

IES Fernando Aguilar Quignon

2º ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED

SEGUIMIENTO DEL CURSO IPTABLES.

José María Riol Sánchez

15 de febrero de 2023

Índice

Introducción a la tarea.	2
1. Prueba de vida.	3
1.1. Implementar un cortafuegos personal.	3
1.2. Cortafuegos perimetral parte I.	7
1.3. Cortafuegos perimetral parte II.	9
1.4. Reglas NAT.	10
1.5. Extensiones de IPtables.	11
1.6. Seguimiento de la conexión.	13
1.7. Nuevas cadenas.	16
1.8. Guardando las iptables	19

Introducción a la tarea

Vamos a seguir un curso de [Alberto Molina](#) de OpenWebinars acerca de iptables en el que llevaremos a cabo una serie de ejercicios.

El esquema que usaremos es el que tiene pensado Alberto de forma simplificada para no tener muchas máquinas con las que hacer la demostración, tendremos una DMZ con un equipo servidor que se encontrará en la red **192.168.200.0/24** y luego nuestra red local con nuestro equipo que se encontrará en la red **192.168.100.0/24**. En ambas redes tendremos un Switch para hacer más fiel la simulación, para el caso en el que hubiera más de un equipo en cada subred.

Luego nuestro Router/firewall es el que hace de firewall con 3 interfaces, una para la DMZ, otra para la red interna local y otra con el router isp actuando como firewall perimetral. Por último nuestro isp llamado router es el que estará conectado con el exterior y con nuestro firewall

El laboratorio que usaremos para hacer las prácticas se pueden encontrar en mi [GitHub](#), por si se quiere descargar y hacer uso de él y realizar los ejercicios reflejados en este documento.

Esquema.

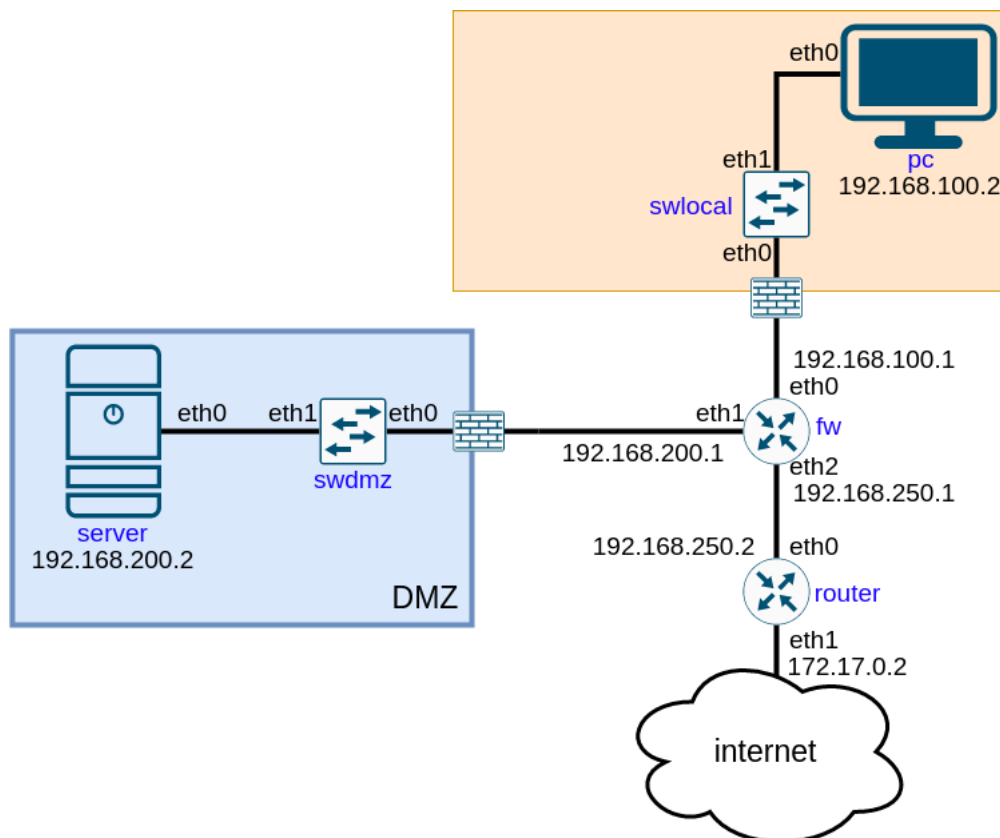


Figura 1: Esquema de la red a usar.

1. Prueba de vida.

Sintaxis de iptables MIN 11:51

1.1. Implementar un cortafuegos personal.

Probamos primero a hacer ping a un DNS (1.1.1.1) y lo hace sin problema, incluso puede navegar por la red sin problemas.

```
root@pc: /

inet 192.168.100.2 netmask 255.255.255.0 broadcast 0.0.0.0
ether 9e:50:18:0b:05:83 txqueuelen 1000 (Ethernet)
RX packets 489 bytes 415664 (405.9 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 8 bytes 604 (604.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

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

root@pc:/# 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=20.5 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=54 time=20.8 ms
^C
--- 1.1.1.1 ping statistics ---
3 packets transmitted, 2 received, 33.3333% packet loss, time 6ms
rtt min/avg/max/mdev = 20.463/20.617/20.771/0.154 ms
root@pc:/#
```

Figura 2: Ping al DNS.

Hacemos un curl porque no tiene GUI nuestra máquina.

```
root@pc: /

root@pc:/# curl https://openwebinars.net | head -n 40
  % Total    % Received % Xferd  Average Speed   Time    Time     Current
                                 Dload  Upload   Total   Spent    Left     Speed
  0   0   0    0    0    0     0      0      0  0  0  0  0  0  0  0  0  0  0
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="google-site-verification" content="3x1l8wvbZfWpW20s_ecG1PPwQuE11U5J9Ko2bbas"/>
<meta property="fb:pages" content="411445362304338"/>
<title>Cursos online de Programación y Sistemas en video | OpenWebinars</title>
<link rel="shortcut icon" href="/static/public/images/favicons/Favicon.ico" sizes="64x64"/>
<link rel="apple-touch-icon" sizes="57x57" href="/static/public/images/favicons/xapple-icon-57x57.png, pagespeed, ic.70yMpP566m.png">
<link rel="apple-touch-icon" sizes="60x60" href="/static/public/images/favicons/xapple-icon-60x60.png, pagespeed, ic.2Mu0sdKt-v.png">
<link rel="apple-touch-icon" sizes="72x72" href="/static/public/images/favicons/xapple-icon-72x72.png, pagespeed, ic.0cX12U6yaF.png">
<link rel="apple-touch-icon" sizes="76x76" href="/static/public/images/favicons/xapple-icon-76x76.png, pagespeed, ic.0H69G2UxLF.png">
<link rel="apple-touch-icon" sizes="114x114" href="/static/public/images/favicons/xapple-icon-114x114.png, pagespeed, ic.r6hp2Wfcl.png">
<link rel="apple-touch-icon" sizes="120x120" href="/static/public/images/favicons/xapple-icon-120x120.png, pagespeed, ic.0VXVsTsPtv.png">
<link rel="apple-touch-icon" sizes="144x144" href="/static/public/images/favicons/xapple-icon-144x144.png, pagespeed, ic.Mzcew6Mskz.png">
<link rel="apple-touch-icon" sizes="152x152" href="/static/public/images/favicons/xapple-icon-152x152.png, pagespeed, ic.a775KwHnKE.png">
<link rel="apple-touch-icon" sizes="180x180" href="/static/public/images/favicons/xapple-icon-180x180.png, pagespeed, ic.oxc109U2mj.png">
<link rel="icon" type="image/png" sizes="192x192" href="/static/public/images/favicons/xandroid-icon-192x192.png, pagespeed, ic.Yq1lxL07Kc.png">
<link rel="icon" type="image/png" sizes="32x32" href="/static/public/images/favicons/xfavicon-32x32.png, pagespeed, ic.lRFJ6u4vw4.png">
<link rel="icon" type="image/png" sizes="96x96" href="/static/public/images/favicons/xfavicon-96x96.png, pagespeed, ic.tlH11T3X4k.png">
<link rel="icon" type="image/png" sizes="16x16" href="/static/public/images/favicons/xfavicon-16x16.png, pagespeed, ic.eLdn8rJyTZ.png">
<link rel="manifest" href="/static/public/images/favicons/manifest.json">
<meta name="msapplication-TileColor" content="#ffffff">
<meta name="msapplication-TileImage" content="/static/public/images/favicons/ms-icon-144x144.png">
<meta name="theme-color" content="#ffffff">
<meta name="robots" content="index, follow">
<link rel="canonical" href="https://openwebinars.net"/>
<meta name="description" content="Aprende tecnología desde cero. Cursos de Ethical Hacking, Cloud Computing, Devops, Big Data, Sistemas, Programación, Frameworks y Metodologías."/>
<!-- Google Authorship and Publisher Markup -->
<link type="text/plain" rel="author" href="http://openwebinars.net/humans.txt"/>
<link rel="alternate" type="application/rss+xml" title="OpenWebinars.net &quot;Feed" href="/feed"/>
<!-- Twitter Card data -->
```

Figura 3: Puede navegar por la red.

