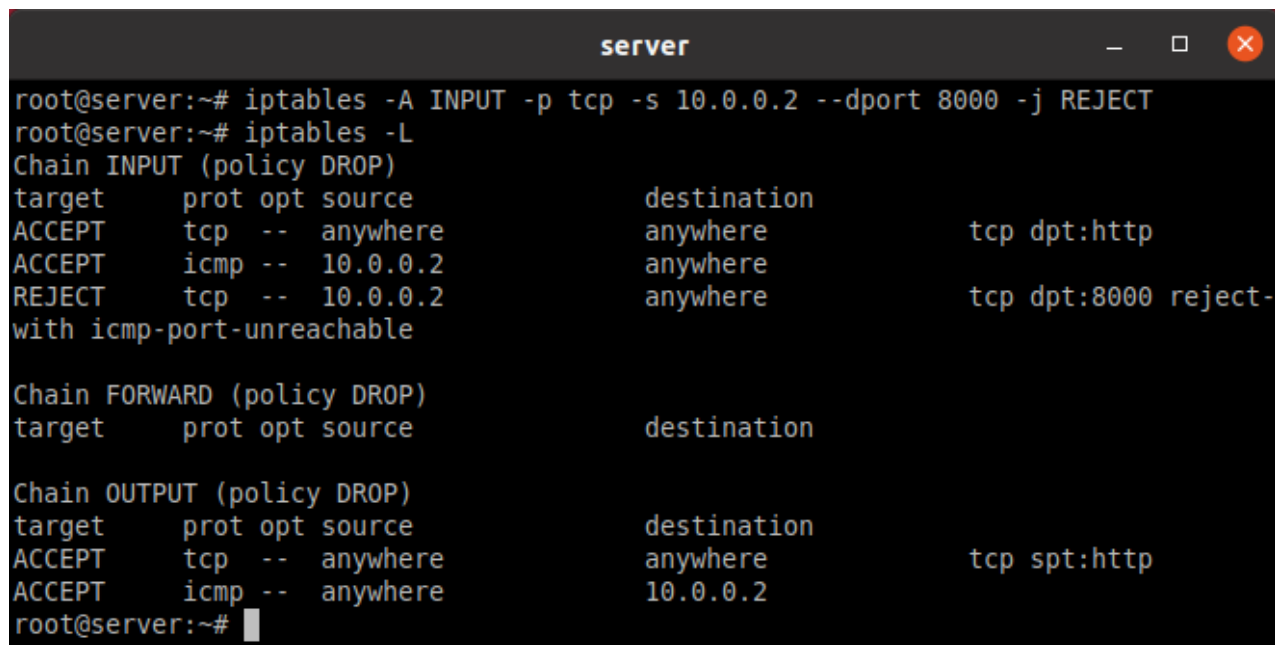
13. Ahora vamos a cambiar las reglas para avisarle al host cliente que rechazamos sus intentos de conexión. Para ello haremos uso de la acción reject.

**iptables -A INPUT -p tcp -s 10.0.0.2 --dport 8000 -j REJECT**

Realice una captura nuevamente con el comando **vdump A > reject.pcap** y realice nuevamente desde el host client en intento de conexión **nc 10.0.0.1 8000** . ¿Qué observa en esta nueva captura a diferencia de la captura anterior?

