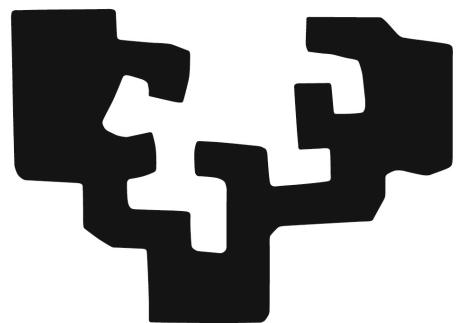


Práctica : Firewall

Grupo 01



UPV EHU

Alex Rivas Machín , Borja Gómez Calvo y Martín López de Ipiña Muñoz

| | |
|--|----|
| Ejercicio 1: Puesta en marcha sin control de accesos..... | 2 |
| Ejercicio 2:..... | 5 |
| Ejercicio 3: Añadir las nuevas reglas del fichero fw-ejerc3.txt..... | 7 |
| Ejercicio 4: Añadir las nuevas reglas del fichero fw-ejerc4.txt..... | 11 |
| Ejercicio 5:..... | 13 |
| Ejercicio 6: Las máquinas LAN no deben de ser directamente visibles de Internet..... | 15 |
| Ejercicio 7: Filtrado entre la red DMZ e INTERNET..... | 17 |
| Ejercicio 8: El servidor web de la DMZ debe tener acceso público desde Internet, pero nunca directamente a su dirección privada 10.10.10.30..... | 19 |
| Tiempo dedicado..... | 21 |

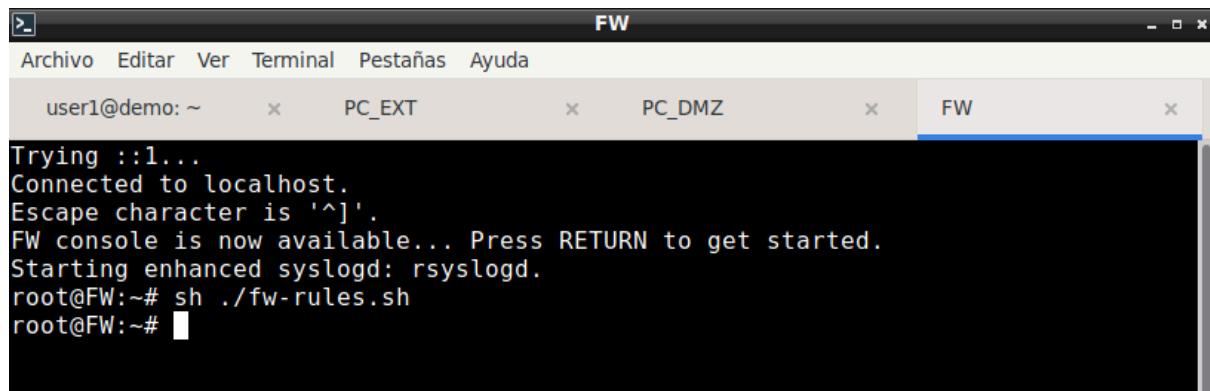
Muchas faltas de ortografía, es recomendable
pasar un corrector ortográfico.

Ejercicio 1: Puesta en marcha sin control de accesos

Todos los dispositivos tienen realizada su configuración IP **3**. Está activado el encaminamiento IP sobre el dispositivo FW (firewall) pero sin reglas de filtrado/traducción definidas.