Vemos si hay alguna regla para hacer flush o no.

```
root@pc: /
root@pc:/# 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@pc:/#
```

Figura 4: Listado de reglas.

Cambiamos las iptables haciendo que no tenga conexión ninguna.

```
root@pc: /
root@pc:/# iptables -P OUTPUT DROP
root@pc:/# iptables -P INPUT DROP
root@pc:/# 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
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- 1.1.1.1 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 104ms

root@pc:/# curl https://openwebinars.net | head -n 40
% Total % Received % Xferd Average Speed Time Time Time Current
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0curl: (6) Could not resolve host: openwebinars.net
root@pc:/#
```

Figura 5: Cambio en la iptables a drop dejándolo sin conexión.

Si llevamos a cabo un iptables para permitir el ping, esta vez permitirá la operación pero no tendremos respuesta, porque los paquetes que le llegan tienen la política drop entonces no los recibe.

```
root@pc: /
root@pc:/# iptables -A OUTPUT -o eth0 -p icmp -j ACCEPT
root@pc:/# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
^C
--- 1.1.1.1 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 97ms

root@pc:/# 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
    5  420 ACCEPT    icmp  --  *      eth0    0.0.0.0/0               0.0.0.0/0
root@pc:/#
```

Figura 6: Hace ping pero no le llega las respuestas.

Recordemos que las reglas van por pares así que tengo que hacer otra regla INPUT que me permita tener los paquetes de vuelta sin problemas.

```
root@pc: /

root@pc:/# iptables -A INPUT -i eth0 -p icmp -j ACCEPT
root@pc:/# 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=18.8 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=54 time=32.1 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=54 time=20.5 ms
^C
--- 1.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 5ms
rtt min/avg/max/mdev = 18.806/23.799/32.086/5.900 ms
root@pc:/# 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
    8   672 ACCEPT     icmp -- *      eth0    0.0.0.0/0            0.0.0.0/0
root@pc:/#
```

Figura 7: Hace ping pero no le llega las respuestas.

Es bueno observar los paquetes enviados y recibidos en la figura 6 y 7, en la figura 6 tenemos enviados 5 y recibidos ninguno y luego en la figura 7 enviados 8 y recibidos 3.

Las reglas toman la interfaz 0 como entrada/salida, si hacemos ping al 127.0.0.1 que es nuestra propia máquina volverá a ser una operación no permitida.

```
root@pc: /

root@pc:/# 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
ping: sendmsg: Operation not permitted
^C
--- 127.0.0.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 41ms
root@pc:/#
```

Figura 8: Ping interno.

Para permitirlo pondremos las reglas **iptables -A INPUT -i lo -j ACCEPT** y **iptables -A OUTPUT -o lo -j ACCEPT** siendo lo loopback.

No podríamos hacer consultas DNS como por ejemplo con dig, para ello, debemos poner también reglas que permitan ese tipo de tráfico, para ello...

```
root@pc: /

root@pc:/# iptables -A OUTPUT -o eth0 -d 8.8.8.8 -p udp --dport 53 -j ACCEPT
root@pc:/# iptables -A INPUT -i eth0 -s 8.8.8.8 -p udp --sport 53 -j ACCEPT
root@pc:/# dig @8.8.8.8 www.google.es

; <<>> DiG 9.11.5-P4-5.1+deb10u8-Debian <<>> @8.8.8.8 www.google.es
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47127
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.google.es.                190     IN      A      172.217.17.3

;; Query time: 19 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Feb 14 11:39:37 UTC 2023
;; MSG SIZE rcvd: 58

root@pc:/#
```

Figura 9: Consulta DNS.

Seguimos sin tener tráfico web, como podremos observar.

```
root@pc: /

root@pc:/# curl https://openwebinars.net | head -n 40
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
  0     0    0     0    0     0      0      0  --:--:--  0:00:02 --:--:--    0^C
root@pc:/#
```

Figura 10: Sin tráfico web.

Por ello tenemos que añadir las reglas de ACCEPT para los puertos 80 y 443 tanto de entrada como salida.

```
root@pc: /

root@pc:/# curl https://openwebinars.net | head -n 40
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
0    0    0    0    0    0    0  --:--:--  0:00:02 --:--:--    0^C
root@pc:/# iptables -A OUTPUT -o eth0 -p tcp --dport 80 -j ACCEPT
root@pc:/# iptables -A INPUT -i eth0 -p tcp --sport 80 -j ACCEPT
root@pc:/# curl https://openwebinars.net | head -n 40
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
0    0    0    0    0    0    0  --:--:--  0:00:06 --:--:--    0^C
root@pc:/# iptables -A OUTPUT -o eth0 -p tcp --dport 443 -j ACCEPT
root@pc:/# iptables -A INPUT -i eth0 -p tcp --sport 443 -j ACCEPT
root@pc:/# curl https://openwebinars.net | head -n 40
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
0    0    0    0    0    0    0  --:--:--  --:--:--  --:--:--    0

<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="google-site-verification" content="3xIl6rwbzFMPw20s_ecSGlPPwrQuE11U5J9Ko2bbas"/>
<meta property="fb:pages" content="411445362304338"/>
<title>Cursos online de Programación y Sistemas en vídeo | OpenWebinars</title>
<link rel="shortcut icon" href="/static/public/images/favicons/favicon.ico" sizes="64x64"/>
```

Figura 11: Con tráfico web.

1.2. Cortafuegos perimetral parte I.

Antes de todo tenemos que permitir que haya tráfico desde la máquina server a la máquina pc, aunque nosotros por hacer el laboratorio en Kathara no tenemos que hacer este paso, pero siguiendo los pasos de Alberto es necesario activar el bit de forward, de la siguiente forma.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Como hicimos en el ejercicio anterior debemos listar los iptables para limpiar todos los iptables que pudiéramos tener.

Vamos por defecto a dejar una política drop y evitamos cualquier tipo de conexión y luego vamos añadiendo reglas que permitan lo que queramos.

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

root@pc: /
--- Startup Commands Log
++ ip addr add 192.168.100.2/24 dev eth0
++ ip r add default via 192.168.100.1
--- End Startup Commands Log

root@pc:/# ping google.es
PING google.es (142.250.200.67) 56(84) bytes of data:
64 bytes from mad07s24-in-f3.1e100.net (142.250.200.67): icmp_seq=1 ttl=114 time
=19.9 ms
64 bytes from mad07s24-in-f3.1e100.net (142.250.200.67): icmp_seq=2 ttl=114 time
=18.6 ms
^C
--- google.es ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 3ms
rtt min/avg/max/mdev = 18.584/19.247/19.910/0.663 ms
root@pc:/# ping google.es
^C
root@pc:/#
```

Figura 12: Políticas drop.

Vamos a permitir primero el paso de paquetes icmp desde el fw.


```

root@fw: /
root@fw:~# 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@fw:~# iptables -P INPUT DROP
root@fw:~# iptables -P OUTPUT DROP
root@fw:~# iptables -P FORWARD DROP
root@fw:~# iptables -A OUTPUT -o eth0 -p icmp -j ACCEPT
root@fw:~# iptables -A INPUT -i eth0 -p icmp -j ACCEPT
root@fw:~# iptables -A OUTPUT -o eth2 -p icmp -j ACCEPT
root@fw:~# iptables -A INPUT -i eth2 -p icmp -j ACCEPT

```

Figura 13: Permitimos icmp.

Tal y como está ahora, con pc podemos hacer ping a la interfaz de nuestro fw pero no lo atraviesa, es decir, no podemos hacer ping a server, es porque necesitamos las reglas forward.