Ahora, configuro el server para poder notificarle al cliente cuando se rechazan los intentos de conexión. Para ello, utilizo el comando:

```
iptables -A INPUT -p tcp -s 10.0.0.2 --dport 8000 -j REJECT
```



```
server
root@server:~# iptables -A INPUT -p tcp -s 10.0.0.2 --dport 8000 -j REJECT
root@server:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source                destination            tcp dpt:http
ACCEPT     tcp  --  anywhere              anywhere               tcp dpt:http
ACCEPT     icmp --  10.0.0.2              anywhere               tcp dpt:8000 reject-
with icmp-port-unreachable

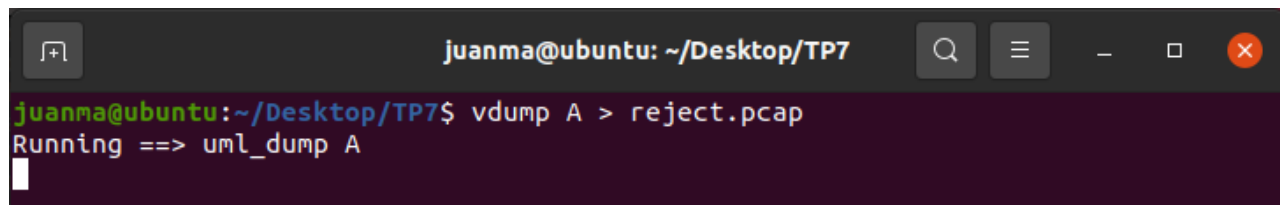
Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination            tcp spt:http
ACCEPT     tcp  --  anywhere              anywhere               tcp spt:http
ACCEPT     icmp --  anywhere              10.0.0.2
root@server:~#
```

Efectivamente, se aplicó correctamente el cambio.

Ahora, inicio una captura utilizando el comando:

```
vdump A > reject.pcap
```



```
juanma@ubuntu: ~/Desktop/TP7
juanma@ubuntu:~/Desktop/TP7$ vdump A > reject.pcap
Running ==> uml_dump A
```

Por último, intento realizar nuevamente la conexión utilizando netcat:

```
client
root@client:~# nc 10.0.0.1 8000
nc: unable to connect to address 10.0.0.1, service 8000
root@client:~#
```

Una vez realizado todo esto, verifico la captura:

wireshark reject.pcap

Time	Source	Destination	Protocol	Length	Info
0.000000	10.0.0.2	10.0.0.1	TCP	74	60509 → 8000 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PER
0.000043	8a:22:5a:0b:f8:f5	Broadcast	ARP	42	Who has 10.0.0.2? Tell 10.0.0.1
0.000382	f6:ef:6d:d4:49:fe	8a:22:5a:0b:f8:f5	ARP	42	10.0.0.2 is at f6:ef:6d:d4:49:fe
0.000418	10.0.0.1	10.0.0.2	ICMP	102	Destination unreachable (Port unreachable)

En este caso, el servidor en lugar de rechazar los paquetes (causando que expire el timer y que TCP intente realizar nuevamente la conexión) y no enviarle ningún mensaje al servidor, se envía un mensaje ICMP, lo que implica que el cliente obtendrá el mensaje:

nc: unable to connect to address 10.0.0.1, service 8000.

14. Nos arrepentimos, ya no queremos avisarle al host “client” que rechazamos sus peticiones. Para ello agregamos la regla:

**iptables -A INPUT -s 10.0.0.2 -p tcp --dport 8000 -j DROP**

Para evitar que se notifique al cliente el rechazo de las peticiones, configuro la regla de la cadena INPUT como **DROP** para todos los mensajes que provengan desde la IP 10.0.0.2 (la ip del cliente) con el protocolo TCP y cuyo puerto de destino sea el 8000.

Para realizar esto, utilizo el comando:

```
iptables -A INPUT -s 10.0.0.2 -p tcp --dport 8000 -j DROP
```

```
server
root@server:~# iptables -A INPUT -s 10.0.0.2 -p tcp --dport 8000 -j DROP
root@server:~#
```

15. Verifique nuevamente haciendo desde el host client el intento de nc 10.0.0.1 8000. ¿Funcionó? ¿Nos dejó de avisar que nos rechaza la conexión? Claro que no, pero ¿por qué sigue avisando el server que rechaza la conexión si claramente ahora lo debería dropear directamente? Verifique las reglas existentes con **iptables -L**. Como puede verse, la regla está efectivamente definida. Sin embargo lo que sucede es que iptables respeta el orden en que se definen las reglas. Es decir, una vez que una regla hace “match” con el paquete entrante se aplica esa regla y se deja de verificar la tabla. En este caso la primera acción es “REJECT”. La solución en este caso será eliminar las últimas dos reglas, ya que la política por defecto ya era dropear. Para ello usaremos -D para borrar ambas reglas, de la siguiente manera:

**iptables -D INPUT -p tcp -s 10.0.0.2 --dport 8000 -j REJECT**  
**iptables -D INPUT -s 10.0.0.2 -p tcp --dport 8000 -j DROP**

Ahora, realizo nuevamente un intento de conexión con el comando:

```
nc 10.0.0.1 8000
```

```
client
root@client:~# nc 10.0.0.1 8000
nc: unable to connect to address 10.0.0.1, service 8000
root@client:~#
```

Como se puede apreciar, el paquete no se dropea, si no que pasa lo que está detallado en el enunciado:

Al respetar el orden de definición de reglas, el paquete llega y “matchea” con la primera que coincida, en este caso la que definimos con target REJECT.

Para eliminar estas políticas definidas, utilizo los siguientes comandos:

```
server
root@server:~# iptables -D INPUT -p tcp -s 10.0.0.2 --dport 8000 -j REJECT
root@server:~# iptables -D INPUT -p tcp -s 10.0.0.2 --dport 8000 -j DROP
root@server:~#
```

## 16. Utilizando el comando **iptables -L** liste las reglas existentes para ver cómo ha quedado todo. Esta configuración ¿es stateful o stateless?

Para visualizar como quedó configurado todo, utilizo el comando:

```
iptables -L
```

```
server
root@server:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source                destination            tcp dpt:http
ACCEPT     tcp  --  anywhere              anywhere
ACCEPT     icmp --  10.0.0.2              anywhere

Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination            tcp spt:http
ACCEPT     tcp  --  anywhere              anywhere
ACCEPT     icmp --  anywhere              10.0.0.2
root@server:~#
```

En este caso, la configuración es Stateless, ya que solo revisa los paquetes y los deja pasar o no basándose en una lista de reglas (las que definimos previamente) y en las características de dicho paquete (ip origen, ip destino, puerto origen, puerto destino, protocolo, etc.).

17. Las reglas definidas para aceptar y responder peticiones http al puerto 80, si bien funcionan, podrían ser aún más “seguras”. Esto es por ejemplo por que la regla **iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT** permitiría que nuestro servidor envíe respuestas a cualquier IP destino sin importar si alguien hizo con nosotros o no una conexión previamente. Nuestro servidor, tal como descubrimos en el punto anterior, tiene un firewall sin estado por lo que ignora las conexiones previas y su relación con el paquete que está examinando. Vamos a cambiar esto y para ello vamos a borrar las reglas que habíamos definido en el firewall del servidor de la siguiente manera:

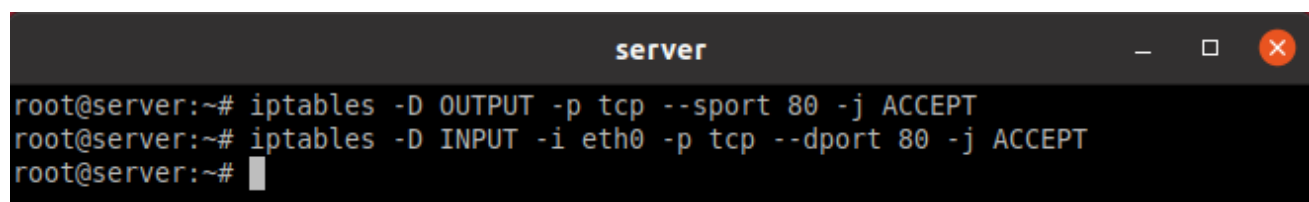
```
iptables -D OUTPUT -p tcp --sport 80 -j ACCEPT
iptables -D INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
```

Verificamos con wget desde el host “client” y comprobamos que nuevamente no podemos hacer peticiones. Ahora agregaremos la reglas:

```
iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
```