- Cargar la configuración inicial de las reglas de iptables en FW:

```
sh ./fw-rules.sh
```



```
Trying ::1...
Connected to localhost.
Escape character is '^]'.
FW console is now available... Press RETURN to get started.
Starting enhanced syslogd: rsyslogd.
root@FW:~# sh ./fw-rules.sh
root@FW:~#
```

- Comprobar la configuración de las interfaces (ifconfig) y el contenido de las tablas de encaminamiento (route -n) de los PC's y de FW:

Se ha comprobado la configuración de las interfaces y contenido de las tablas de encaminamiento pero solo se va a mostrar la del FW para abreviarlo.

```

root@FW:~# ifconfig
eth0      Link encap:Ethernet HWaddr 96:c2:08:2a:d4:07
          inet addr:203.0.113.10 Bcast:203.0.113.255 Mask:255.255.255.0
          inet6 addr: fe80::94c2:8ff:fe2a:d407/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:690 (690.0 B) TX bytes:936 (936.0 B)

eth1      Link encap:Ethernet HWaddr 9a:96:5d:c2:74:e2
          inet addr:192.168.1.1 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::9896:5dff:fec2:74e2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:9 errors:0 dropped:1 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:690 (690.0 B) TX bytes:936 (936.0 B)

eth2      Link encap:Ethernet HWaddr 3e:29:5f:a8:bc:41
          inet addr:10.10.10.1 Bcast:10.10.10.255 Mask:255.255.255.0
          inet6 addr: fe80::3c29:5fff:fea8:bc41/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:756 (756.0 B) TX bytes:936 (936.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

root@FW:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref  Use Iface
0.0.0.0         203.0.113.1   0.0.0.0        UG    0      0    0 eth0
10.10.10.0      0.0.0.0       255.255.255.0  U      0      0    0 eth2
192.168.1.0     0.0.0.0       255.255.255.0  U      0      0    0 eth1
203.0.113.0     0.0.0.0       255.255.255.0  U      0      0    0 eth0

```

En el FW podemos observar cómo cada interfaz está asignada a una zona distinta:

- eth0 con IP 203.0.113.10/24 conecta al exterior (Internet), usando 203.0.113.1 como puerta de enlace por defecto;
 - eth1 con IP 192.168.1.1/24 sirve a la red interna (LAN);
 - eth2 con IP 10.10.10.1/24 se destina a la DMZ.
- Además, la tabla de enrutamiento muestra la ruta por defecto (0.0.0.0/0) hacia Internet vía eth0, junto con rutas directas a las subredes LAN y DMZ por sus respectivas interfaces.

- c) Verificar que desde cualquier PC se puede acceder al resto de PC's (ping/traceroute -l):

Ejemplo de acceso desde el PC exterior al PC dentro del NAT

```
root@PC_EXT:~# traceroute -I 192.168.1.10
traceroute to 192.168.1.10 (192.168.1.10), 30 hops max, 60 byte packets
 1  172.16.0.1 (172.16.0.1)  0.933 ms  0.949 ms  0.951 ms
 2  89.54.0.1 (89.54.0.1)  0.951 ms  0.952 ms  0.953 ms
 3  FW (203.0.113.10)  0.954 ms  0.954 ms  0.956 ms
 4  PC_LAN (192.168.1.10)  2.527 ms  2.529 ms  2.531 ms
root@PC_EXT:~#
```

d) Probar acceso a los servidores web desde el PC en la red LAN (wget/lynx):

Acceso al servidor web externo y DMZ desde el PC en la red LAN

```
root@PC_LAN:~# curl 10.10.10.30
<!DOCTYPE html>
<HTML>
<HEAD>
    <TITLE> WEB PC DMZ </TITLE>      sin personalizar con nombre de grupo
</HEAD>

<BODY>
    <H1> WEB PC DMZ </H1>
</BODY>

</HTML>
root@PC_LAN:~# curl 172.16.0.20
<!DOCTYPE html>
<HTML>
<HEAD>
    <TITLE> WEB PC EXT </TITLE>
</HEAD>

<BODY>
    <H1> WEB PC EXT </H1>
</BODY>

</HTML>root@PC_LAN:~#
```

e) Examinar acceso a otros puertos TCP/UDP (nc):

```
root@PC_LAN:~# nc 172.16.0.20 22
(UNKNOWN) [172.16.0.20] 22 (ssh) : Connection refused
root@PC_LAN:~# nc 172.16.0.20 443
(UNKNOWN) [172.16.0.20] 443 (https) : Connection refused
root@PC_LAN:~# nc 172.16.0.20 25
(UNKNOWN) [172.16.0.20] 25 (smtp) : Connection refused
root@PC_LAN:~# Con nc se puede poner en marcha un servicio para comprobar su accesibilidad
```

Esto se debe a que si hacemos (netstat -tau) en los PCs no es que dichos puertos estén bloqueados, sino que no tienen corriendo ningún servicio en ellos.

