



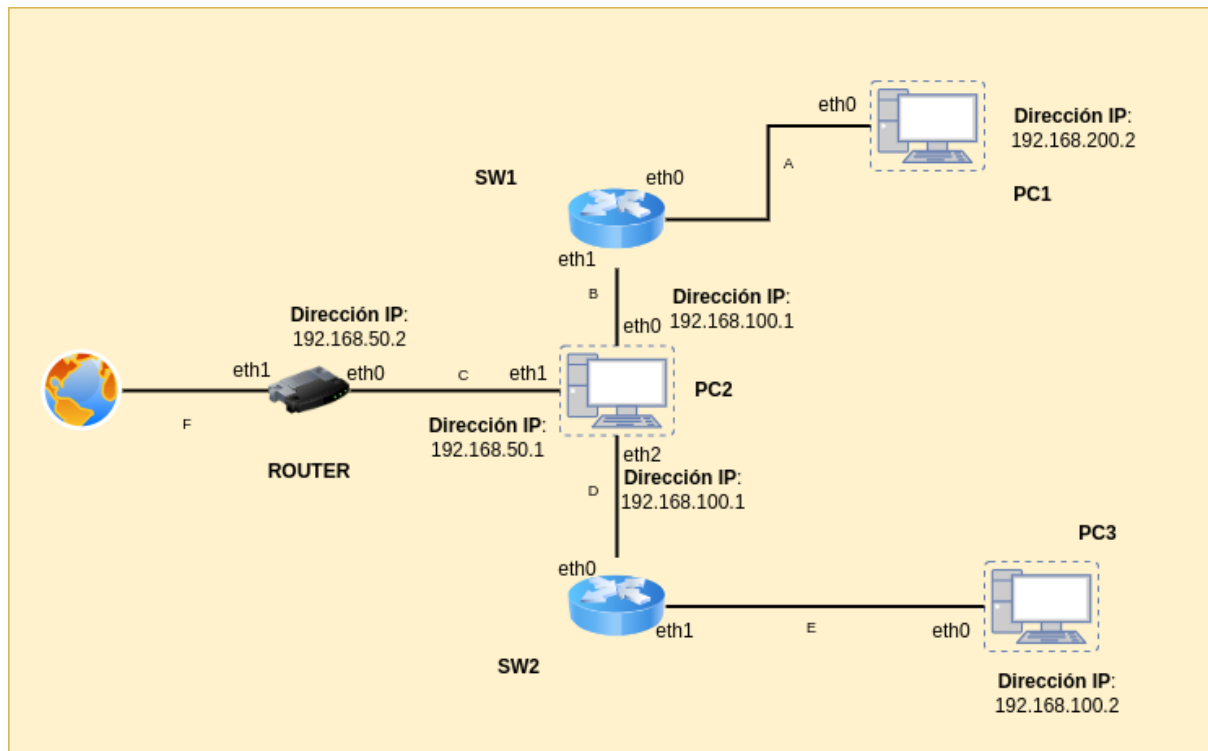
## Seguimiento del Curso iptables



<b>Seguimiento del Curso iptables</b>	<b>1</b>
<b>Escenario</b>	<b>2</b>
<b>Practicamos lo aprendido</b>	<b>4</b>
<b>Montamos cortafuegos personal</b>	<b>7</b>
<b>Montamos cortafuegos perimetral</b>	<b>11</b>
<b>Añadimos reglas nat al cortafuegos perimetral</b>	<b>15</b>
<b>Extensiones de iptables</b>	<b>16</b>
<b>Seguimiento de la conexión</b>	<b>18</b>
<b>Nuevas cadenas</b>	<b>20</b>
<b>Guardamos las iptables</b>	<b>23</b>

## Escenario

El escenario será el siguiente:



Donde posee una conectividad hacia internet y 2 redes virtuales siendo:

- DMZ, con la dirección IP 192.168.200.0/24
- Red local, con la dirección IP 192.168.100.0/24

Mediante este equipo se tendrá que hacer que pueda dar conexión a internet tanto a la red del DMZ como a la de la red local.

## Practicamos lo aprendido

Vamos a aprender a utilizar los parámetros ACCEPT y DROP que serían para aceptar y denegar los paquetes de entrada o salida dependiendo de como lo establezcamos mediante iptables.

```
root@pc2: /
root@pc2:/# iptables -t filter -P FORWARD ACCEPT
root@pc2:/# iptables -t filter -P FORWARD DROP
root@pc2:/#
```

Una forma de ver las reglas que están ya establecidas de forma numérica y con información suficiente sería con la siguiente sentencia:

```
root@pc2: /
root@pc2:/# iptables -L -nv
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source         destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source         destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source         destination
root@pc2:/#
```

Otra forma de ver las reglas establecidas de forma sencilla sería de la siguiente manera:

```
root@pc2: /
root@pc2:/# iptables -L
Chain FORWARD (policy DROP)
target      prot opt source         destination

Chain INPUT (policy ACCEPT)
target      prot opt source         destination

Chain OUTPUT (policy ACCEPT)
target      prot opt source         destination
root@pc2:/#
```

También estaría con el parámetro -S que nos salen las políticas:

```
root@pc2: /
root@pc2:/# iptables -S
-P FORWARD DROP
-P INPUT ACCEPT
-P OUTPUT ACCEPT
root@pc2:/#
```

Añadimos una nueva regla de iptables aceptando todo el tráfico que venga de la red 192.168.200.0 por la interfaz eth0 que sería la que yo le he establecido a dicha máquina al crear el laboratorio y listamos las reglas para ver que se ha añadido con éxito.

```
root@pc2: /
root@pc2:~# iptables -A INPUT -p icmp -s 192.168.200.0/24 -i eth0 -j ACCEPT
root@pc2:~# iptables -L -nv
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination
    0    0 ACCEPT    icmp -- eth0    *        192.168.200.0/24  0.0.0.0/0

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination

root@pc2:~#
```

Comprobamos haciendo ping del pc1 (192.168.200.2) a la máquina pc2 donde he realizado la regla.

```
root@pc1: /
root@pc1:~# ping 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data:
64 bytes from 192.168.200.1: icmp_seq=1 ttl=64 time=0.132 ms
64 bytes from 192.168.200.1: icmp_seq=2 ttl=64 time=0.139 ms
64 bytes from 192.168.200.1: icmp_seq=3 ttl=64 time=0.099 ms
^C
--- 192.168.200.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 54ms
rtt min/avg/max/mdev = 0.099/0.123/0.139/0.019 ms
root@pc1:~#
```

Si comprobamos el listado de reglas nuevamente veremos los paquetes que hemos aceptado.

```
root@pc2: /
root@pc2:~# iptables -L -nv
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination
    3   252 ACCEPT    icmp -- eth0    *        192.168.200.0/24  0.0.0.0/0

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination

root@pc2:~#
```

También estará la política de DROP que denegará todo el tráfico.

```
root@pc2: /
root@pc2:~# iptables -P OUTPUT DROP
root@pc2:~#
```

Si hacemos ping veremos que no contesta el PC2 por la política establecida.

```
root@pc1: /
root@pc1:~# ping 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data:
^C
root@pc1:~#
```

Si listamos las reglas veremos que aunque no acepte le siguen llegando los paquetes a pc2 y se habrán sumado a los paquetes que estaban anteriormente.

```
root@pc2: /
root@pc2:/# iptables -P OUTPUT DROP
root@pc2:/# iptables -L -nv
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
 13 1092 ACCEPT    icmp -- eth0    *        192.168.200.0/24  0.0.0.0/0

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
root@pc2:/#
```

Volvemos a asignar la política de ACCEPT.

```
root@pc2: /
root@pc2:/# iptables -P OUTPUT ACCEPT
root@pc2:/#
```

Y volveremos a poder hacer ping a la máquina PC2.

```
root@pc1: /
root@pc1:/# ping 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data:
64 bytes from 192.168.200.1: icmp_seq=1 ttl=64 time=0.126 ms
64 bytes from 192.168.200.1: icmp_seq=2 ttl=64 time=0.107 ms
^C
--- 192.168.200.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 16ms
rtt min/avg/max/mdev = 0.107/0.116/0.126/0.014 ms
root@pc1:/#
```

Con el parámetro -C podremos chequear que todavía tenemos la regla que hayamos puesto asignada. Por lo que será más fácil comprobar si nos hemos dejado alguna regla de fondo y queramos eliminarla.

```
root@pc2: /
root@pc2:/# iptables -C INPUT -p icmp -s 192.168.200.0/24 -i eth0 -j ACCEPT
root@pc2:/#
```