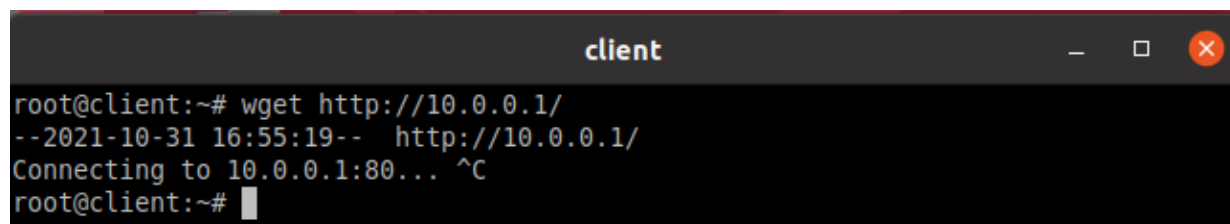
Estas reglas permiten que solo se acepten conexiones nuevas y establecidas y que envíen paquetes solo a aquellas conexiones que ya fueron establecidas. De esta manera ahora el firewall necesita mantener información de las conexiones establecidas por lo que acabamos de convertirlo en “stateful”.

Para cambiar el hecho de que el firewall ignore las conexiones previas (por ser stateless), primero debo borrar las reglas previamente definidas:



```
server
root@server:~# iptables -D OUTPUT -p tcp --sport 80 -j ACCEPT
root@server:~# iptables -D INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
root@server:~#
```

Luego, verifico con el wget si puedo realizar o no peticiones:



```
client
root@client:~# wget http://10.0.0.1/
--2021-10-31 16:55:19-- http://10.0.0.1/
Connecting to 10.0.0.1:80... ^C
root@client:~#
```

Efectivamente, no se pueden realizar peticiones.

Ahora, se procede a transformar el firewall en Stateful utilizando los siguientes comandos:

```
iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
```

```
client
root@client:~# iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW, ESTABLISHED -j ACCEPT
root@client:~# iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
root@client:~#
```

Por último, verifico que se aplicaron todas las reglas con el comando:

```
iptables -L
```

```
client
root@client:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination           tcp dpt: http state
ACCEPT     tcp  --  anywhere              anywhere              tcp dpt: http state
NEW, ESTABLISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination           tcp spt: http state
ACCEPT     tcp  --  anywhere              anywhere              tcp spt: http state
ESTABLISHED
root@client:~#
```

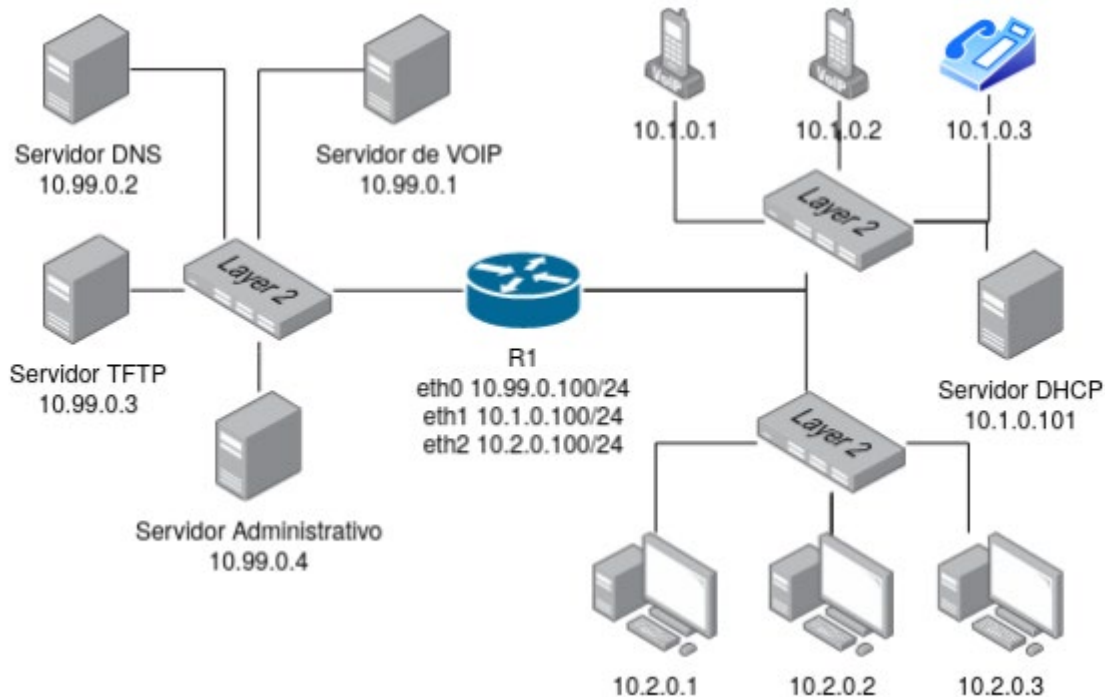
**18. Hasta ahora todas las reglas se han aplicado para filtrar los paquetes que se envían o reciben en un mismo host. ¿Qué cambios habría que efectuar en las reglas definidas para que funcionen en un router que opera como firewall?**