Ejemplo desde PC_LAN(el resto de PC devuelven el mismo resultado):

```
root@PC_LAN:~# netstat -tau
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp6     0      0  [::]:http              [::]:*                LISTEN
root@PC_LAN:~#
```

Ejercicio 2:

Para iniciar este ejercicio copiamos el contenido de fw-ejer2.txt en donde se nos indica mediante comentarios en fw-rules.sh.

```
cat fw-ejer2.txt
nano fw-rules.sh
#Aqui añadimos donde se nos indica el contenido

sh ./fw-rules.sh

root@FW:~# iptables -L -n -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target  prot opt in     out      source        destination
  0    0 ACCEPT   all  --  lo      *       127.0.0.1      0.0.0.0/0
  0    0 ACCEPT   all  --  lo      *       127.0.1.1      0.0.0.0/0
  0    0 ACCEPT   all  --  lo      *       192.168.1.1    0.0.0.0/0
  0    0 ACCEPT   all  --  lo      *       10.10.10.1     0.0.0.0/0
  0    0 ACCEPT   all  --  lo      *       203.0.113.10   0.0.0.0/0
  0    0 LOG      all  --  *      *       0.0.0.0/0      0.0.0.0/0          LOG flags 0 level 4
prefix "INPUT DROPPED:""

Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target  prot opt in     out      source        destination
  0    0 LOG      all  --  *      *       0.0.0.0/0      0.0.0.0/0          LOG flags 0 level 4
prefix "FORWARD DROPPED:""

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target  prot opt in     out      source        destination
  0    0 ACCEPT   all  --  *      *       127.0.0.1      0.0.0.0/0
  0    0 ACCEPT   all  --  *      *       127.0.1.1      0.0.0.0/0
  0    0 ACCEPT   all  --  *      *       192.168.1.1    0.0.0.0/0
  0    0 ACCEPT   all  --  *      *       10.10.10.1     0.0.0.0/0
  0    0 ACCEPT   all  --  *      *       203.0.113.10   0.0.0.0/0
  0    0 LOG      all  --  *      *       0.0.0.0/0      0.0.0.0/0          LOG flags 0 level 4
prefix "OUTPUT DROPPED:""
```

a) ¿En qué cambia el comportamiento del firewall?

El comportamiento del firewall cambia de forma drástica de forma que **tidas kas** cadenas que teníamos antes pasa a DROP, de forma que todo paquete que no **coincida con una normal** en explícito es descartado.

b) Comprobar la respuesta dada en el apartado anterior.

Para comprobar esto vamos haciendo pruebas desde las maquinas:

Desde PC_LAN:

```
ping -c2 pc_dmz
```

```
root@PC_LAN:~# ping -c2 pc_dmz
PING PC_DMZ (10.10.10.30) 56(84) bytes of data.

--- PC_DMZ ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1022ms
```

```
ping -c2 fw
```

```
root@PC_LAN:~# ping -c2 fw
PING FW (192.168.1.1) 56(84) bytes of data.

--- FW ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1010ms
```

Desde FW

```
ping -c3 8.8.8.8
```

Estamos en una simulación, no hay salida real a Internet

```
root@FW:~# ping -c3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2025ms
```