Al comprobar la existencia de dicha regla luego eliminamos la regla con -D.

```
root@pc2: /
root@pc2:/# iptables -C INPUT -p icmp -s 192.168.200.0/24 -i eth0 -j ACCEPT
root@pc2:/# iptables -D INPUT -p icmp -s 192.168.200.0/24 -i eth0 -j ACCEPT
root@pc2:/#
root@pc2:/# iptables -A INPUT -p icmp -s 192.168.200.0/24 -i eth0 -j ACCEPT
root@pc2:/# iptables -L -nv --line-numbers
Chain FORWARD (policy DROP 0 packets, 0 bytes)
num  pkts bytes target    prot opt in     out     source            destination
tion

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target    prot opt in     out     source            destination
tion
1      0    0 ACCEPT    icmp -- eth0    *        192.168.200.0/24  0.0.0.0/0

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target    prot opt in     out     source            destination
tion
root@pc2:/#
```

Podremos hacer un flush de las reglas mediante el parámetro -F donde limpiará todas las reglas que hayamos establecido.

```
root@pc2: /
root@pc2:~# iptables -F INPUT
root@pc2:~# iptables -L -nv
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target      prot opt in     out     source            destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target      prot opt in     out     source            destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target      prot opt in     out     source            destination
root@pc2:~#
```

También podremos limpiar con el -Z que sería mediante un contador.

```
root@pc2: /
 pkts bytes target      prot opt in     out     source            destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target      prot opt in     out     source            destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target      prot opt in     out     source            destination
root@pc2:~# iptables -Z
root@pc2:~# iptables -L -nv
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target      prot opt in     out     source            destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target      prot opt in     out     source            destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target      prot opt in     out     source            destination
root@pc2:~#
```

## Montamos cortafuegos personal

Vemos la interfaz que posee pc3.

```
root@pc3: /
root@pc3:~# iptables
iptables v1.8.2 (nf_tables): no command specified
Try 'iptables -h' or 'iptables --help' for more information.
root@pc3:~# ip a show dev eth0
45: eth0@if44: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    group default qlen 1000
    link/ether ce:1e:4e:09:b9:04 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.100.2/24 scope global eth0
        valid_lft forever preferred_lft forever
root@pc3:~# ip r
default via 192.168.100.1 dev eth0
192.168.100.0/24 dev eth0 proto kernel scope link src 192.168.100.2
root@pc3:~#
```

Limpiamos todas las reglas que tenga establecida pc3 realizando un flush.

```
root@pc3: /
root@pc3:~# iptables -F
root@pc3:~# iptables -t nat -F
root@pc3:~#
```

Listamos las reglas y veremos que estará todo limpio.

```
root@pc3: /
root@pc3:~# iptables -L -nv
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination
root@pc3:~#
```

Crearemos las reglas que denieguen la entrada y salida de paquetes.

```
root@pc3: /
root@pc3:~# iptables -P OUTPUT DROP
root@pc3:~# iptables -P INPUT DROP
root@pc3:~#
```

Probamos a hacer ping al 1.1.1.1 y veremos que no podremos realizarlo ya que hemos denegado la salida y entrada de paquetes.

```
root@pc3: /
root@pc3:~# iptables -P OUTPUT DROP
root@pc3:~# iptables -P INPUT DROP
root@pc3:~# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
root@pc3:~#
```

Por lo que haremos una regla que permita aceptar los paquetes icmp que procedan de la interfaz eth0.

```
root@pc3: /
root@pc3:~# iptables -A OUTPUT -o eth0 -p icmp -j ACCEPT
root@pc3:~# ping -c 3 192.168.100.1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
^C
--- 192.168.100.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 55ms
root@pc3:~#
```

Veremos el listado de reglas que está establecida la regla que acabamos de crear.

```
root@pc3: /
root@pc3:~# iptables -L -nv
Chain INPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination
    3    252 ACCEPT      icmp -- *          eth0    0.0.0.0/0    0.0.0.0/0
root@pc3:~#
```



Si probamos a hacer ping a pc2 podremos ver que se conectan.

```
root@pc3: /
root@pc3:~# iptables -A INPUT -i eth0 -p icmp -j ACCEPT
root@pc3:~# ping -c 3 192.168.100.1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1: icmp_seq=1 ttl=64 time=0.173 ms
64 bytes from 192.168.100.1: icmp_seq=2 ttl=64 time=0.139 ms
64 bytes from 192.168.100.1: icmp_seq=3 ttl=64 time=0.142 ms

--- 192.168.100.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 49ms
rtt min/avg/max/mdev = 0.139/0.151/0.173/0.018 ms
root@pc3:~#
```

Veremos al listar las reglas los paquetes que han sido enviados.

```
root@pc3: /
root@pc3:~# iptables -L -nv
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    3   252 ACCEPT     icmp -- eth0    *           0.0.0.0/0           0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    6   504 ACCEPT     icmp -- *       eth0    0.0.0.0/0           0.0.0.0/0
root@pc3:~#
```

Probaremos a hacer ping a la dirección localhost y nos seguirá poniendo que no está permitido.

```
root@pc3: /
root@pc3:~# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
root@pc3:~#
```

Realizamos las reglas que acepten la entrada y salida por la interfaz lo (siendo esta la de localhost).

```
root@pc3: /
root@pc3:~# iptables -A INPUT -i lo -j ACCEPT
root@pc3:~# iptables -A OUTPUT -o lo -j ACCEPT
root@pc3:~#
```

Si listamos veremos la nueva regla establecida.

```
root@pc3: /
root@pc3:~# iptables -L -nv
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    3   252 ACCEPT     icmp -- eth0    *           0.0.0.0/0           0.0.0.0/0

    0     0 ACCEPT     all  -- lo      *           0.0.0.0/0           0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    6   504 ACCEPT     icmp -- *       eth0    0.0.0.0/0           0.0.0.0/0

    0     0 ACCEPT     all  -- *       lo      0.0.0.0/0           0.0.0.0/0
root@pc3:~#
```

Veremos que ya puede conectarse.

```
root@pc3: /
root@pc3:~# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.038 ms
```

Hacemos reglas que acepten la entrada y salida por la dirección 192.168.1.1 (siendo esta el nameserver para poder hacer dig a [www.openwebinars.net](http://www.openwebinars.net)) por el puerto 53.

```
root@pc3: /
root@pc3:~# iptables -A OUTPUT -o eth0 -d 192.168.1.1/32 -p udp --dport 53 -j ACCEPT
root@pc3:~# iptables -A INPUT -i eth0 -s 192.168.1.1/32 -p udp --sport 53 -j ACCEPT
root@pc3:~#
```

Veremos que podremos hacer dig a la página sin problema alguno:

```
root@pc3: /
root@pc3:~# dig 192.168.1.1 www.openwebinars.net

;<> DiG 9.11.5-P4-5.1+deb10u8-Debian <> 192.168.1.1 www.openwebinars.net
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NXDOMAIN, id: 12891
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;192.168.1.1. IN A

;; AUTHORITY SECTION:
* 86095 IN SOA a.root-servers.net. nstld.verisign-grs.com. 2023021501 1800 900 604800 86400

;; Query time: 26 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Wed Feb 15 22:41:20 UTC 2023
;; MSG SIZE rcvd: 115

;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 4966
```

Realizamos reglas que acepten entrada y salida por la interfaz eth0 por el puerto 80.

```
root@pc3: /
root@pc3:~# iptables -A OUTPUT -o eth0 -p tcp --dport 80 -j ACCEPT
root@pc3:~# iptables -A INPUT -i eth0 -p tcp --sport 80 -j ACCEPT
Bad argument '80'
Try 'iptables -h' or 'iptables --help' for more information.
root@pc3:~# iptables -A INPUT -i eth0 -p tcp --sport 80 -j ACCEPT
root@pc3:~#
```

Creamos más reglas que permiten la entrada y salida por el puerto 443 por la misma interfaz.

```
root@pc3: /
root@pc3:~# iptables -A OUTPUT -o eth0 -p tcp --dport 80 -j ACCEPT
root@pc3:~# iptables -A INPUT -i eth0 -p tcp --sport 80 -j ACCEPT
Bad argument '80'
Try 'iptables -h' or 'iptables --help' for more information.
root@pc3:~# iptables -A INPUT -i eth0 -p tcp --sport 80 -j ACCEPT
root@pc3:~# iptables -A OUTPUT -o eth0 -p tcp --dport 443 -j ACCEPT
root@pc3:~# iptables -A INPUT -i eth0 -p tcp --sport 443 -j ACCEPT
root@pc3:~#
```