Como bien dice el enunciado, estas reglas solo se aplican para filtrar los paquetes entre un mismo host, pero en el caso de un router, este último no solo recibe peticiones y envía respuestas, sino que también cumple la función de dispositivo intermedio, reenviando los paquetes que le lleguen.

Para que esto funcione, debo modificar la cadena FORWARD, permitiendo la recepción y envío de paquetes a dispositivos definidos.

## Trabajo práctico

### 1. Analice el siguiente escenario:



Se han dado las siguientes directivas acerca de cómo debe funcionar la red:

- a. Los teléfonos deben poder comunicarse con el servidor de VOIP que utiliza SIP como protocolo de señalización. Nos informan además que uno de los teléfonos es de otra marca y que no soporta los mismos CODEC que los demás.

```
iptables -A FORWARD -s 10.1.0.1 -i eth1 -d 10.99.0.1 -o eth0 --dport 5060 -j ACCEPT
iptables -A FORWARD -s 10.1.0.2 -i eth1 -d 10.99.0.1 -o eth0 --dport 5060 -j ACCEPT
iptables -A FORWARD -s 10.1.0.3 -i eth1 -d 10.99.0.1 -o eth0 --dport 5060 -j ACCEPT
```

```
iptables -A FORWARD -s 10.99.0.1 -i eth0 -d 10.1.0.1 -o eth1 --sport 5060 -j ACCEPT
iptables -A FORWARD -s 10.99.0.1 -i eth0 -d 10.1.0.2 -o eth1 --sport 5060 -j ACCEPT
iptables -A FORWARD -s 10.99.0.1 -i eth0 -d 10.1.0.3 -o eth1 --sport 5060 -j ACCEPT
```

El puerto 5060 es el correspondiente al protocolo SIP. En caso de utilizar SIP-TLS, se debe cambiar los valores de dport y sport a 5061.

- b. Los teléfonos VOIP toman su IP del servidor DHCP.

No necesita una regla en el Firewall ya que, si está todo bien configurado, el switch se encarga de reenviar las tramas desde los teléfonos VOIP hacia el servidor DHCP y viceversa.

- c. Los teléfonos VOIP toman su configuración del servidor TFTP.

```
iptables -A FORWARD -s 10.1.0.1 -i eth1 -d 10.99.0.3 -o eth0 -j ACCEPT
iptables -A FORWARD -s 10.1.0.2 -i eth1 -d 10.99.0.3 -o eth0 -j ACCEPT
iptables -A FORWARD -s 10.1.0.3 -i eth1 -d 10.99.0.3 -o eth0 -j ACCEPT
```



```
iptables -A FORWARD -s 10.99.0.3 -i eth0 -d 10.1.0.1 -o eth1 -j ACCEPT
iptables -A FORWARD -s 10.99.0.3 -i eth0 -d 10.1.0.2 -o eth1 -j ACCEPT
iptables -A FORWARD -s 10.99.0.3 -i eth0 -d 10.1.0.3 -o eth1 -j ACCEPT
```

**d. Todos los equipos deben ser capaces de hacer consultas al servidor DNS.**

```
iptables -A FORWARD -d 10.99.0.2 -o eth0 -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -d 10.99.0.2 -o eth0 -p tcp --dport 53 -j ACCEPT
iptables -A FORWARD -s 10.99.0.2 -i eth0 -p udp --sport 53 -j ACCEPT
iptables -A FORWARD -s 10.99.0.2 -i eth0 -p tcp --sport 53 -j ACCEPT
```