Para monitorizar:

```
iptables -L INPUT -n -v --line-numbers
iptables -L FORWARD -n -v --line-numbers
```

```
root@FW:~# iptables -L INPUT -n -v --line-numbers
Chain INPUT (policy DROP 5 packets, 504 bytes)
num  pkts bytes target  prot opt in     out      source        destination
1      0   0 ACCEPT   all  --  lo      *       127.0.0.1    0.0.0.0/0
2      0   0 ACCEPT   all  --  lo      *       127.0.1.1    0.0.0.0/0
3      0   0 ACCEPT   all  --  lo      *       192.168.1.1  0.0.0.0/0
4      0   0 ACCEPT   all  --  lo      *       10.10.10.1   0.0.0.0/0
5      0   0 ACCEPT   all  --  lo      *       203.0.113.10 0.0.0.0/0
6      5  504 LOG      all  --  *      *       0.0.0.0/0    0.0.0.0/0          LOG flags 0 lev
el 4 prefix "INPUT DROPPED"
root@FW:~# iptables -L FORWARD -n -v --line-numbers
Chain FORWARD (policy DROP 12 packets, 840 bytes)
num  pkts bytes target  prot opt in     out      source        destination
1      12  840 LOG     all  --  *      *       0.0.0.0/0    0.0.0.0/0          LOG flags 0 lev
el 4 prefix "FORWARD DROPPED"
```

Ejercicio 3: Añadir las nuevas reglas del fichero fw-ejerc3.txt

```
cat fw-ejerc3.txt  
nano fw-rules.sh  
#Aqui añadimos donde se nos indica el contenido  
sh ./fw-rules.sh
```

¿para qué?

```
root@FW:~# iptables -L -n -v  
Chain INPUT (policy DROP 0 packets, 0 bytes)  
pkts bytes target prot opt in out source destination  
0 0 ACCEPT all -- lo * 127.0.0.1 0.0.0.0/0  
0 0 ACCEPT all -- lo * 127.0.1.1 0.0.0.0/0  
0 0 ACCEPT all -- lo * 192.168.1.1 0.0.0.0/0  
0 0 ACCEPT all -- lo * 10.10.10.1 0.0.0.0/0  
0 0 ACCEPT all -- lo * 203.0.113.10 0.0.0.0/0  
0 0 ACCEPT icmp -- eth1 * 192.168.1.0/24 0.0.0.0/0 icmp-type 8 state NEW  
,RELATED,ESTABLISHED  
0 0 ACCEPT icmp -- eth2 * 10.10.10.0/24 0.0.0.0/0 icmp-type 8 state NEW  
,RELATED,ESTABLISHED  
0 0 LOG all -- * * 0.0.0.0/0 0.0.0.0/0 LOG flags 0 level 4  
prefix "INPUT DROPPED:"  
  
Chain FORWARD (policy DROP 0 packets, 0 bytes)  
pkts bytes target prot opt in out source destination  
0 0 LOG all -- * * 0.0.0.0/0 0.0.0.0/0 LOG flags 0 level 4  
prefix "FORWARD DROPPED:"  
  
Chain OUTPUT (policy DROP 0 packets, 0 bytes)  
pkts bytes target prot opt in out source destination  
0 0 ACCEPT all -- * * 127.0.0.1 0.0.0.0/0  
0 0 ACCEPT all -- * * 127.0.1.1 0.0.0.0/0  
0 0 ACCEPT all -- * * 192.168.1.1 0.0.0.0/0  
0 0 ACCEPT all -- * * 10.10.10.1 0.0.0.0/0  
0 0 ACCEPT all -- * * 203.0.113.10 0.0.0.0/0  
0 0 ACCEPT icmp -- * eth1 0.0.0.0/0 192.168.1.0/24 icmp-type 0 state REL  
ATED,ESTABLISHED  
0 0 ACCEPT icmp -- * eth2 0.0.0.0/0 10.10.10.0/24 icmp-type 0 state REL  
ATED,ESTABLISHED  
0 0 LOG all -- * * 0.0.0.0/0 0.0.0.0/0 LOG flags 0 level 4  
prefix "OUTPUT DROPPED:"
```

El objetivo de esto permitir únicamente los pings entre LAN y DMZ hacia las IPs del propio firewall, manteniendo el resto del tráfico bloqueado aún por el ejercicio 2.