```

root@fw: /
6 504 ACCEPT icmp -- * eth2 0,0,0,0/0 0,0,0,0/0
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source                   destination
7 588 ACCEPT icmp -- eth0 * 0,0,0,0/0 0,0,0,0/0
6 504 ACCEPT icmp -- eth2 * 0,0,0,0/0 0,0,0,0/0
Chain FORWARD (policy DROP 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
7 588 ACCEPT icmp -- * eth0 0,0,0,0/0 0,0,0,0/0
6 504 ACCEPT icmp -- * eth2 0,0,0,0/0 0,0,0,0/0
root@fw:~#

```

```

root@pc: /
valid_lft forever preferred_lft forever
root@pc:~# ping 192.168.200.2
PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data.
^C
--- 192.168.200.2 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 209ms

root@pc:~# 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=0.121 ms
64 bytes from 192.168.100.1: icmp_seq=2 ttl=64 time=0.085 ms
64 bytes from 192.168.100.1: icmp_seq=3 ttl=64 time=0.068 ms
64 bytes from 192.168.100.1: icmp_seq=4 ttl=64 time=0.056 ms
64 bytes from 192.168.100.1: icmp_seq=5 ttl=64 time=0.080 ms
64 bytes from 192.168.100.1: icmp_seq=6 ttl=64 time=0.067 ms
64 bytes from 192.168.100.1: icmp_seq=7 ttl=64 time=0.081 ms
64 bytes from 192.168.100.1: icmp_seq=8 ttl=64 time=0.082 ms
64 bytes from 192.168.100.1: icmp_seq=9 ttl=64 time=0.058 ms
64 bytes from 192.168.100.1: icmp_seq=10 ttl=64 time=0.064 ms
^C
--- 192.168.100.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 205ms
rtt min/avg/max/mdev = 0.056/0.078/0.121/0.018 ms
root@pc:~#

```

Figura 14: No hay forward.

Procedemos a hacer las reglas de Forward y ver que ahora si se hacen ping.

```

root@fw:~# iptables -A FORWARD -i eth0 -o eth1 -p icmp -j ACCEPT
root@fw:~# iptables -A FORWARD -o eth0 -i eth1 -p icmp -j ACCEPT
root@fw:~# iptables -A FORWARD -i eth2 -o eth1 -p icmp -j ACCEPT
root@fw:~# iptables -A FORWARD -i eth2 -i eth1 -p icmp -j ACCEPT
iptables v1.8.2 (nf_tables): multiple -i flags not allowed
Try 'iptables -h' or 'iptables --help' for more information.
root@fw:~# iptables -A FORWARD -o eth2 -i eth1 -p icmp -j ACCEPT
root@fw:~# iptables -A FORWARD -i eth2 -o eth0 -p icmp -j ACCEPT
root@fw:~# iptables -A FORWARD -o eth2 -i eth0 -p icmp -j ACCEPT
root@fw:~#

```

```

root@pc:~# 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=0.111 ms
64 bytes from 192.168.100.1: icmp_seq=2 ttl=64 time=0.080 ms
64 bytes from 192.168.100.1: icmp_seq=3 ttl=64 time=0.066 ms
^C
--- 192.168.100.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 31ms
rtt min/avg/max/mdev = 0.066/0.085/0.111/0.021 ms

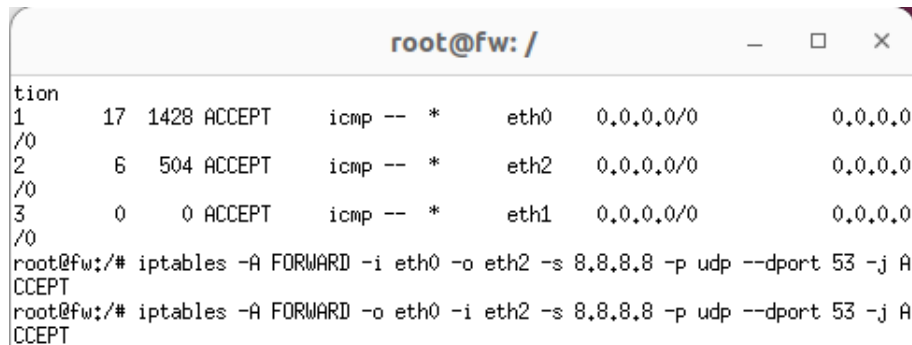
root@pc:~# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=54 time=42.4 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=54 time=25.7 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=54 time=20.9 ms
^C
--- 1.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 7ms
rtt min/avg/max/mdev = 20.920/29.684/42.429/9.221 ms
root@pc:~#

```

Figura 15: Sí hay forward.

Para hacer consultas DNS procedemos de la misma forma. Procedemos a hacer las reglas de Forward

y ver que ahora si se hacen ping.



```
root@fw: /
tion
1      17 1428 ACCEPT    icmp -- *      eth0    0.0.0.0/0      0.0.0.0
/0
2       6  504 ACCEPT    icmp -- *      eth2    0.0.0.0/0      0.0.0.0
/0
3       0   0  ACCEPT    icmp -- *      eth1    0.0.0.0/0      0.0.0.0
/0
root@fw:/# iptables -A FORWARD -i eth0 -o eth2 -s 8.8.8.8 -p udp --dport 53 -j ACCEPT
root@fw:/# iptables -A FORWARD -o eth0 -i eth2 -s 8.8.8.8 -p udp --dport 53 -j ACCEPT
```

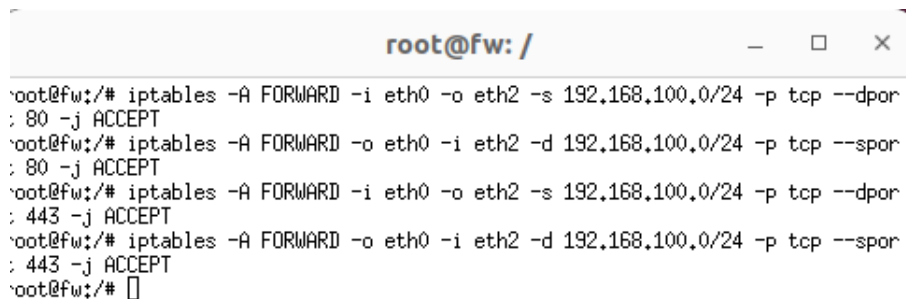
Figura 16: Habilitando consultas DNS.

Si nos sobran reglas tenemos que borrarlas haciendo -D y podemos usar los numeros que listan con la opción --line-numbers.

Podríamos hacer lo mismo para permitirle el paso a la máquina server.

1.3. Cortafuegos perimetral parte II.

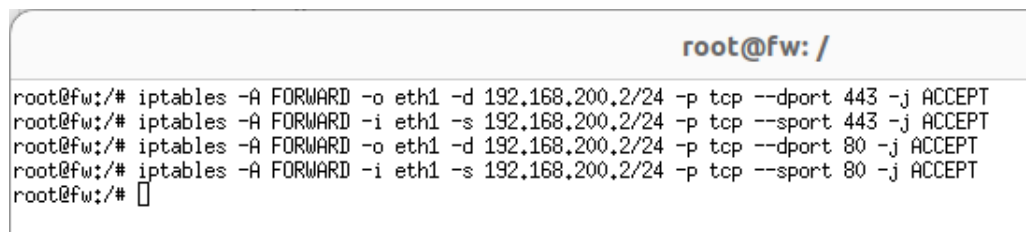
Habilitamos ahora el paso web con las siguientes reglas.



```
root@fw:/# iptables -A FORWARD -i eth0 -o eth2 -s 192.168.100.0/24 -p tcp --dport 80 -j ACCEPT
root@fw:/# iptables -A FORWARD -o eth0 -i eth2 -d 192.168.100.0/24 -p tcp --sport 80 -j ACCEPT
root@fw:/# iptables -A FORWARD -i eth0 -o eth2 -s 192.168.100.0/24 -p tcp --dport 443 -j ACCEPT
root@fw:/# iptables -A FORWARD -o eth0 -i eth2 -d 192.168.100.0/24 -p tcp --sport 443 -j ACCEPT
root@fw:/#
```

Figura 17: Habilitando web.

La máquina que tenemos en la DMZ es inaccesible así que vamos a darle solo acceso de salida con la siguiente regla.