**e. Las computadoras deben poder comunicarse con el Servidor Administrativo que funciona en el puerto TCP 9896.**

```
iptables -A FORWARD -s 10.2.0.1 -i eth2 -d 10.99.0.4 -o eth0 -p tcp --dport 9896 -j ACCEPT
iptables -A FORWARD -s 10.2.0.2 -i eth2 -d 10.99.0.4 -o eth0 -p tcp --dport 9896 -j ACCEPT
iptables -A FORWARD -s 10.2.0.3 -i eth2 -d 10.99.0.4 -o eth0 -p tcp --dport 9896 -j ACCEPT
```

```
iptables -A FORWARD -s 10.99.0.4 -i eth0 -d 10.2.0.1 -o eth2 -p tcp --sport 9896 -j ACCEPT
iptables -A FORWARD -s 10.99.0.4 -i eth0 -d 10.2.0.2 -o eth2 -p tcp --sport 9896 -j ACCEPT
iptables -A FORWARD -s 10.99.0.4 -i eth0 -d 10.2.0.3 -o eth2 -p tcp --sport 9896 -j ACCEPT
```

**f. La computadora con ip 10.2.0.3 debe poder comunicarse con el servidor VOIP a la interfaz administrativa web que funciona con HTTPS.**

```
iptables -A FORWARD -s 10.2.0.3 -i eth2 -d 10.99.0.1 -o eth0 --dport 443 -j ACCEPT
iptables -A FORWARD -s 10.99.0.1 -i eth0 -d 10.2.0.3 -o eth2 --sport 443 -j ACCEPT
```

**g. El router tiene una interfaz de configuración ssh a la que debe poder acceder la PC 10.2.0.3. Todos los demás equipos deberían recibir un REJECT si intentan hacerlo.**

```
iptables -A INPUT -s 10.2.0.3 -i eth2 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -d 10.2.0.3 -o eth2 -p tcp --sport 22 -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 22 -j REJECT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT (tiene que poder salir)
```

**h. El router debe poder contestar enviar y recibir mensajes ICMP.**

```
iptables -A INPUT -p icmp -j ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT
```

Su tarea consiste en configurar, a conciencia, el firewall que se encuentra en el router. Para hacerlo debe completar la siguiente tabla de ACL. Considere que el firewall tiene por defecto la acción DROP para todas sus cadenas. Donde el valor de Cadena puede ser INPUT (Entrada), OUTPUT (Salida), FORWARD (Reenvío), entre otras, y el valor de la columna Acción puede ser ACCEPT (Permitir), REJECT (Rechazar) o DROP (Descartar), entre otras.

**[Adjunto Archivo de Excel con la resolución del Ejercicio 1]**

2. Dada la ACL que usted ha confeccionado, cree un script que configure mediante iptables el firewall de ese router. Para ello puede usar de guía el script de ejemplo que se encuentra en el Anexo I así también como pueden consultar el Anexo II que contiene más información acerca del funcionamiento de iptables. Además puede consultar los snippets presentes en la siguiente web <http://www.thegeekstuff.com/2011/06/iptables-rules-examples>.