# Montamos cortafuegos perimetral

Comprobamos la ip de pc3 y vemos que pueda comunicarse con pc1.

```
root@pc3: /
root@pc3:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
80: eth0@if79: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 42:cf:a5:e4:6c:99 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.100.2/24 scope global eth0
        valid_lft forever preferred_lft forever
root@pc3:~# ip r
default via 192.168.100.1 dev eth0
192.168.100.0/24 dev eth0 proto kernel scope link src 192.168.100.2
root@pc3:~# ping 192.168.100.1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1: icmp_seq=1 ttl=64 time=0.245 ms
^C
--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.245/0.245/0.245/0.000 ms
root@pc3:~#
```

Comprobamos la ip que tiene pc1 y vemos que puede comunicarse con pc3.

```
root@pc1: /
root@pc1:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
78: eth0@if77: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 56:cb:62:0b:b1:c1 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.200.2/24 scope global eth0
        valid_lft forever preferred_lft forever
root@pc1:~# ip r
default via 192.168.200.1 dev eth0
192.168.200.0/24 dev eth0 proto kernel scope link src 192.168.200.2
root@pc1:~# ping 192.168.100.2
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
64 bytes from 192.168.100.2: icmp_seq=1 ttl=63 time=0.225 ms
64 bytes from 192.168.100.2: icmp_seq=2 ttl=63 time=0.159 ms
^C
--- 192.168.100.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 4ms
rtt min/avg/max/mdev = 0.159/0.192/0.225/0.033 ms
root@pc1:~#
```

Añadimos el bit forward en la máquina firewall (siendo esta pc2).

```
root@pc2:~# echo 1 > /proc/sys/net/ipv4/ip_forward
bash: /proc/sys/net/ipv4/ip_forward: Read-only file system
root@pc2:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Probamos a hacer ping al pc3 y veremos que si conecta:

```
root@pc1: /
root@pc1:~# ping 192.168.100.2
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
64 bytes from 192.168.100.2: icmp_seq=1 ttl=63 time=0.199 ms
64 bytes from 192.168.100.2: icmp_seq=2 ttl=63 time=0.187 ms
^C
--- 192.168.100.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 13ms
rtt min/avg/max/mdev = 0.187/0.193/0.199/0.006 ms
root@pc1:~#
```

Listamos las reglas.

```
root@pc2: /
root@pc2:~# iptables -L -nv
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source                   destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source                   destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source                   destination
root@pc2:~#
```

Denegamos la entrada, salida y sobrepase los paquetes.

```
root@pc2: /
root@pc2:~# iptables -P INPUT DROP
root@pc2:~# iptables -P OUTPUT DROP
root@pc2:~# iptables -P FORWARD DROP
root@pc2:~#
```

Creamos una regla que acepte entrada y salida de paquetes icmp por la interfaz del bridge.

```
root@pc2: /
root@pc2:~# iptables -A OUTPUT -o br0 -p icmp -j ACCEPT
root@pc2:~# iptables -A INPUT -i br0 -p icmp -j ACCEPT
root@pc2:~#
```

Permitimos la entrada y salida de paquetes icmp por las redes 192.168.100.0 y 192.168.200.0.

```
root@pc2: /
root@pc2:~# iptables -A INPUT -i eth2 -p icmp -s 102.168.100.0/24 -j ACCEPT
root@pc2:~# iptables -A OUTPUT -o eth2 -p icmp -d 102.168.100.0/24 -j ACCEPT
root@pc2:~# iptables -A INPUT -i eth0 -p icmp -s 102.168.200.0/24 -j ACCEPT
root@pc2:~# iptables -A OUTPUT -o eth0 -p icmp -d 102.168.200.0/24 -j ACCEPT
root@pc2:~#
```

Probamos a hacer ping de pc1 a pc3:

```
root@pc1: /
root@pc1:~# ping 192.168.100.2
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
 64 bytes from 192.168.100.2: icmp_seq=1 ttl=63 time=0.197 ms
 64 bytes from 192.168.100.2: icmp_seq=2 ttl=63 time=0.192 ms
^C
--- 192.168.100.2 ping statistics ---
 2 packets transmitted, 2 received, 0% packet loss, time 11ms
 rtt min/avg/max/mdev = 0.192/0.194/0.197/0.014 ms
root@pc1:~#
```

Probamos hacer ping a la máquina de pc1.

```
root@pc3: /
root@pc3:~# ping 192.168.200.2
PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data.
 64 bytes from 192.168.200.2: icmp_seq=1 ttl=63 time=0.200 ms
 64 bytes from 192.168.200.2: icmp_seq=2 ttl=63 time=0.231 ms
 64 bytes from 192.168.200.2: icmp_seq=3 ttl=63 time=0.209 ms
^C
--- 192.168.200.2 ping statistics ---
 3 packets transmitted, 3 received, 0% packet loss, time 45ms
 rtt min/avg/max/mdev = 0.200/0.213/0.231/0.017 ms
root@pc3:~#
```

Hacemos las reglas de tal forma que permita el tráfico por la red 192.168.100.0 y 192.168.200.0 por el puerto 53.

```
root@pc2: /
root@pc2:/# iptables -A FORWARD -i eth2 -o eth1 -s 192.168.100.0/24 -p udp --dpo
rt 53 -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth2 -i eth1 -d 192.168.100.0/24 -p udp --spo
rt 53 -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth0 -o eth1 -s 192.168.200.0/24 -p udp --dpo
rt 53 -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth0 -i eth1 -d 192.168.200.0/24 -p udp --spo
rt 53 -j ACCEPT
root@pc2:/#
```

Si hacemos un dig a la página de openwebinars veremos que nos lo hace sin problemas.

```
root@pc1: /
root@pc1:/# dig 192.168.1.1 www.openwebinars.net

;<> DiG 9.11.5-P4-5.1+deb10u8-Debian <> 192.168.1.1 www.openwebinars.net
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NXDOMAIN, id: 32069
;; flags: qr rd ra ad: QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags: udp: 512
;; QUESTION SECTION:
;192.168.1.1. IN A

;; AUTHORITY SECTION:
* 86395 IN SOA a.root-servers.net. nstld.verisi
gn-grs.com. 2023021501 1800 900 604800 86400

;; Query time: 22 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Wed Feb 15 22:32:27 UTC 2023
;; MSG SIZE rcvd: 115

;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NXDOMAIN, id: 58255
```

Listamos las reglas y lo tendremos de la siguiente forma:

```
root@pc2: /
root@pc2:/# iptables -L
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT icmp -- anywhere anywhere
ACCEPT icmp -- 102.168.100.0/24 anywhere
ACCEPT icmp -- 102.168.200.0/24 anywhere

Chain FORWARD (policy DROP)
target prot opt source destination
ACCEPT icmp -- anywhere anywhere
ACCEPT icmp -- anywhere anywhere
ACCEPT icmp -- anywhere anywhere
ACCEPT icmp -- anywhere anywhere
ACCEPT udp -- 192.168.100.0/24 anywhere udp dpt:domain
ACCEPT udp -- anywhere 192.168.100.0/24 udp spt:domain
ACCEPT udp -- 192.168.200.0/24 anywhere udp dpt:domain
ACCEPT udp -- anywhere 192.168.200.0/24 udp spt:domain

Chain OUTPUT (policy DROP)
target prot opt source destination
ACCEPT icmp -- anywhere anywhere
ACCEPT icmp -- anywhere 102.168.100.0/24
ACCEPT icmp -- anywhere 102.168.200.0/24
root@pc2:/#
```

Añadimos el servidor de nombre de openwebinar tal como usa alberto molina:

```
root@pc2: /
GNU nano 3.2 /etc/resolv.conf
nameserver 8.8.8.8
nameserver 192.168.1.1
```

Realizamos reglas permitiendo en la red 192.168.100.0 la transferencia de paquetes en los puertos 80 y 443.