```
root@fw:/# iptables -A FORWARD -o eth1 -d 192.168.200.2/24 -p tcp --dport 443 -j ACCEPT
root@fw:/# iptables -A FORWARD -i eth1 -s 192.168.200.2/24 -p tcp --sport 443 -j ACCEPT
root@fw:/# iptables -A FORWARD -o eth1 -d 192.168.200.2/24 -p tcp --dport 80 -j ACCEPT
root@fw:/# iptables -A FORWARD -i eth1 -s 192.168.200.2/24 -p tcp --sport 80 -j ACCEPT
root@fw:/#
```

Figura 18: Habilitando web a server.

Si tenemos clara la configuración podríamos copiar todas las reglas y pegarlas y de una tenemos todas las reglas que pusimos poco a poco.

1.4. Reglas NAT.

Comenzamos añadiendo la regla NAT que permitirá enmascarar la ip de nuestra red y tener acceso al exterior.

```
root@fw:/ # iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o eth2 -j MASQUERADE
root@fw:/ # iptables -t nat -L -nv
Chain DOCKER_OUTPUT (1 references)
pkts bytes target prot opt in out source destination
0 0 DNAT tcp -- * * 0,0,0,0/0 127,0,0,11 tcp dpt:53 to:127,0,0,11:46625
0 0 DNAT udp -- * * 0,0,0,0/0 127,0,0,11 udp dpt:53 to:127,0,0,11:33060

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
0 0 DOCKER_OUTPUT all -- * * 0,0,0,0/0 127,0,0,11

Chain DOCKER_POSTROUTING (1 references)
pkts bytes target prot opt in out source destination
0 0 SNAT tcp -- * * 127,0,0,11 0,0,0,0/0 tcp spt:46625 to:53
0 0 SNAT udp -- * * 127,0,0,11 0,0,0,0/0 udp spt:33060 to:53

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
0 0 DOCKER_POSTROUTING all -- * * 0,0,0,0/0 127,0,0,11
19 1529 SNAT all -- * eth2 192.168.0,0/16 0,0,0,0/0 to:192.168.250.1
0 0 MASQUERADE all -- * eth2 192.168.100,0/24 0,0,0,0/0

Chain PREROUTING (policy ACCEPT 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
root@fw:/ #
```

Figura 19: NAT y listado.

Nosotros tenemos algunas reglas más pues son las que tenemos en el fichero.startup de nuestro fw.

Especificamos ahora que los paquetes que vengan de fuera con puerto 80, se vayan a la máquina que especificamos.

```
root@fw:/ # iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j DNAT --to 192.168.200.2
root@fw:/ # iptables -t nat -L -nv
Chain DOCKER_OUTPUT (1 references)
pkts bytes target prot opt in out source destination
0 0 DNAT tcp -- * * 0,0,0,0/0 127,0,0,11 tcp dpt:53 to:127,0,0,11:46625
0 0 DNAT udp -- * * 0,0,0,0/0 127,0,0,11 udp dpt:53 to:127,0,0,11:33060

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
0 0 DOCKER_OUTPUT all -- * * 0,0,0,0/0 127,0,0,11

Chain DOCKER_POSTROUTING (1 references)
pkts bytes target prot opt in out source destination
0 0 SNAT tcp -- * * 127,0,0,11 0,0,0,0/0 tcp spt:46625 to:53
0 0 SNAT udp -- * * 127,0,0,11 0,0,0,0/0 udp spt:33060 to:53

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
0 0 DOCKER_POSTROUTING all -- * * 0,0,0,0/0 127,0,0,11
19 1529 SNAT all -- * eth2 192.168.0,0/16 0,0,0,0/0 to:192.168.250.1
0 0 MASQUERADE all -- * eth2 192.168.100,0/24 0,0,0,0/0

Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
0 0 DNAT tcp -- eth1 * 0,0,0,0/0 0,0,0,0/0 tcp dpt:80 to:192.168.200.2

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

Figura 20: NAT y listado.

Procederíamos a hacer lo mismo con el puerto 443 para servicios web por ejemplo.

1.5. Extensiones de IPtables.

Empezamos poniendo una regla más restrictiva que la que pusimos anteriormente, que podremos ver al listar.

```
root@fw: /
root@fw:~# iptables -A OUTPUT -o eth2 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw:~# iptables -L OUTPUT -nv
Chain OUTPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source         destination
    17  1428 ACCEPT      icmp -- *       eth0     0.0.0.0/0      0.0.0.0/0
     6   504 ACCEPT      icmp -- *       eth2     0.0.0.0/0      0.0.0.0/0
     0     0 ACCEPT      icmp -- *       eth1     0.0.0.0/0      0.0.0.0/0
     0     0 ACCEPT      icmp -- *       eth2     0.0.0.0/0      0.0.0.0/0      icmpype 8
root@fw:~#
```

Figura 21: Listado de OUTPUT.

```
root@fw:~# 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=55 time=29.6 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=55 time=18.5 ms
^C
--- 1.1.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 3ms
rtt min/avg/max/mdev = 18.465/24.025/29.586/5.562 ms
root@fw:~# iptables -L OUTPUT -nv
Chain OUTPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source         destination
    17  1428 ACCEPT      icmp -- *       eth0     0.0.0.0/0      0.0.0.0/0
     8   672 ACCEPT      icmp -- *       eth2     0.0.0.0/0      0.0.0.0/0
     0     0 ACCEPT      icmp -- *       eth1     0.0.0.0/0      0.0.0.0/0
     0     0 ACCEPT      icmp -- *       eth2     0.0.0.0/0      0.0.0.0/0      icmpype 8
root@fw:~#
```

Figura 22: Listado de OUTPUT.

Si hacemos ping y listamos vemos que los paquetes obedecen a la regla menos restrictiva y que esté por encima, podríamos en realidad borrar la regla que hemos creado y en vez de hacer append hacemos un insert para que esté arriba del todo.

```
root@fw:~# iptables -D OUTPUT -o eth2 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw:~# iptables -I OUTPUT -o eth2 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw:~# 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=55 time=20.7 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=55 time=24.0 ms
^C
--- 1.1.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 3ms
rtt min/avg/max/mdev = 20.712/22.371/24.031/1.666 ms
root@fw:~# iptables -L OUTPUT -nv
Chain OUTPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source         destination
     2   168 ACCEPT      icmp -- *       eth2     0.0.0.0/0      0.0.0.0/0      icmpype 8
    17  1428 ACCEPT      icmp -- *       eth0     0.0.0.0/0      0.0.0.0/0
     8   672 ACCEPT      icmp -- *       eth2     0.0.0.0/0      0.0.0.0/0
     0     0 ACCEPT      icmp -- *       eth1     0.0.0.0/0      0.0.0.0/0
root@fw:~#
```

Figura 23: Reubicación de la regla y listado.

Si hacemos lo mismo con INPUT y borramos las reglas que teníamos de icmp a cualquiera estoy restringiendo y haciendo unas reglas más restrictivas.

```

root@fw: /
0 0 ACCEPT icmp -- * eth1 0.0.0.0/0 0.0.0.0/0
root@fw: /# iptables -D OUTPUT -o eth2 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw: /# iptables -I OUTPUT -o eth2 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw: /# 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=55 time=20.7 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=55 time=24.0 ms
^C
--- 1.1.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 3ms
rtt min/avg/max/mdev = 20.712/22.371/24.031/1.666 ms
root@fw: /# iptables -I OUTPUT -o eth2 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
Chain OUTPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
2 168 ACCEPT icmp -- * eth2 0.0.0.0/0 0.0.0.0/0 icmp-type 8
17 1428 ACCEPT icmp -- * eth0 0.0.0.0/0 0.0.0.0/0
8 672 ACCEPT icmp -- * eth2 0.0.0.0/0 0.0.0.0/0
0 0 ACCEPT icmp -- * eth1 0.0.0.0/0 0.0.0.0/0
root@fw: /# iptables -I INPUT -i eth2 -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@fw: /# iptables -D INPUT 2
root@fw: /# iptables -I OUTPUT 2
root@fw: /# 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=55 time=25.6 ms
^C
--- 1.1.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 25.610/25.610/25.610/0.000 ms
root@fw: /

root@pc: /
root@pc: /# ping 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data:
^C
--- 192.168.200.1 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 81ms
root@pc: /#

```

Figura 24: Restricción.