### Anexo I.

```
#!/bin/sh
#####
# Script para implementación de reglas de filtrado en firewall utilizando netfilter
# Interfaces:
# eth0: 172.210.97.200 - Red Interna de la Organización
# eth1: 172.210.96.200 - Red de Acceso público de la Organización
# eth2: 172.210.0.200 - Red de Interconexión a Proveedor de acceso a Internet
#####

#####
# Políticas por defecto
# Denegar todo el tráfico que no se habilite de manera específica
#####

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

#####
# Eliminar todas las reglas existentes
#####
iptables -F
iptables -X
iptables -Z

#####
# Permitir al firewall enviar/recibir tráfico de servicios básicos
#####
## DNS
iptables -A OUTPUT -d 172.210.96.1 -p udp --dport 53 -j ACCEPT
iptables -A OUTPUT -d 172.210.96.1 -p tcp --dport 53 --syn -j ACCEPT
iptables -A INPUT -s 172.210.96.1 -p udp --sport 53 -j ACCEPT

## NTP
iptables -A OUTPUT -d 172.210.96.1 -p udp --dport 123 -j ACCEPT
iptables -A INPUT -s 172.210.96.1 -p udp --sport 123 -j ACCEPT

## SMTP - Correo electrónico saliente
iptables -A OUTPUT -d 172.210.96.3 -p tcp --dport 25 --syn -j ACCEPT

## Aceptar paquetes de conexiones ya establecidas
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

## Aceptar conexiones locales
iptables -A INPUT -i lo -j ACCEPT
```

```
#####
# Paquetes en tránsito (tráfico que se rutea)
#####

## Permitir paquetes pertenecientes a conexiones ya establecidas
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT

#####
# a) Accedo desde red privada a servicios en servidores corporativos
#####
## Acceso a servidor de correo electrónico
iptables -A FORWARD -i eth0 -s 172.210.97.0/24 -p tcp -d 172.210.96.3 \
    -m multiport --dports 25,110,143 --syn -j ACCEPT

## Acceso a servidor dns y ntp
iptables -A FORWARD -i eth0 -s 172.210.97.0/24 -d 172.210.96.1 -p udp --dport 53 -
j ACCEPT
iptables -A FORWARD -i eth0 -s 172.210.97.0/24 -d 172.210.96.1 -p tcp --dport 53 \
    --syn -j ACCEPT
iptables -A FORWARD -i eth0 -s 172.210.97.0/24 -d 172.210.96.1 -p udp --dport 123 -
j ACCEPT

## Acceso a servidor web
iptables -A FORWARD -i eth0 -s 172.210.97.0/24 -d 172.210.96.2 -p tcp \
    -m multiport --dports 80,443 --syn -j ACCEPT

#####
# b) Desde Internet es posible acceder al servidor web de la empresa.
#####
iptables -A FORWARD -i eth2 -d 172.210.96.2 -p tcp -m multiport --dports 80,443 \
    --syn -j ACCEPT

#####
# c) Desde Internet es posible enviar correo electrónico al servidor de correo
# corporativo.
#####
iptables -A FORWARD -i eth2 -d 172.210.96.3 -p tcp --dport 25 --syn -j ACCEPT

#####
# d) Desde Internet es posible realizar consultas DNS al servidor de nombres
# de dominio de la organización.
#####
iptables -A FORWARD -i eth2 -d 172.210.96.1 -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -i eth2 -d 172.210.96.1 -p tcp --dport 53 --syn -j ACCEPT

#####
# e) El servidor de nombres de dominio debe sincronizarse con otros servidores
# NTP accesibles a través de Internet.

#####
iptables -A FORWARD -i eth1 -s 172.210.96.1 -p udp --dport 123 -j ACCEPT
```

2)

```
#!/bin/sh
```

```
#####
```

```
# Script para implementación de reglas de filtrado en firewall utilizando netfilter
```

```
# Interfaces:
```

```
# eth0: 10.99.0.100 - Red de Servidores (VOIP, DNS, TFTP, Administrativo).
```

```
# eth1: 10.1.0.100 - Red de Teléfonos VOIP y Servidor DHCP.
```

# eth2: 10.2.0.100 - Red de Ordenadores

#####

#####

# Políticas por defecto

# Denegar todo el tráfico que no se habilite de manera específica

#####

iptables -P INPUT DROP

iptables -P OUTPUT DROP

iptables -P FORWARD DROP

#####

#####

# Eliminar todas las reglas existentes

#####

iptables -F

iptables -X

iptables -Z

#####

#####

# Permitir la comunicación entre los teléfonos VOIP y el Servidor VOIP (10.99.0.1)

#####

iptables -A FORWARD -s 10.1.0.1 -i eth1 -d 10.99.0.1 -o eth0 --dport 5060 -j ACCEPT

iptables -A FORWARD -s 10.1.0.2 -i eth1 -d 10.99.0.1 -o eth0 --dport 5060 -j ACCEPT

iptables -A FORWARD -s 10.1.0.3 -i eth1 -d 10.99.0.1 -o eth0 --dport 5060 -j ACCEPT

iptables -A FORWARD -s 10.99.0.1 -i eth0 -d 10.1.0.1 -o eth1 --sport 5060 -j ACCEPT

iptables -A FORWARD -s 10.99.0.1 -i eth0 -d 10.1.0.2 -o eth1 --sport 5060 -j ACCEPT

iptables -A FORWARD -s 10.99.0.1 -i eth0 -d 10.1.0.3 -o eth1 --sport 5060 -j ACCEPT