```
root@pc2: /
root@pc2:/# iptables -A FORWARD -i eth2 -o eth1 -s 192.168.100.0/24 -p tcp --dport 80 -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth2 -i eth1 -d 192.168.100.0/24 -p tcp --sport 80 -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth2 -i eth1 -d 192.168.100.0/24 -p tcp --sport 443 -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth2 -o eth1 -s 192.168.100.0/24 -p tcp --dport 443 -j ACCEPT
root@pc2:/#
```

Así serían las reglas.

```
root@pc2: /

Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source               destination
  7   588 ACCEPT      icmp -- eth2    eth0    0.0.0.0/0            0.0.0.0/0
  7   588 ACCEPT      icmp -- eth0    eth2    0.0.0.0/0            0.0.0.0/0
   0     0 ACCEPT      icmp -- eth0    eth2    0.0.0.0/0            0.0.0.0/0
   0     0 ACCEPT      icmp -- eth2    eth0    0.0.0.0/0            0.0.0.0/0
   0     0 ACCEPT      udp -- eth2    eth1    192.168.100.0/24     0.0.0.0/0
   0     0 ACCEPT      udp dpt:53 -- eth1    eth2    0.0.0.0/0            192.168.100.
0/24
   3   267 ACCEPT      udp -- eth0    eth1    192.168.200.0/24     0.0.0.0/0
   0     0 ACCEPT      udp dpt:53 -- eth1    eth0    0.0.0.0/0            192.168.200.
0/24
   0     0 ACCEPT      tcp -- eth2    eth1    192.168.100.0/24     0.0.0.0/0
   0     0 ACCEPT      tcp dpt:80 -- eth1    eth2    0.0.0.0/0            192.168.100.
0/24
   0     0 ACCEPT      tcp spt:80 -- eth1    eth2    0.0.0.0/0            192.168.100.
0/24
   0     0 ACCEPT      tcp spt:443 -- eth2    eth1    192.168.100.0/24     0.0.0.0/0
   0     0 ACCEPT      tcp dpt:443 -- eth2    eth1    192.168.100.0/24     0.0.0.0/0

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source               destination
   0     0 ACCEPT      icmp -- *      br0     0.0.0.0/0            0.0.0.0/0
   0     0 ACCEPT      icmp -- *      eth2    0.0.0.0/0            102.168.100.
0/24
   0     0 ACCEPT      icmp -- *      eth0    0.0.0.0/0            102.168.200.
0/24
root@pc2:/#
```

Realizaremos un forward para la red 192.168.200.0 aceptando los paquetes del puerto origen y destino de 443 y 80.

```
root@pc2: /
root@pc2:/# iptables -A FORWARD -o eth0 -d 192.168.200.2/32 -p tcp --dport 443 -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth0 -s 192.168.200.2/32 -p tcp --sport 443 -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth0 -d 192.168.200.2/32 -p tcp --dport 80 -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth0 -s 192.168.200.2/32 -p tcp --sport 80 -j ACCEPT
root@pc2:/#
```

## Añadimos reglas nat al cortafuegos perimetral

Permitiremos el tráfico desde la máquina firewall a la máquina pc, por lo que activaremos el bit de forward.

```
root@pc2: /
root@pc2:~# echo 1 > /proc/sys/net/ipv4/ip_forward
bash: /proc/sys/net/ipv4/ip_forward: Read-only file system
root@pc2:~#
```

También tendremos que descomentar en el fichero `/etc/sysctl.conf` la siguiente línea:

```
root@pc2: /
GNU nano 3.2 /etc/sysctl.conf Modified
# 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
```

Haremos un nat en la red de la máquina pc1.

```
root@pc2: /
root@pc2:~# iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o eth1 -j MASQUERADE
root@pc2:~#
```

Permitiremos el tráfico entre los puertos 80, 443 y 25.

```
root@pc2: /
root@pc2:~# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j DNAT --to 192.168.200.2
root@pc2:~# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 443 -j DNAT --to 192.168.200.2
root@pc2:~# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 25 -j DNAT --to 192.168.200.2
root@pc2:~#
```

# Extensiones de iptables

Mediante: man iptables-extensions

```
root@pc2: /
iptables-extensions(8)      iptables 1.8.2      iptables-extensions(8)

NAME
    iptables-extensions - list of extensions in the standard iptables distribution

SYNOPSIS
    iptables [-m name [module-options...]] [-j target-name [target-options...]]

    iptables [-m name [module-options...]] [-j target-name [target-options...]]

MATCH EXTENSIONS
    iptables can use extended packet matching modules with the -m or --match options, followed by the matching module name; after these, various extra command line options become available, depending on the specific module. You can specify multiple extended match modules in one line, and you can use the -h or --help options after the module has been specified to receive help specific to that module. The extended match modules are evaluated in the order they are specified in the rule.

    If the -p or --protocol was specified and if and only if an unknown option is encountered, iptables will try load a match module of the same name as the protocol, to try making the option available.

add-type
    This module matches packets based on their address type. Address types are used within the kernel networking stack and categorize addresses into various groups. The exact definition of that group depends on the specific layer three protocol.

    The following address types are possible:

    UNSPEC an unspecified address (i.e. 0.0.0.0)

    UNICAST an unicast address
```

Añadimos extensiones al cortafuegos perimetral. Si listamos veremos los paquetes que obedecen a la regla menos restrictiva.

```
root@pc2: /

root@pc2: /# iptables -A OUTPUT -o eth1 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@pc2: /# iptables -L OUTPUT -n
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination          icmp_type 8
ACCEPT    icmp --  0.0.0.0/0            0.0.0.0/0            icmp_type 8
ACCEPT    icmp --  0.0.0.0/0            0.0.0.0/0            icmp_type 8
root@pc2: /# iptables -L OUTPUT -nv
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out    source               destination          icmp_type 8
0      0 ACCEPT    icmp -- *     eth1   0.0.0.0/0          0.0.0.0/0            icmp_type 8
0      0 ACCEPT    icmp -- *     eth1   0.0.0.0/0          0.0.0.0/0            icmp_type 8
root@pc2: /#

root@pc2: /# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=54 time=16.1 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=54 time=18.7 ms
^C
--- 1.1.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 2ms
rtt min/avg/max/mdev = 16.089/17.387/18.685/1.298 ms
root@pc2: /#
```



Hacemos lo mismo con INPUT y borramos las reglas que teníamos de icmp a cualquiera. Realizamos unas reglas más restrictivas.

The image shows two terminal windows. The left window, titled 'root@pc2: /', displays the following commands and output:

```
root@pc2:~# iptables -D OUTPUT -o eth2 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@pc2:~# iptables -I OUTPUT -o eth2 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@pc2:~# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data:
64 bytes from 1.1.1.1: icmp_seq=1 ttl=54 time=16.2 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=54 time=16.0 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=54 time=16.3 ms
^C
--- 1.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 5ms
rtt min/avg/max/mdev = 15.997/16.170/16.237/0.163 ms
root@pc2:~# iptables -L OUTPUT -nv
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source       destination
0      0 ACCEPT    icmp -- *      eth2    0.0.0.0/0    0.0.0.0/0
root@pc2:~#
```

The right window, titled 'root@pc1: /', displays the following commands and output:

```
--- Startup Commands Log
++ ip link set dev eth0
++ ip addr add 192.168.200.2/24 dev eth0
++ ip link set dev eth0 up
++ route add default gw 192.168.200.1
--- End Startup Commands Log

root@pc1:~# ping 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data:
64 bytes from 192.168.200.1: icmp_seq=1 ttl=64 time=0.202 ms
64 bytes from 192.168.200.1: icmp_seq=2 ttl=64 time=0.109 ms
^C
--- 192.168.200.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 31ms
rtt min/avg/max/mdev = 0.109/0.155/0.202/0.048 ms
root@pc1:~#
```

Restringimos entre la DMZ y nuestra red lan local:

The terminal window, titled 'root@pc2: /', shows the following commands:

```
root@pc2:~# iptables -A FORWARD -i eth0 -o eth2 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@pc2:~# iptables -A FORWARD -o eth0 -i eth2 -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@pc2:~# iptables -A FORWARD -i eth2 -o eth0 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@pc2:~# iptables -A FORWARD -o eth2 -i eth0 -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@pc2:~#
```

Limitamos las conexiones simultáneas:

The terminal window, titled 'root@pc2: /', shows the following commands:

```
root@pc2:~# iptables -A FORWARD -i eth2 -o eth1 -p tcp --syn --dport 25 \
> -m connlimit --connlimit-above 2 -j REJECT --reject-with tcp-reset
root@pc2:~# iptables -A FORWARD -i eth2 -o eth1 -p tcp --syn --dport 80 \-m connlimit --connlimit-above 2 -j REJECT --reject-with tcp-reset
root@pc2:~# iptables -D FORWARD -i eth2 -o eth1 -p tcp --syn --dport 80 \-m connlimit --connlimit-above 2 -j REJECT --reject-with tcp-reset
root@pc2:~# iptables -A FORWARD -i eth2 -o eth1 -p tcp --syn --dport 80 \-m connlimit --connlimit-above 15 -j REJECT --reject-with tcp-reset
root@pc2:~# iptables -A FORWARD -i eth2 -o eth1 -p tcp --syn --dport 443 \-m connlimit --connlimit-above 15 -j REJECT --reject-with tcp-reset
root@pc2:~#
```

Usamos el módulo time para poder limitar por tiempos las conexiones:

```
root@pc2: /
root@pc2:/# iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 53 -m multiport \
> --sports 1024:65535 -s 192.168.200.0/24 -m time \
> --timestart 12:00 --timestop 12:30 -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth1 -i eth2 -p udp --sport 53 -m multiport -
-sports 1024:65535 -s 192.168.200.0/24 -m time --timestart 12:00 --timestop 12:3
0 -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 80 -m multiport -
-sports 1024:65535 -s 192.168.200.0/24 -m time --timestart 12:00 --timestop 12:3
0 -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth1 -i eth2 -p udp --sport 80 -m multiport -
-sports 1024:65535 -s 192.168.200.0/24 -m time --timestart 12:00 --timestop 12:3
0 -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 443 -m multiport
--sports 1024:65535 -s 192.168.200.0/24 -m time --timestart 12:00 --timestop 12:
30 -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth1 -i eth2 -p udp --sport 443 -m multiport
--sports 1024:65535 -s 192.168.200.0/24 -m time --timestart 12:00 --timestop 12:
30 -j ACCEPT
root@pc2:/#
```

Establecemos un rango de tiempo que la máquina podrá navegar.

## Seguimiento de la conexión

Limpiamos lo realizado en el apartado anterior para realizar el nuevo ejercicio:

```
root@pc2: /
root@pc2:/# iptables -F
root@pc2:/# iptables -t nat -F
root@pc2:/# iptables -Z
root@pc2:/# iptables -t nat -Z
root@pc2:/# iptables -P INPUT DROP
root@pc2:/# iptables -P OUTPUT DROP
root@pc2:/# iptables -P FORWARD DROP
root@pc2:/#
```

Realizamos las reglas de icmp.

```
root@pc2: /
root@pc2:/# iptables -A OUTPUT -o eth2 -p icmp -m icmp --icmp-type echo-request
-j ACCEPT
root@pc2:/# iptables -A INPUT -i eth2 -p icmp -m icmp --icmp-type echo-reply -j
ACCEPT
root@pc2:/# iptables -A INPUT -i eth2 -p icmp -s 192.168.100.0/24 -j ACCEPT
root@pc2:/# iptables -A OUTPUT -o eth2 -p icmp -d 192.168.100.0/24 -j ACCEPT
root@pc2:/# iptables -A INPUT -i eth0 -p icmp -s 192.168.200.0/24 -j ACCEPT
root@pc2:/# iptables -A OUTPUT -o eth0 -p icmp -d 192.168.200.0/24 -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth2 -o eth0 -p icmp -m icmp --icmp-type echo
-request -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth2 -i eth0 -p icmp -m icmp --icmp-type echo
-reply -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth0 -o eth2 -p icmp -m icmp --icmp-type echo
-request -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth0 -i eth2 -p icmp -m icmp --icmp-type echo
-reply -j ACCEPT
root@pc2:/#
```

Realizamos las reglas de estado:

```
root@pc2: /
root@pc2:/# iptables -A FORWARD -i eth2 -o eth0 -s 192.168.100.0/24 \
> -p udp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth2 -i eth0 -s 192.168.100.0/24 \ -p udp --d
port 53 -m state --state NEW,ESTABLISHED -j ACCEPT
Bad argument '-p'
Try 'iptables -h' or 'iptables --help' for more information.
root@pc2:/# iptables -A FORWARD -o eth2 -i eth0 -d 192.168.100.0/24 \
> -p udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
root@pc2:/#
```

Hacemos NAT para que el pc pueda hacer Dig.

```
root@pc2: /
root@pc2:/# iptables -A FORWARD -i eth2 -o eth0 -s 192.168.100.0/24 \
> -p udp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth2 -i eth0 -s 192.168.100.0/24 \ -p udp --d
port 53 -m state --state NEW,ESTABLISHED -j ACCEPT
Bad argument '-p'
Try 'iptables -h' or 'iptables --help' for more information.
root@pc2:/# iptables -A FORWARD -o eth2 -i eth0 -d 192.168.100.0/24 \
> -p udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
root@pc2:/# iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o eth2 -j MASQU
RADE
root@pc2:/#
```

```
--- End Startup Commands Log
root@pc1:/# ping 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data.
64 bytes from 192.168.200.1: icmp_seq=1 ttl=64 time=0.202 ms
64 bytes from 192.168.200.1: icmp_seq=2 ttl=64 time=0.109 ms
^C
--- 192.168.200.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 31ms
rtt min/avg/max/mdev = 0.109/0.155/0.202/0.048 ms
root@pc1:/# dig 08.8.8.8 www.openwebinars.net

; <> DiG 9.11.5-P4-5.1+deb10u8-Debian <> 08.8.8.8 www.openwebinars.net
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
root@pc1:/# dig 08.8.8.8 www.openwebinars.net

; <> DiG 9.11.5-P4-5.1+deb10u8-Debian <> 08.8.8.8 www.openwebinars.net
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
root@pc1:/#
```

```
root@pc2: /
root@pc2:/# iptables -A FORWARD -i eth2 -o eth1 -s 192.168.100.0/24 \
> -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth2 -i eth1 -s 192.168.100.0/24 -p tcp --dpo
rt 80 -m state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth2 -o eth1 -s 192.168.100.0/24 -p tcp --dpo
rt 443 -m state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth2 -i eth1 -s 192.168.100.0/24 -p tcp --dpo
rt 443 -m state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/#
```

Habilitamos la navegación web local y pasamos con la parte de la máquina DMZ.

```
root@pc2: /
root@pc2:/# iptables -A FORWARD -i eth1 -o eth1 -p tcp --syn --dport 25 \
> -m connlimit --connlimit-above 2 -j REJECT --reject-with tcp-reset
root@pc2:/# iptables -A FORWARD -i eth1 -o eth1 -p tcp --syn --dport 80 -m connl
imit --connlimit-above 2 -j REJECT --reject-with tcp-reset
root@pc2:/# iptables -A FORWARD -i eth1 -o eth1 -p tcp --syn --dport 443 -m conn
limit --connlimit-above 2 -j REJECT --reject-with tcp-reset
root@pc2:/# iptables -A FORWARD -o eth0 -d 192.168.200.2/24 -p tcp --dport 80 \
> -m state --state NEW,ESTABLISHED -j ACCEPT
iptables v1.8.2 (nf_tables): Chain 'FORWARD' does not exist
root@pc2:/# iptables -A FORWARD -o eth0 -d 192.168.200.2/24 -p tcp --dport 80 -m
state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth0 -d 192.168.200.2/24 -p tcp --dport 80 -m
state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth0 -d 192.168.200.2/24 -p tcp --dport 443 -
m state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth0 -d 192.168.200.2/24 -p tcp --dport 443 -
m state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/# iptables -A FORWARD -o eth0 -d 192.168.200.2/24 -p tcp --dport 25 -m
state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth0 -d 192.168.200.2/24 -p tcp --dport 25 -m
state --state NEW,ESTABLISHED -j ACCEPT
root@pc2:/#
```

Así habría quedado las reglas:

```
root@pc2: /
root@pc2:~# iptables -L FORWARD -n
Chain FORWARD (policy DROP)
target     prot opt source                destination            icmp_type
ACCEPT     icmp -- 0.0.0.0/0              0.0.0.0/0              icmp_type 8
ACCEPT     icmp -- 0.0.0.0/0              0.0.0.0/0              icmp_type 0
ACCEPT     icmp -- 0.0.0.0/0              0.0.0.0/0              icmp_type 8
ACCEPT     icmp -- 0.0.0.0/0              0.0.0.0/0              icmp_type 0
ACCEPT     udp  -- 192.168.100.0/24       0.0.0.0/0              udp dpt:53 state N
EW,ESTABLISHED
ACCEPT     udp  -- 0.0.0.0/0              192.168.100.0/24       udp spt:53 state E
STABLISHED
ACCEPT     tcp  -- 192.168.100.0/24       0.0.0.0/0              tcp dpt:80 state N
EW,ESTABLISHED
ACCEPT     tcp  -- 192.168.100.0/24       0.0.0.0/0              tcp dpt:80 state N
EW,ESTABLISHED
ACCEPT     tcp  -- 192.168.100.0/24       0.0.0.0/0              tcp dpt:443 state
NEW,ESTABLISHED
ACCEPT     tcp  -- 192.168.100.0/24       0.0.0.0/0              tcp dpt:443 state
NEW,ESTABLISHED
REJECT     tcp  -- 0.0.0.0/0              0.0.0.0/0              tcp dpt:25 flags:0
x17/0x02 #conn src/32 > 2 reject-with tcp-reset
REJECT     tcp  -- 0.0.0.0/0              0.0.0.0/0              tcp dpt:80 flags:0
x17/0x02 #conn src/32 > 2 reject-with tcp-reset
REJECT     tcp  -- 0.0.0.0/0              0.0.0.0/0              tcp dpt:443 flags:
0x17/0x02 #conn src/32 > 2 reject-with tcp-reset
ACCEPT     tcp  -- 0.0.0.0/0              192.168.200.0/24       tcp dpt:80 state N
EW,ESTABLISHED
ACCEPT     tcp  -- 0.0.0.0/0              192.168.200.0/24       tcp dpt:80 state N
EW,ESTABLISHED
ACCEPT     tcp  -- 0.0.0.0/0              192.168.200.0/24       tcp dpt:443 state
NEW,ESTABLISHED
ACCEPT     tcp  -- 0.0.0.0/0              192.168.200.0/24       tcp dpt:443 state
NEW,ESTABLISHED
ACCEPT     tcp  -- 0.0.0.0/0              192.168.200.0/24       tcp dpt:25 state N
EW,ESTABLISHED
ACCEPT     tcp  -- 0.0.0.0/0              192.168.200.0/24       tcp dpt:25 state N
EW,ESTABLISHED
root@pc2:~#
```

Realizamos un nat DNAT para tener acceso desde el exterior:

```
root@pc2: /
root@pc2:~# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j DNAT --to 192.168.200.2
root@pc2:~# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 443 -j DNAT --to 192.168.200.2
root@pc2:~# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 25 -j DNAT --to 192.168.200.2
root@pc2:~#
```

## Nuevas cadenas

Limpiamos todas las políticas que estén establecidas y aplicaremos las reglas drop además de las icmp.

```
root@pc2: /
root@pc2:~# iptables -F
root@pc2:~# iptables -t nat -F
root@pc2:~# iptables -Z
root@pc2:~# iptables -t nat -Z
root@pc2:~# iptables -P INPUT DROP
root@pc2:~# iptables -P OUTPUT DROP
root@pc2:~# iptables -P FORWARD DROP
root@pc2:~# iptables -A OUTPUT -o eth1 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@pc2:~# iptables -A INPUT -i eth1 -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@pc2:~# iptables -A INPUT -i eth2 -p icmp -s 192.168.100.0/24 -j ACCEPT
root@pc2:~# iptables -A OUTPUT -o eth2 -p icmp -d 192.168.100.0/24 -j ACCEPT
root@pc2:~# iptables -A INPUT -i eth0 -p icmp -s 192.168.200.0/24 -j ACCEPT
root@pc2:~# iptables -A OUTPUT -o eth0 -p icmp -d 192.168.200.0/24 -j ACCEPT
root@pc2:~#
```

Crearemos una nueva cadena LAN\_A\_INTERNET.

```
root@pc2: /
root@pc2:~# iptables -N LAN_A_INTERNET
root@pc2:~# iptables -A FORWARD -i eth0 -o eth1 -s 192.168.100.0/24 -m state \
> --state NEW,ESTABLISHED -j LAN_A_INTERNET
root@pc2:~#
```

Simplificamos la escritura de reglas:

```
root@pc2: /
root@pc2:~# iptables -A LAN_A_INTERNET -p udp --dport 53 -j ACCEPT
root@pc2:~#
```

```
root@pc2: /
root@pc2:~# iptables -A LAN_A_INTERNET -p udp --dport 53 -j ACCEPT
root@pc2:~# iptables -A LAN_A_INTERNET -p tcp --dport 80 -j ACCEPT
root@pc2:~# iptables -A LAN_A_INTERNET -p tcp --dport 443 -j ACCEPT
root@pc2:~# iptables -L -nv
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination
  0      0 ACCEPT     icmp -- eth1    *        0.0.0.0/0         0.0.0.0/0
  icmp type 0
  0      0 ACCEPT     icmp -- eth2    *    192.168.100.0/24  0.0.0.0/0
  0      0 ACCEPT     icmp -- eth0    *    192.168.200.0/24  0.0.0.0/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination
  0      0 LAN_A_INTERNET all -- eth0    eth1    192.168.100.0/24  0.0.0.0/0
    state NEW,ESTABLISHED

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination
  0      0 ACCEPT     icmp -- *        eth1    0.0.0.0/0         0.0.0.0/0
  icmp type 8
  0      0 ACCEPT     icmp -- *        eth2    0.0.0.0/0         192.168.100.0/24
  0      0 ACCEPT     icmp -- *        eth0    0.0.0.0/0         192.168.200.0/24

Chain LAN_A_INTERNET (1 references)
  pkts bytes target     prot opt in     out     source            destination
  0      0 ACCEPT     udp  -- *        *        0.0.0.0/0         0.0.0.0/0
  udp dpt:53
  0      0 ACCEPT     udp  -- *        *        0.0.0.0/0         0.0.0.0/0
  udp dpt:53
  0      0 ACCEPT     tcp  -- *        *        0.0.0.0/0         0.0.0.0/0
  tcp dpt:80
  0      0 ACCEPT     tcp  -- *        *        0.0.0.0/0         0.0.0.0/0
  tcp dpt:443
root@pc2:~#
```

Una vez autorizado el tráfico, permitiremos las respuestas creando una nueva cadena y volvemos a aplicar las reglas con la cadena creada.

```
root@pc2: /
root@pc2:~# iptables -N INTERNET_A_LAN
root@pc2:~# iptables -A FORWARD -o eth2 -i eth1 -d 192.168.100.0/24 -m state \
> --state ESTABLISHED -j INTERNET_A_LAN
root@pc2:~# iptables -A INTERNET_A_LAN -p udp --dport 53 -j ACCEPT
root@pc2:~# iptables -A INTERNET_A_LAN -p tcp --dport 80 -j ACCEPT
root@pc2:~# iptables -A INTERNET_A_LAN -p tcp --dport 443 -j ACCEPT
root@pc2:~# iptables -L -nv
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination
  0      0 ACCEPT     icmp -- eth1    *        0.0.0.0/0         0.0.0.0/0
  icmp type 0
  0      0 ACCEPT     icmp -- eth2    *    192.168.100.0/24  0.0.0.0/0
  0      0 ACCEPT     icmp -- eth0    *    192.168.200.0/24  0.0.0.0/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination
  0      0 LAN_A_INTERNET all -- eth0    eth1    192.168.100.0/24  0.0.0.0/0
    state NEW,ESTABLISHED
  0      0 INTERNET_A_LAN all -- eth1    eth2    0.0.0.0/0         192.168.100.0/24
    state ESTABLISHED

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination
  0      0 ACCEPT     icmp -- *        eth1    0.0.0.0/0         0.0.0.0/0
  icmp type 8
  0      0 ACCEPT     icmp -- *        eth2    0.0.0.0/0         192.168.100.0/24
  0      0 ACCEPT     icmp -- *        eth0    0.0.0.0/0         192.168.200.0/24

Chain LAN_A_INTERNET (1 references)
  pkts bytes target     prot opt in     out     source            destination
  0      0 ACCEPT     udp  -- *        *        0.0.0.0/0         0.0.0.0/0
  udp dpt:53
  0      0 ACCEPT     udp  -- *        *        0.0.0.0/0         0.0.0.0/0
  udp dpt:53
  0      0 ACCEPT     tcp  -- *        *        0.0.0.0/0         0.0.0.0/0
  tcp dpt:80
  0      0 ACCEPT     tcp  -- *        *        0.0.0.0/0         0.0.0.0/0
  tcp dpt:443