Sería conveniente restringir entre la DMZ y nuestra red lan local de la siguiente forma.

```

root@fw: /# iptables -A FORWARD -i eth1 -o eth0 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw: /# iptables -A FORWARD -o eth1 -i eth0 -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@fw: /# iptables -A FORWARD -i eth0 -o eth1 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw: /# iptables -A FORWARD -o eth0 -i eth1 -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@fw: /#

```

Figura 25: Restricción entre DMZ y la LAN.

Sería conveniente limitar las conexiones simultáneas de la siguiente forma.

```

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

```

Figura 26: Limitación de conexiones simultáneas.

Usando el módulo time para poder limitar por tiempos las conexiones. Primero borramos las reglas que ya teníamos y ejecutamos las más restrictivas.

```
root@fw: /

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

Figura 27: Limitación de conexiones simultáneas.

Esos son los tiempos en los que la máquina podría navegar. Otra vez podemos ver que si todo va bien poco a poco, podemos copiarlo todo y hacerlo todo de una.

1.6. Seguimiento de la conexión.

Hacemos primero una limpieza para posteriormente comenzar a poner las reglas de este taller.

```
root@fw: /

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

Figura 28: Limpieza con flush.

Insertamos las reglas referentes a icmp.

```

root@fw: /

root@fw:~# iptables -A OUTPUT -o eth2 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw:~# iptables -A INPUT -i eth2 -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@fw:~# iptables -A INPUT -i eth0 -p icmp -s 192.168.100.0/24 -j ACCEPT
root@fw:~# iptables -A OUTPUT -o eth0 -p icmp -d 192.168.100.0/24 -j ACCEPT
root@fw:~# iptables -A INPUT -i eth1 -p icmp -s 192.168.200.0/24 -j ACCEPT
root@fw:~# iptables -A OUTPUT -o eth1 -p icmp -d 192.168.200.0/24 -j ACCEPT
root@fw:~# iptables -A FORWARD -i eth0 -o eth1 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw:~# iptables -A FORWARD -o eth0 -i eth1 -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@fw:~# iptables -A FORWARD -i eth1 -o eth0 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw:~# iptables -A FORWARD -o eth1 -i eth0 -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@fw:~# █

```

Figura 29: Reglas icmp.

Ahora comenzamos con las reglas de estado.

```

root@fw: /

root@fw:~# iptables -A FORWARD -i eth0 -o eth2 -s 192.168.100.0/24 \
> -p udp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
root@fw:~# iptables -A FORWARD -o eth0 -i eth2 -d 192.168.100.0/24 \
> -p udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
root@fw:~# █

```

Figura 30: Reglas DNS con estado.

Volvemos a hacer el NAT para que pc pueda hacer dig.

```

root@fw: /

root@fw:~# iptables -A FORWARD -i eth0 -o eth2 -s 192.168.100.0/24 \
> -p udp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
root@fw:~# iptables -A FORWARD -o eth0 -i eth2 -d 192.168.100.0/24 \
> -p udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
root@fw:~# iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o eth2 -j MASQUERADE
iptables v1.8.2 (nf_tables): Chain 'POSTROUTING' does not exist
root@fw:~# iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o eth2 -j MASQUERADE
iptables v1.8.2 (nf_tables): Chain 'POSTROUTING' does not exist
root@fw:~# iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o eth2 -j MASQUERADE
root@fw:~# █

```

```

root@pc: /

group default qlen 1000
link/ether 3a2b41da2a2e7115 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@pc:~# dig 0.0.0.0 google
root@pc:~# dig 0.0.0.0 www.openwebinars.net
root@pc:~# dig 0.0.0.0 www.openwebinars.net

;<> DIG 9.11.5-P4-5.1+deb10u8-Ubuntu <> 0.0.0.0 www.openwebinars.net
;; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 5974
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.openwebinars.net.      300     IN      CNAME  openwebinars.net.
openwebinars.net.          300     IN      A      198.199.125.132

;; Query time: 26 msec
;; SERVER: 0.0.0.0#53(0.0.0.0)
;; WHEN: Wed Feb 15 11:38:03 UTC 2023
;; MSG SIZE  rcvd: 79

root@pc:~# █

```

Figura 31: NAT y dig.

```

root@fw: /
root@fw:~# iptables -A FORWARD -i eth0 -o eth2 -s 192.168.100.0/24 \
> -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
root@fw:~# iptables -A FORWARD -o eth0 -i eth2 -d 192.168.100.0/24 \
> -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
root@fw:~# iptables -A FORWARD -i eth0 -o eth2 -s 192.168.100.0/24 \
> -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
root@fw:~# iptables -A FORWARD -o eth0 -i eth2 -d 192.168.100.0/24 \
> -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEPT
root@fw:~# █

```

Figura 32: Reglas web.

Así habilitamos la navegación web local. Vamos ahora con la parte de la DMZ.

```

root@fw: /
root@fw:~# iptables -A FORWARD -i eth2 -o eth1 -p tcp --syn --dport 25 \
> -m connlimit --connlimit-above 2 -j REJECT --reject-with tcp-reset
root@fw:~# iptables -A FORWARD -i eth2 -o eth1 -p tcp --syn --dport 80 \
> -m connlimit --connlimit-above 15 -j REJECT --reject-with tcp-reset
root@fw:~# iptables -A FORWARD -i eth2 -o eth1 -p tcp --syn --dport 443 \
> -m connlimit --connlimit-above 15 -j REJECT --reject-with tcp-reset
root@fw:~# iptables -A FORWARD -o eth1 -d 192.168.200.2/32 -p tcp --dport 80 \
> -m state --state NEW,ESTABLISHED -j ACCEPT
root@fw:~# iptables -A FORWARD -i eth1 -s 192.168.200.2/32 -p tcp --sport 80 \
> -m state --state ESTABLISHED -j ACCEPT
root@fw:~# iptables -A FORWARD -o eth1 -d 192.168.200.2/32 -p tcp --dport 443 \
> -m state --state NEW,ESTABLISHED -j ACCEPT
root@fw:~# iptables -A FORWARD -i eth1 -s 192.168.200.2/32 -p tcp --sport 443 \
> -m state --state ESTABLISHED -j ACCEPT
root@fw:~# iptables -A FORWARD -o eth1 -d 192.168.200.2/32 -p tcp --dport 25 \
> -m state --state NEW,ESTABLISHED -j ACCEPT
root@fw:~# iptables -A FORWARD -i eth1 -s 192.168.200.2/32 -p tcp --sport 25 \
> -m state --state ESTABLISHED -j ACCEPT
root@fw:~# █

```

Figura 33: Reglas web de la DMZ.

Listamos las reglas.

```

root@fw:~# 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 NEW,ESTABLISHED
ACCEPT     udp -- 0.0.0.0/0              192.168.100.0/24      udp spt:53 state ESTABLISHED
ACCEPT     tcp -- 192.168.100.0/24       0.0.0.0/0             tcp dpt:80 state NEW,ESTABLISHED
ACCEPT     tcp -- 0.0.0.0/0              192.168.100.0/24      tcp spt:80 state ESTABLISHED
ACCEPT     tcp -- 192.168.100.0/24     0.0.0.0/0             tcp dpt:443 state NEW,ESTABLISHED
ACCEPT     tcp -- 0.0.0.0/0              192.168.100.0/24      tcp spt:443 state ESTABLISHED
REJECT     tcp -- 0.0.0.0/0              0.0.0.0/0             tcp dpt:25 flags:0x17/0x02 #conn src/32 > 2 reject-with tcp-reset
REJECT     tcp -- 0.0.0.0/0              0.0.0.0/0             tcp dpt:80 flags:0x17/0x02 #conn src/32 > 15 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 > 15 reject-with tcp-reset
ACCEPT     tcp -- 0.0.0.0/0              192.168.200.2         tcp dpt:80 state NEW,ESTABLISHED
ACCEPT     tcp -- 192.168.200.2     0.0.0.0/0             tcp spt:80 state ESTABLISHED
ACCEPT     tcp -- 0.0.0.0/0              192.168.200.2         tcp dpt:443 state NEW,ESTABLISHED
ACCEPT     tcp -- 192.168.200.2     0.0.0.0/0             tcp spt:443 state ESTABLISHED
ACCEPT     tcp -- 0.0.0.0/0              192.168.200.2         tcp dpt:25 state NEW,ESTABLISHED
ACCEPT     tcp -- 192.168.200.2     0.0.0.0/0             tcp spt:25 state ESTABLISHED
root@fw:~# █

```

Figura 34: Reglas web de la DMZ.