#####

```
#####
# Permitir la comunicación entre los teléfonos VOIP y el Servidor TFTP (10.99.0.3)
#####
iptables -A FORWARD -s 10.1.0.1 -i eth1 -d 10.99.0.3 -o eth0 -j ACCEPT
iptables -A FORWARD -s 10.1.0.2 -i eth1 -d 10.99.0.3 -o eth0 -j ACCEPT
iptables -A FORWARD -s 10.1.0.3 -i eth1 -d 10.99.0.3 -o eth0 -j ACCEPT
iptables -A FORWARD -s 10.99.0.3 -i eth0 -d 10.1.0.1 -o eth1 -j ACCEPT
iptables -A FORWARD -s 10.99.0.3 -i eth0 -d 10.1.0.2 -o eth1 -j ACCEPT
iptables -A FORWARD -s 10.99.0.3 -i eth0 -d 10.1.0.3 -o eth1 -j ACCEPT
#####

#####
# Permitir las consultas al Servidor DNS (10.99.0.2)
#####
iptables -A FORWARD -d 10.99.0.2 -o eth0 -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -d 10.99.0.2 -o eth0 -p tcp --dport 53 -j ACCEPT
iptables -A FORWARD -s 10.99.0.2 -i eth0 -p udp --sport 53 -j ACCEPT
iptables -A FORWARD -s 10.99.0.2 -i eth0 -p tcp --sport 53 -j ACCEPT
#####

#####
# Permitir la comunicación entre las computadoras y el Servidor Administrativo (10.99.0.4)
#####
iptables -A FORWARD -s 10.2.0.1 -i eth2 -d 10.99.0.4 -o eth0 -p tcp --dport 9896 -j ACCEPT
iptables -A FORWARD -s 10.2.0.2 -i eth2 -d 10.99.0.4 -o eth0 -p tcp --dport 9896 -j ACCEPT
iptables -A FORWARD -s 10.2.0.3 -i eth2 -d 10.99.0.4 -o eth0 -p tcp --dport 9896 -j ACCEPT
iptables -A FORWARD -s 10.99.0.4 -i eth0 -d 10.2.0.1 -o eth2 -p tcp --sport 9896 -j ACCEPT
iptables -A FORWARD -s 10.99.0.4 -i eth0 -d 10.2.0.2 -o eth2 -p tcp --sport 9896 -j ACCEPT
iptables -A FORWARD -s 10.99.0.4 -i eth0 -d 10.2.0.3 -o eth2 -p tcp --sport 9896 -j ACCEPT
#####
```

```
#####
# Permitir que la PC 10.2.0.3 se comuniquen con el servidor VOIP a la interfaz administrativa
# web
#####
iptables -A FORWARD -s 10.2.0.3 -i eth2 -d 10.99.0.1 -o eth0 --dport 443 -j ACCEPT
iptables -A FORWARD -s 10.99.0.1 -i eth0 -d 10.2.0.3 -o eth2 --sport 443 -j ACCEPT
#####

#####
# Permitir que la PC 10.2.0.3 acceda a la interfaz de configuración del router SSH y
# denegar los mensajes del resto de los equipos.
#####
iptables -A INPUT -s 10.2.0.3 -i eth2 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -d 10.2.0.3 -o eth2 -p tcp --sport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -j REJECT
iptables -A OUTPUT -p tcp --sport 22 -j REJECT
#####

#####
# Permitir que el Router reciba y envíe mensajes ICMP
#####
iptables -A INPUT -p icmp -j ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT
#####
```