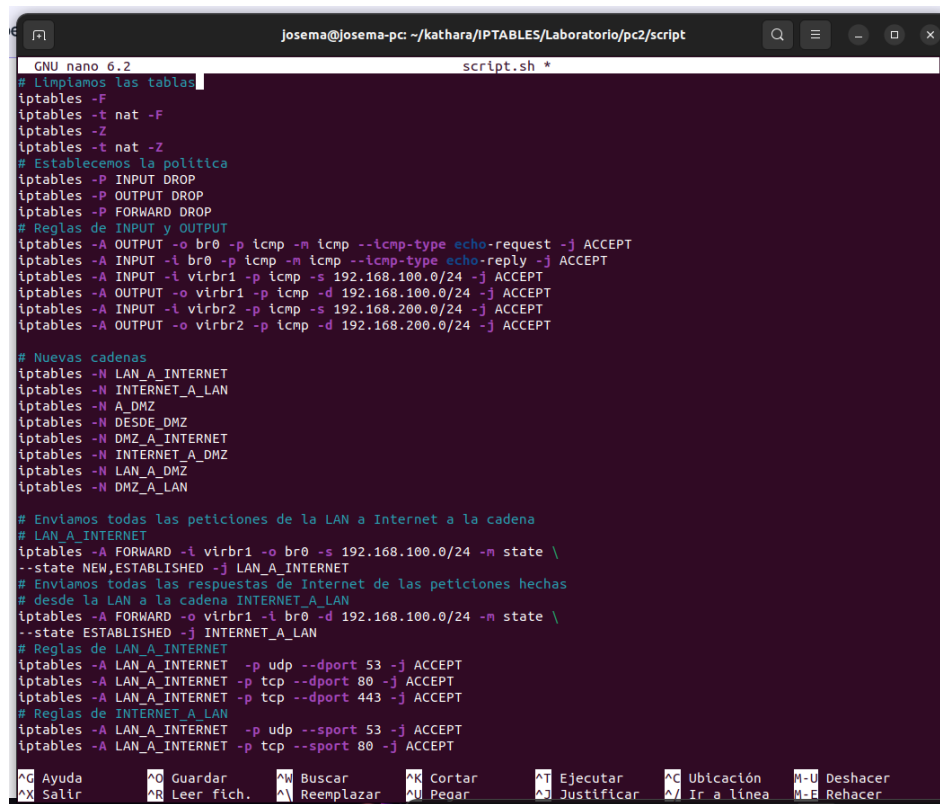
Chain INTERNET_A_LAN (1 references)
  pkts bytes target     prot opt in     out     source            destination
  0      0 ACCEPT     udp  -- *        *        0.0.0.0/0         0.0.0.0/0
  udp dpt:53
  0      0 ACCEPT     tcp  -- *        *        0.0.0.0/0         0.0.0.0/0
  tcp dpt:80
  0      0 ACCEPT     tcp  -- *        *        0.0.0.0/0         0.0.0.0/0
  tcp dpt:443
root@pc2:~#
```

Para la DMZ haremos lo mismo en todas las direcciones, ya sea interna o externa entre la DMZ y la red local.

```
root@pc2: /
root@pc2:/# iptables -N A_DMZ
root@pc2:/# iptables -N DESDE_DMZ
root@pc2:/# iptables -N DMZ_A_INTERNET
root@pc2:/# iptables -N INTERNET_A_DMZ
root@pc2:/# iptables -N LAN_A_DMZ
root@pc2:/# iptables -N DMZ_A_LAN
root@pc2:/# iptables -A FORWARD -o eth0 -d 192.168.200.0/24 -m state \
> --state NEW,ESTABLISHED -j A_DMZ
root@pc2:/#
root@pc2:/# iptables -A A_DMZ -p tcp --dport 25 -j ACCEPT
root@pc2:/# iptables -A A_DMZ -p tcp --dport 80 -j ACCEPT
root@pc2:/# iptables -A A_DMZ -p tcp --dport 443 -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth0 -s 192.168.200.0/24 -m state \
> --state ESTABLISHED -j DESDE_DMZ
iptables v1.8.2 (nf_tables): unknown option "--sate"
Try 'iptables -h' or 'iptables --help' for more information.
root@pc2:/# iptables -A FORWARD -i eth0 -s 192.168.200.0/24 -m state --state ESTABLISHED
-j DESDE_DMZ
root@pc2:/# iptables -A DESDE_DMZ -p tcp --sport 25 -j ACCEPT
root@pc2:/# iptables -A DESDE_DMZ -p tcp --sport 80 -j ACCEPT
root@pc2:/# iptables -A DESDE_DMZ -p tcp --sport 443 -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth0 -o eth1 -s 192.168.200.0/24 \
> -m time --timestart 12:00 --timestop 12:30 -m state --state \
> NEW,ESTABLISHED -j DMZ_A_INTERNET
root@pc2:/#
root@pc2:/# iptables -A DMZ_A_INTERNET -p udp --dport 53 -j ACCEPT
root@pc2:/# iptables -A DMZ_A_INTERNET -p tcp --dport 80 -j ACCEPT
iptables v1.8.2 (nf_tables): unknown protocol "tcp" specified
Try 'iptables -h' or 'iptables --help' for more information.
root@pc2:/# iptables -A DMZ_A_INTERNET -p tcp --dport 80 -j ACCEPT
root@pc2:/# iptables -A DMZ_A_INTERNET -p tcp --dport 443 -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth0 -o eth1 -s 192.168.200.0/24 -m time --timestart 1
2:00 --timestop 12:30 -m state --state NEW,ESTABLISHED -j INTERNET_A_DMZ
root@pc2:/# iptables -A INTERNET_A_DMZ -p udp --dport 53 -j ACCEPT
root@pc2:/# iptables -A INTERNET_A_DMZ -p tcp --dport 80 -j ACCEPT
root@pc2:/# iptables -A INTERNET_A_DMZ -p tcp --dport 443 -j ACCEPT
root@pc2:/# iptables -A FORWARD -i eth2 -o eth0 -s 192.168.100.0/24 \
> -d 192.168.200.0/24 -m state --state NEW,ESTABLISHED -j LAN_A_DMZ
root@pc2:/# iptables -A LAN_A_DMZ -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@pc2:/# iptables -A LAN_A_DMZ -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@pc2:/# iptables -A LAN_A_DMZ -p tcp --sport 3306 -j ACCEPT
root@pc2:/#
root@pc2:/# iptables -A FORWARD -o eth2 -i eth0 -d 192.168.100.0/24 \
> -s 192.168.200.0/24 -m state --state NEW,ESTABLISHED -j DMZ_A_LAN
root@pc2:/# iptables -A DMZ_A_LAN -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@pc2:/# iptables -A DMZ_A_LAN -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@pc2:/# iptables -A DMZ_A_LAN -p tcp --dport 3306 -j ACCEPT
root@pc2:/#
root@pc2:/# iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o br0 -j MASQUERADE
root@pc2:/# iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o br0 -m time \
> --timestart 12:00 --timestop 12:30 -j MASQUERADE
Bad argument 'MASQUERADE'
Try 'iptables -h' or 'iptables --help' for more information.
root@pc2:/# iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o br0 -m time --tim
estop 12:30 -j MASQUERADE
root@pc2:/# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j DNAT --to 192.168.200.2
root@pc2:/# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 443 -j DNAT --to 192.168.200.2
root@pc2:/# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 25 -j DNAT --to 192.168.200.2
root@pc2:/#
```

## Guardamos las iptables

Guardamos todas las reglas que hicimos anteriormente en un fichero para poder ejecutarlo cuando se inicie la máquina para no perder toda la configuración.

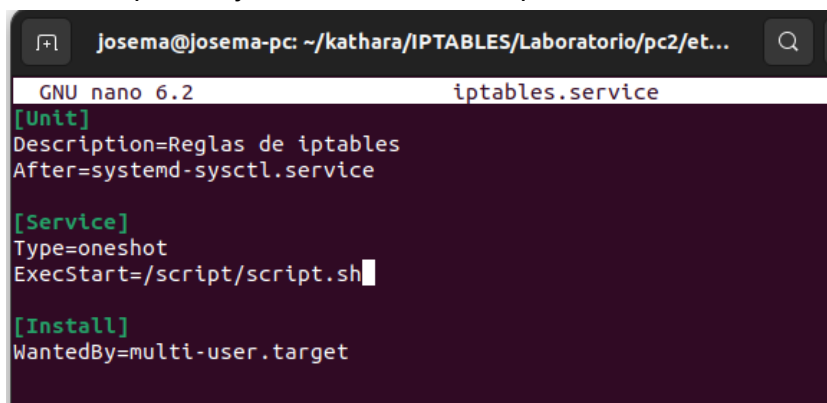