Y antes de terminar tenemos que hacer el nat DNAT, para que tenga acceso desde el exterior, con las reglas siguientes:


```

root@fw:/# iptables -t nat -A PREROUTING -i eth2 -p tcp --dport 80 -j DNAT --to 192.168.200.2
root@fw:/# iptables -t nat -A PREROUTING -i eth2 -p tcp --dport 443 -j DNAT --to 192.168.200.2
root@fw:/# iptables -t nat -A PREROUTING -i eth2 -p tcp --dport 25 -j DNAT --to 192.168.200.2
root@fw:/# █

```

Figura 35: Añadiendo el DNAT.

1.7. Nuevas cadenas.

Como en el anterior apartado comenzamos limpiando y aplicando las políticas drop además de meter las reglas de ICMP. A partir de aquí tenemos lo nuevo.

```

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

```

Figura 36: COnfiguración inicial.

Nueva cadena LAN a INTERNET (en realidad es -i eth0, no eth1 en este caso de la imagen)

```

root@fw: /
root@fw:/# iptables -N LAN_A_INTERNET
root@fw:/# iptables -A FORWARD -i eth1 -o eth2 -s 192.168.100.0/24 -m state \
> --state NEW,ESTABLISHED -j LAN_A_INTERNET
root@fw:/# █

```

Figura 37: Nueva cadena creada.

Esto simplifica la escritura de reglas, como podemos ver en la siguiente imagen.

```

root@fw: /
root@fw:/# iptables -A LAN_A_INTERNET -p udp --dport 53 -j ACCEPT
root@fw:/# █

```

Figura 38: Nueva cadena creada.

```

root@fw: /
root@fw:~# iptables -A LAN_A_INTERNET -p udp --dport 53 -j ACCEPT
root@fw:~# iptables -A LAN_A_INTERNET -p tcp --dport 80 -j ACCEPT
root@fw:~# iptables -A LAN_A_INTERNET -p tcp --dport 443 -j ACCEPT
root@fw:~# iptables -L -nv
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination           icmpype 0
    0    0 ACCEPT      icmp -- eth2    *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT      icmp -- eth0    *       192.168.100.0/24     0.0.0.0/0
    0    0 ACCEPT      icmp -- eth1    *       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           state
    0    0 LAN_A_INTERNET all -- eth0    eth2    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           icmpype 8
    0    0 ACCEPT      icmp -- *      eth2    0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT      icmp -- *      eth0    0.0.0.0/0            192.168.100.0/24
    0    0 ACCEPT      icmp -- *      eth1    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           udp dpt:53
    0    0 ACCEPT      udp -- *      *       0.0.0.0/0            0.0.0.0/0
    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@fw:~#

```

Figura 39: Nueva cadena creada.

Tenemos autorizado un sentido el tráfico, ahora tenemos que permitir las respuestas, por lo que creamos una nueva cadena y volveremos a aplicar las reglas con esa cadena nueva creada.

```

root@fw: /
root@fw:~# iptables -N INTERNET_A_LAN
root@fw:~# iptables -A FORWARD -o eth0 -i eth2 -d 192.168.100.0/24 -m state \
> --state ESTABLISHED -j INTERNET_A_LAN
root@fw:~# iptables -A INTERNET_A_LAN -p udp --dport 53 -j ACCEPT
root@fw:~# iptables -A INTERNET_A_LAN -p tcp --dport 80 -j ACCEPT
root@fw:~# iptables -A INTERNET_A_LAN -p tcp --dport 443 -j ACCEPT
root@fw:~# iptables -L -nv
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination           icmpype 0
    0    0 ACCEPT      icmp -- eth2    *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT      icmp -- eth0    *       192.168.100.0/24     0.0.0.0/0
    0    0 ACCEPT      icmp -- eth1    *       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           state
    0    0 LAN_A_INTERNET all -- eth0    eth2    192.168.100.0/24     0.0.0.0/0            state NEW,ESTABLISHED
    0    0 INTERNET_A_LAN all -- eth2    eth0    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           icmpype 8
    0    0 ACCEPT      icmp -- *      eth2    0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT      icmp -- *      eth0    0.0.0.0/0            192.168.100.0/24
    0    0 ACCEPT      icmp -- *      eth1    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           udp dpt:53
    0    0 ACCEPT      udp -- *      *       0.0.0.0/0            0.0.0.0/0
    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           udp dpt:53
    0    0 ACCEPT      udp -- *      *       0.0.0.0/0            0.0.0.0/0
    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@fw:~#

```

Figura 40: Nueva cadena creada.

Haríamos lo mismo con la DMZ en todas las direcciones, tanto para el exterior, como exterior a la DMZ y entre DMZ y red local.