```
GNU nano 6.2 script.sh *
# Limpiamos las tablas
iptables -F
iptables -t nat -F
iptables -Z
iptables -t nat -Z
# Establecemos la política
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
# Reglas de INPUT y OUTPUT
iptables -A OUTPUT -o br0 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT -i br0 -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
iptables -A INPUT -i virbr1 -p icmp -s 192.168.100.0/24 -j ACCEPT
iptables -A OUTPUT -o virbr1 -p icmp -d 192.168.100.0/24 -j ACCEPT
iptables -A INPUT -i virbr2 -p icmp -s 192.168.200.0/24 -j ACCEPT
iptables -A OUTPUT -o virbr2 -p icmp -d 192.168.200.0/24 -j ACCEPT

# Nuevas cadenas
iptables -N LAN_A_INTERNET
iptables -N INTERNET_A_LAN
iptables -N A_DMZ
iptables -N DESDE_DMZ
iptables -N DMZ_A_INTERNET
iptables -N INTERNET_A_DMZ
iptables -N LAN_A_DMZ
iptables -N DMZ_A_LAN

# Enviamos todas las peticiones de la LAN a Internet a la cadena
# LAN_A_INTERNET
iptables -A FORWARD -i virbr1 -o br0 -s 192.168.100.0/24 -m state \
--state NEW,ESTABLISHED -j LAN_A_INTERNET
# Enviamos todas las respuestas de Internet de las peticiones hechas
# desde la LAN a la cadena INTERNET_A_LAN
iptables -A FORWARD -o virbr1 -i br0 -d 192.168.100.0/24 -m state \
--state ESTABLISHED -j INTERNET_A_LAN
# Reglas de LAN A INTERNET
iptables -A LAN_A_INTERNET -p udp --dport 53 -j ACCEPT
iptables -A LAN_A_INTERNET -p tcp --dport 80 -j ACCEPT
iptables -A LAN_A_INTERNET -p tcp --dport 443 -j ACCEPT
# Reglas de INTERNET A LAN
iptables -A LAN_A_INTERNET -p udp --sport 53 -j ACCEPT
iptables -A LAN_A_INTERNET -p tcp --sport 80 -j ACCEPT
```

Crearemos un servicio dentro del directorio /etc/systemd/system/iptables.services el cual haremos que se ejecute al iniciar la máquina.



```
GNU nano 6.2 iptables.service
[Unit]
Description=Reglas de iptables
After=systemd-sysctl.service

[Service]
Type=oneshot
ExecStart=/script/script.sh

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

Guardará todas las reglas y paquetes en un fichero, lo mejor será redireccionar todo este contenido a un fichero para poder ejecutarlo de forma automática.

```
root@pc2:~# iptables-save
# Generated by xtables-save v1.8.2 on Wed Feb 15 20:03:51 2023
*nat
:DOCKER_OUTPUT - [0:0]
:OUTPUT ACCEPT [0:0]
:DOCKER_POSTROUTING - [0:0]
:POSTROUTING ACCEPT [0:0]
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
-A DOCKER_OUTPUT -d 127.0.0.11/32 -p tcp -m tcp --dport 53 -j DNAT --to-destination 127.0.0.11:43657
-A DOCKER_OUTPUT -d 127.0.0.11/32 -p udp -m udp --dport 53 -j DNAT --to-destination 127.0.0.11:35963
-A OUTPUT -d 127.0.0.11/32 -j DOCKER_OUTPUT
-A DOCKER_POSTROUTING -s 127.0.0.11/32 -p tcp -m tcp --sport 43657 -j SNAT --to-source :53
-A DOCKER_POSTROUTING -s 127.0.0.11/32 -p udp -m udp --sport 35963 -j SNAT --to-source :53
-A POSTROUTING -d 127.0.0.11/32 -j DOCKER_POSTROUTING
-A POSTROUTING -s 192.168.100.0/24 -o eth1 -j SNAT --to-source 192.168.50.1
-A POSTROUTING -s 192.168.200.0/24 -o eth1 -j SNAT --to-source 192.168.50.1
COMMIT
# Completed on Wed Feb 15 20:03:51 2023
root@pc2:~# iptables-save
# Generated by xtables-save v1.8.2 on Wed Feb 15 20:03:57 2023
*nat
:DOCKER_OUTPUT - [0:0]
:OUTPUT ACCEPT [0:0]
:DOCKER_POSTROUTING - [0:0]
:POSTROUTING ACCEPT [0:0]
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
-A DOCKER_OUTPUT -d 127.0.0.11/32 -p tcp -m tcp --dport 53 -j DNAT --to-destination 127.0.0.11:43657
-A DOCKER_OUTPUT -d 127.0.0.11/32 -p udp -m udp --dport 53 -j DNAT --to-destination 127.0.0.11:35963
-A OUTPUT -d 127.0.0.11/32 -j DOCKER_OUTPUT
-A DOCKER_POSTROUTING -s 127.0.0.11/32 -p tcp -m tcp --sport 43657 -j SNAT --to-source :53
-A DOCKER_POSTROUTING -s 127.0.0.11/32 -p udp -m udp --sport 35963 -j SNAT --to-source :53
-A POSTROUTING -d 127.0.0.11/32 -j DOCKER_POSTROUTING
-A POSTROUTING -s 192.168.100.0/24 -o eth1 -j SNAT --to-source 192.168.50.1
-A POSTROUTING -s 192.168.200.0/24 -o eth1 -j SNAT --to-source 192.168.50.1
COMMIT
# Completed on Wed Feb 15 20:03:57 2023
root@pc2:~#
```

root@pc2: /script

```
root@pc2:/script# mkdir /etc/iptables
root@pc2:/script# iptables-save > /etc/iptables/reglas.v4
root@pc2:/script# ls /etc/iptables/
reglas.v4
root@pc2:/script# cat /etc/iptables/reglas.v4
# Generated by xtables-save v1.8.2 on Wed Feb 15 20:11:35 2023
*nat
:DOCKER_OUTPUT - [0:0]
:OUTPUT ACCEPT [0:0]
:DOCKER_POSTROUTING - [0:0]
:POSTROUTING ACCEPT [0:0]
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
-A DOCKER_OUTPUT -d 127.0.0.11/32 -p tcp -m tcp --dport 53 -j DNAT --to-destination 127.0.0.11:44229
-A DOCKER_OUTPUT -d 127.0.0.11/32 -p udp -m udp --dport 53 -j DNAT --to-destination 127.0.0.11:57210
-A OUTPUT -d 127.0.0.11/32 -j DOCKER_OUTPUT
-A DOCKER_POSTROUTING -s 127.0.0.11/32 -p tcp -m tcp --sport 44229 -j SNAT --to-source :53
-A DOCKER_POSTROUTING -s 127.0.0.11/32 -p udp -m udp --sport 57210 -j SNAT --to-source :53
-A POSTROUTING -d 127.0.0.11/32 -j DOCKER_POSTROUTING
-A POSTROUTING -s 192.168.100.0/24 -o eth1 -j SNAT --to-source 192.168.50.1
-A POSTROUTING -s 192.168.200.0/24 -o eth1 -j SNAT --to-source 192.168.50.1
COMMIT
# Completed on Wed Feb 15 20:11:35 2023
root@pc2:/script#
```

Restauraremos o cargaremos la configuración del fichero con iptables-restore.

root@pc2: /script

```
root@pc2:/script# iptables-restore /etc/iptables/reglas.v4
root@pc2:/script#
```



Por lo que para que se ejecute de forma automática, modificaremos el fichero .service:

```

┌───┐  josema@josema-pc: ~/kathara/IPTABLES/Laboratorio/pc2/et...  ───┐
│ GNU nano 6.2                                                    iptables.service * │
├───┴──────────────────────────────────────────────────────────────────┴───┘
[Unit]
Description=Reglas de iptables
After=systemd-sysctl.service

[Service]
Type=oneshot
ExecStart=/sbin/iptables-restore /etc/iptables/reglas.v4

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