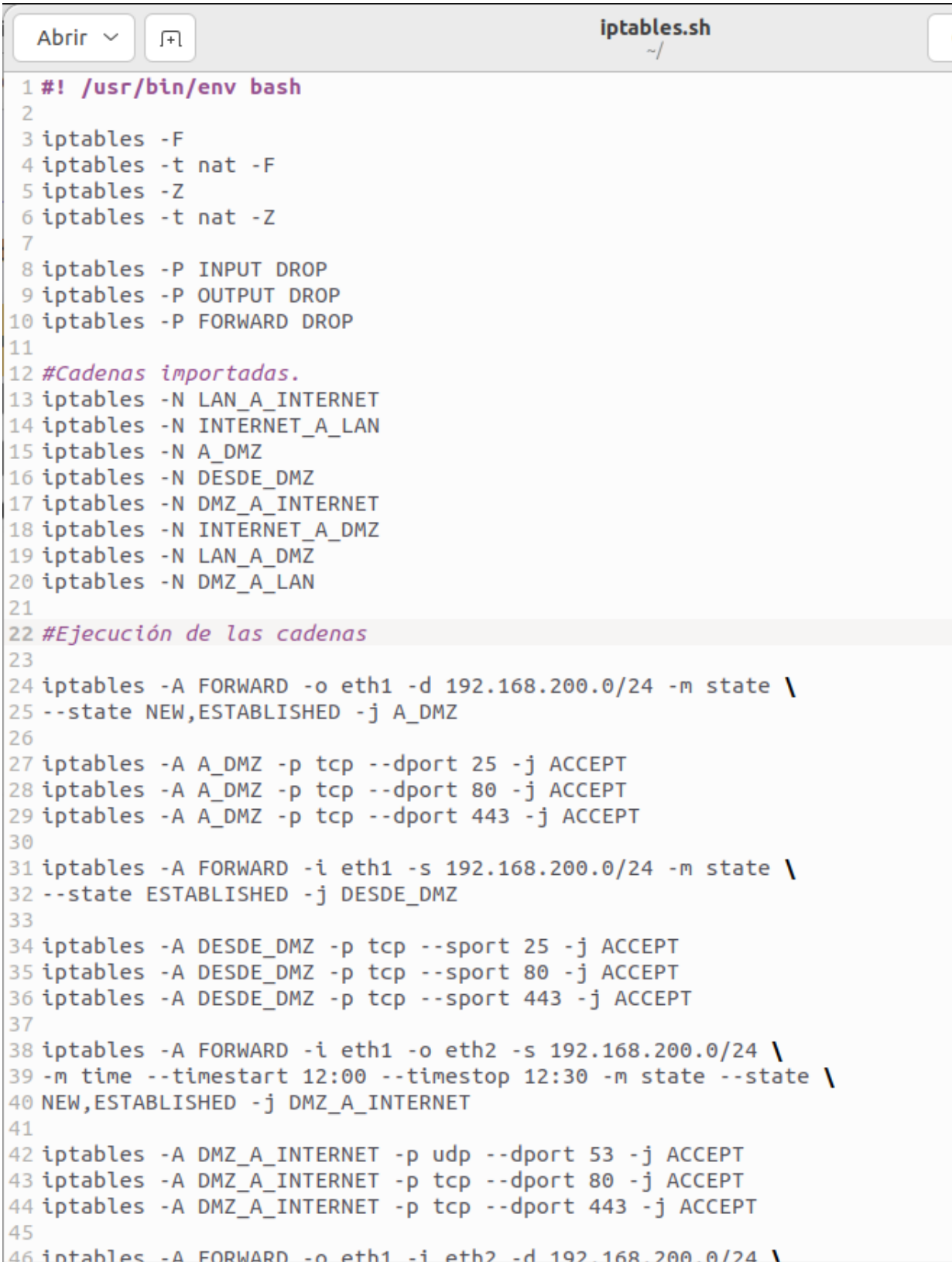
root@fw: /

```
root@fw:/# iptables -N A_DMZ
root@fw:/# iptables -N DESDE_DMZ
root@fw:/# iptables -N DMZ_A_INTERNET
root@fw:/# iptables -N INTERNET_A_DMZ
root@fw:/# iptables -N LAN_A_DMZ
root@fw:/# iptables -N DMZ_A_LAN
root@fw:/# iptables -A FORWARD -o eth1 -d 192.168.200.0/24 -m state \
> --state NEW,ESTABLISHED -j A_DMZ
root@fw:/#
root@fw:/# iptables -A A_DMZ -p tcp --dport 25 -j ACCEPT
root@fw:/# iptables -A A_DMZ -p tcp --dport 80 -j ACCEPT
root@fw:/# iptables -A A_DMZ -p tcp --dport 443 -j ACCEPT
root@fw:/#
root@fw:/# iptables -A FORWARD -i eth1 -s 192.168.200.0/24 -m state \
> --state ESTABLISHED -j DESDE_DMZ
root@fw:/#
root@fw:/# iptables -A DESDE_DMZ -p tcp --sport 25 -j ACCEPT
root@fw:/# iptables -A DESDE_DMZ -p tcp --sport 80 -j ACCEPT
root@fw:/# iptables -A DESDE_DMZ -p tcp --sport 443 -j ACCEPT
root@fw:/#
root@fw:/# iptables -A FORWARD -i eth1 -o eth2 -s 192.168.200.0/24 \
> -m time --timestart 12:00 --timestop 12:30 -m state --state \
> NEW,ESTABLISHED -j DMZ_A_INTERNET
root@fw:/#
root@fw:/# iptables -A DMZ_A_INTERNET -p udp --dport 53 -j ACCEPT
root@fw:/# iptables -A DMZ_A_INTERNET -p tcp --dport 80 -j ACCEPT
root@fw:/# iptables -A DMZ_A_INTERNET -p tcp --dport 443 -j ACCEPT
root@fw:/#
root@fw:/# iptables -A FORWARD -o eth1 -i eth2 -d 192.168.200.0/24 \
> -m time --timestart 12:00 --timestop 12:30 -m state \
> --state ESTABLISHED -j INTERNET_A_DMZ
root@fw:/#
root@fw:/# iptables -A INTERNET_A_DMZ -p udp --dport 53 -j ACCEPT
root@fw:/# iptables -A INTERNET_A_DMZ -p tcp --dport 80 -j ACCEPT
root@fw:/# iptables -A INTERNET_A_DMZ -p tcp --dport 443 -j ACCEPT
root@fw:/#
root@fw:/# iptables -A FORWARD -i eth0 -o eth1 -s 192.168.100.0/24 \
> -d 192.168.200.0/24 -m state --state NEW,ESTABLISHED -j LAN_A_DMZ
root@fw:/#
root@fw:/# iptables -A LAN_A_DMZ -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw:/# iptables -A LAN_A_DMZ -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@fw:/# iptables -A LAN_A_DMZ -p tcp --sport 3306 -j ACCEPT
root@fw:/#
root@fw:/# iptables -A FORWARD -o eth0 -i eth1 -d 192.168.100.0/24 \
> -s 192.168.200.0/24 -m state --state NEW,ESTABLISHED -j DMZ_A_LAN
root@fw:/#
root@fw:/# iptables -A DMZ_A_LAN -p icmp -m icmp --icmp-type echo-request -j ACCEPT
root@fw:/# iptables -A DMZ_A_LAN -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
root@fw:/# iptables -A DMZ_A_LAN -p tcp --dport 3306 -j ACCEPT
root@fw:/#
root@fw:/#
root@fw:/# iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o br0 -j MASQUERADE
root@fw:/# iptables -t nat -A POSTROUTING -s 192.168.200.0/24 -o br0 -m time \
> --timestart 12:00 --timestop 12:30 -j MASQUERADE
root@fw:/# iptables -t nat -A PREROUTING -i eth2 -p tcp --dport 80 -j DNAT --to 192.168.200.2
root@fw:/# iptables -t nat -A PREROUTING -i eth2 -p tcp --dport 443 -j DNAT --to 192.168.200.2
root@fw:/# iptables -t nat -A PREROUTING -i eth2 -p tcp --dport 25 -j DNAT --to 192.168.200.2
root@fw:/# □
```

Figura 41: Nueva cadena creada.

1.8. Guardando las iptables

Lo suyo es usar un fichero donde escribir nuestras reglas.

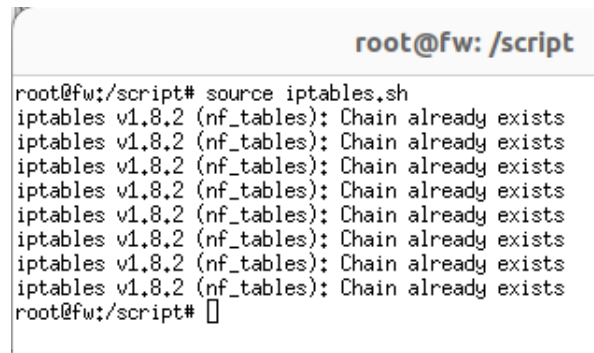


```
1 #! /usr/bin/env bash
2
3 iptables -F
4 iptables -t nat -F
5 iptables -Z
6 iptables -t nat -Z
7
8 iptables -P INPUT DROP
9 iptables -P OUTPUT DROP
10 iptables -P FORWARD DROP
11
12 #Cadenas importadas.
13 iptables -N LAN_A_INTERNET
14 iptables -N INTERNET_A_LAN
15 iptables -N A_DMZ
16 iptables -N DESDE_DMZ
17 iptables -N DMZ_A_INTERNET
18 iptables -N INTERNET_A_DMZ
19 iptables -N LAN_A_DMZ
20 iptables -N DMZ_A_LAN
21
22 #Ejecución de las cadenas
23
24 iptables -A FORWARD -o eth1 -d 192.168.200.0/24 -m state \
25 --state NEW,ESTABLISHED -j A_DMZ
26
27 iptables -A A_DMZ -p tcp --dport 25 -j ACCEPT
28 iptables -A A_DMZ -p tcp --dport 80 -j ACCEPT
29 iptables -A A_DMZ -p tcp --dport 443 -j ACCEPT
30
31 iptables -A FORWARD -i eth1 -s 192.168.200.0/24 -m state \
32 --state ESTABLISHED -j DESDE_DMZ
33
34 iptables -A DESDE_DMZ -p tcp --sport 25 -j ACCEPT
35 iptables -A DESDE_DMZ -p tcp --sport 80 -j ACCEPT
36 iptables -A DESDE_DMZ -p tcp --sport 443 -j ACCEPT
37
38 iptables -A FORWARD -i eth1 -o eth2 -s 192.168.200.0/24 \
39 -m time --timestart 12:00 --timestop 12:30 -m state --state \
40 NEW,ESTABLISHED -j DMZ_A_INTERNET
41
42 iptables -A DMZ_A_INTERNET -p udp --dport 53 -j ACCEPT
43 iptables -A DMZ_A_INTERNET -p tcp --dport 80 -j ACCEPT
44 iptables -A DMZ_A_INTERNET -p tcp --dport 443 -j ACCEPT
45
46 iptables -A FORWARD -o eth1 -i eth2 -d 192.168.200.0/24 \
```

Figura 42: Fichero con reglas.

Para ello reiniciamos el lab en el caso de que estemos con Kathara que es mi caso, introducimos

en la carpeta de fw una carpeta llamada script y ponemos el .sh dentro. Arrancamos el lab y ejecutamos el script.



```
root@fw: /script
root@fw:/script# source iptables.sh
iptables v1.8.2 (nf_tables): Chain already exists
iptables v1.8.2 (nf_tables): Chain already exists
iptables v1.8.2 (nf_tables): Chain already exists
iptables v1.8.2 (nf_tables): Chain already exists
iptables v1.8.2 (nf_tables): Chain already exists
iptables v1.8.2 (nf_tables): Chain already exists
iptables v1.8.2 (nf_tables): Chain already exists
iptables v1.8.2 (nf_tables): Chain already exists
root@fw:/script#
```

Figura 43: Ejecución del script.

Podríamos crear un servicio que iría dentro de `/etc/systemd/system/iptables.service` para que el script se ejecute solo al inicio.



```
Abrir  +  iptables.service ~/
1 [Unit]
2 Description=Reglas de iptables
3 After=systemd-sysctl.service
4
5 [Service]
6 Type=oneshot
7 ExecStart=/script/iptables.sh
8
9 [Install]
10 WantedBy=multi-user.target
```

Figura 44: iptable.service.

Habría que habilitar luego el servicio con el `enable`.

Ahora usaremos la herramienta `iptables-save`

```
root@fw: /script
root@fw:/script# iptables-save
# Generated by xtables-save v1.8.2 on Wed Feb 15 12:29:05 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 POSTROUTING -s 192.168.100.0/24 -o br0 -j MASQUERADE
-A POSTROUTING -s 192.168.200.0/24 -o br0 -m time --timestart 12:00:00 --timesto
p 12:30:00 --datestop 2038-01-19T03:14:07 -j MASQUERADE
-A PREROUTING -i eth2 -p tcp -m tcp --dport 80 -j DNAT --to-destination 192.168.
200.2
-A PREROUTING -i eth2 -p tcp -m tcp --dport 443 -j DNAT --to-destination 192.168
.200.2
-A PREROUTING -i eth -p tcp -m tcp --dport 25 -j DNAT --to-destination 192.168,2
00.2
COMMIT
# Completed on Wed Feb 15 12:29:05 2023
# Generated by xtables-save v1.8.2 on Wed Feb 15 12:29:05 2023
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
:LAN_A_INTERNET - [0:0]
:INTERNET_A_LAN - [0:0]
:A_DMZ - [0:0]
:DESDE_DMZ - [0:0]
:DMZ_A_INTERNET - [0:0]
:INTERNET_A_DMZ - [0:0]
:LAN_A_DMZ - [0:0]
:DMZ_A_LAN - [0:0]
-A FORWARD -d 192.168.200.0/24 -o eth1 -m state --state NEW,ESTABLISHED -j A_DMZ
-A FORWARD -s 192.168.200.0/24 -i eth1 -m state --state ESTABLISHED -j DESDE_DMZ
-A FORWARD -s 192.168.200.0/24 -i eth1 -o eth2 -m time --timestart 12:00:00 --ti
mestop 12:30:00 --datestop 2038-01-19T03:14:07 -m state --state NEW,ESTABLISHED
-j DMZ_A_INTERNET
-A FORWARD -d 192.168.200.0/24 -i eth2 -o eth1 -m time --timestart 12:00:00 --ti
mestop 12:30:00 --datestop 2038-01-19T03:14:07 -m state --state ESTABLISHED -j I
NTERNET_A_DMZ
-A FORWARD -s 192.168.100.0/24 -d 192.168.200.0/24 -i eth0 -o eth1 -m state --st
ate NEW,ESTABLISHED -j LAN_A_DMZ
-A FORWARD -s 192.168.200.0/24 -d 192.168.100.0/24 -i eth1 -o eth0 -m state --st
ate NEW,ESTABLISHED -j DMZ_A_LAN
-A A_DMZ -p tcp -m tcp --dport 25 -j ACCEPT
-A A_DMZ -p tcp -m tcp --dport 80 -j ACCEPT
```

Figura 45: iptables-save.

Esto guarda las reglas e incluso paquetes en un fichero. Aunque lo suyo es redireccionarlo a un fichero acorde a un directorio correcto, por ejemplo, creamos el directorio iptables en ect y lo volcamos en un fichero que estará ahí.

```
root@fw: /script
root@fw:/script# mkdir /etc/iptables
root@fw:/script# iptables-save > /etc/iptables/reglas.v4
root@fw:/script# ls /etc/iptables
reglas.v4
root@fw:/script# cat /etc/iptables/reglas.v4
# Generated by xtables-save v1.8.2 on Wed Feb 15 12:31:52 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 POSTROUTING -s 192.168.100.0/24 -o br0 -j MASQUERADE
-A POSTROUTING -s 192.168.200.0/24 -o br0 -m time --timestart 12:00:00 --timestop 12:30:00 --datestop 2038-01-19T03:14:07 -j MASQUERADE
-A PREROUTING -i eth2 -p tcp -m tcp --dport 80 -j DNAT --to-destination 192.168.200.2
-A PREROUTING -i eth2 -p tcp -m tcp --dport 443 -j DNAT --to-destination 192.168.200.2
-A PREROUTING -i eth -p tcp -m tcp --dport 25 -j DNAT --to-destination 192.168.200.2
COMMIT
# Completed on Wed Feb 15 12:31:52 2023
# Generated by xtables-save v1.8.2 on Wed Feb 15 12:31:52 2023
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
:LAN_A_INTERNET - [0:0]
:INTERNET_A_LAN - [0:0]
:A_DMZ - [0:0]
:DESDE_DMZ - [0:0]
:DMZ_A_INTERNET - [0:0]
:INTERNET_A_DMZ - [0:0]
:LAN_A_DMZ - [0:0]
:DMZ_A_LAN - [0:0]
-A FORWARD -d 192.168.200.0/24 -o eth1 -m state --state NEW,ESTABLISHED -j A_DMZ
-A FORWARD -s 192.168.200.0/24 -i eth1 -m state --state ESTABLISHED -j DESDE_DMZ
-A FORWARD -s 192.168.200.0/24 -i eth1 -o eth2 -m time --timestart 12:00:00 --timestop 12:30:00 --datestop 2038-01-19T03:14:07 -m state --state NEW,ESTABLISHED -j DMZ_A_INTERNET
-A FORWARD -d 192.168.200.0/24 -i eth2 -o eth1 -m time --timestart 12:00:00 --timestop 12:30:00 --datestop 2038-01-19T03:14:07 -m state --state ESTABLISHED -j INTERNET_A_DMZ
-A FORWARD -s 192.168.100.0/24 -d 192.168.200.0/24 -i eth0 -o eth1 -m state --state NEW,ESTABLISHED -j LAN_A_DMZ
```

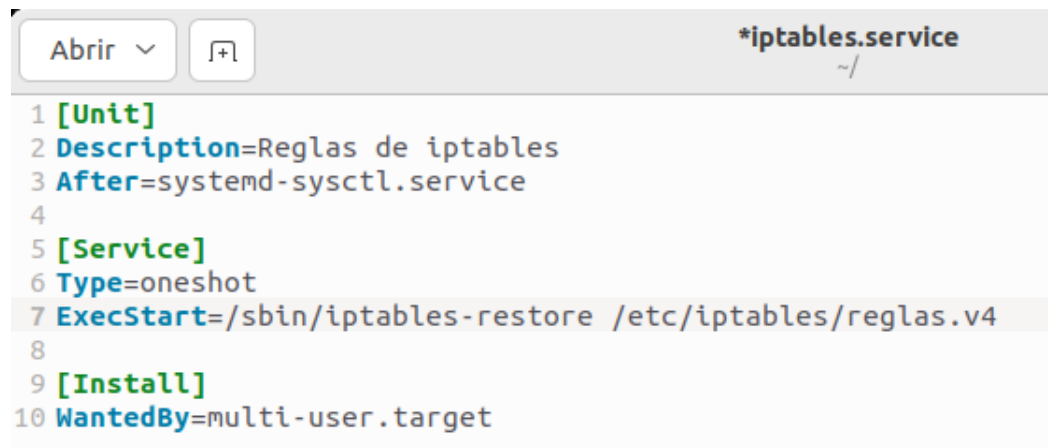
Figura 46: iptables-save.

Podemos restaurar o cargar la configuración del fichero con iptables-restore.

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

Figura 47: iptables-restore.

Como antes lo suyo es que se ejecute automáticamente, para ello vamos a modificar el .service que hicimos.



```
1 [Unit]
2 Description=Reglas de iptables
3 After=systemd-sysctl.service
4
5 [Service]
6 Type=oneshot
7 ExecStart=/sbin/iptables-restore /etc/iptables/reglas.v4
8
9 [Install]
10 WantedBy=multi-user.target
```

Figura 48: iptables-restore como servicio.