- Las dos primeras reglas de ltrado permiten aceptar un comando ping al propio firewall (por la interfaz eth1) proveniente de la red LAN. Testar

Desde PC_LAN:

```
ping -c3 192.168.1.1
```

```
root@PC_LAN:~# ping -c3 192.168.1.1  
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.  
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.288 ms  
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.356 ms  
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.533 ms  
  
--- 192.168.1.1 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2033ms  
rtt min/avg/max/mdev = 0.288/0.392/0.533/0.104 ms  
root@PC_LAN:~#
```

En cambio si: **También funciona ¿?**

```
ping -c3 10.10.10.1
```

```
rtt min/avg/max/mdev = 0.276/0.298/0.329/0.029 ms
root@PC_LAN:~# ping -c3 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=2.41 ms
64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=0.311 ms
64 bytes from 10.10.10.1: icmp_seq=3 ttl=64 time=0.565 ms

--- 10.10.10.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2028ms
rtt min/avg/max/mdev = 0.311/1.097/2.417/0.939 ms
```

b) ¿Y las otras dos?

Las otras dos son para permitir aceptar ping desde DMZ hacia FW.

Desde PC_DMZ:

```
ping -c3 10.10.10.1
```

```
root@PC_DMZ:~# ping -c3 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=0.271 ms

64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=0.336 ms
64 bytes from 10.10.10.1: icmp_seq=3 ttl=64 time=0.335 ms

--- 10.10.10.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2013ms
```

```
ping -c3 192.168.1.1
```

```
root@PC_DMZ:~# ping -c3 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.288 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.299 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.290 ms

--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.288/0.292/0.299/0.014 ms
```

En los dos apartados anteriores hemos comprobado que ambos pueden hacer ping al firewall pero con las reglas establecidas en principio no se podría hacer de LAN a DMZ:

```

root@PC_LAN:~# ping -c3 10.10.10.30
PING 10.10.10.30 (10.10.10.30) 56(84) bytes of data.

--- 10.10.10.30 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2039ms

```

Analizamos los contadores tras las pruebas.

```

root@FW:~# iptables -L INPUT -n -v --line-numbers | grep icmp
6      9 756 ACCEPT    icmp -- eth1   *      192.168.1.0/24      0.0.0.0/0      icmp-type 8 state NEW,RELATED,ESTABLISHED
7      6 504 ACCEPT    icmp -- eth2   *      10.10.10.0/24      0.0.0.0/0      icmp-type 8 state NEW,RELATED,ESTABLISHED
root@FW:~#
root@FW:~# iptables -L OUTPUT -n -v --line-numbers | grep icmp
6      0     0 ACCEPT    icmp -- *      eth1   0.0.0.0/0      192.168.1.0/24      icmp-type 0 state RELATED,ESTABLISHED
7      0     0 ACCEPT    icmp -- *      eth2   0.0.0.0/0      10.10.10.0/24      icmp-type 0 state RELATED,ESTABLISHED
root@FW:~#

```

- c) Si se quiere permitir aceptar un comando ping de la red LAN a la DMZ ¿Qué cadena es necesario modificar? Efectuar la modificación y testar.
FORWARD. El tráfico entre dos interfaces diferentes ($\text{eth1} \leftrightarrow \text{eth2}$) nunca pasa por INPUT/OUTPUT; lo gestiona la cadena FORWARD del *table filter*

```

# 1 Petición LAN → DMZ
$IPTABLES -A FORWARD -i eth1 -o eth2 -s 192.168.1.0/24 -d 10.10.10.0/24 \
-p icmp --icmp-type echo-request -m state --state NEW -j ACCEPT

# 2 Respuesta DMZ → LAN (stateful)
$IPTABLES -A FORWARD -i eth2 -o eth1 -s 10.10.10.0/24 -d 192.168.1.0/24 \
-p icmp --icmp-type echo-reply -m state --state RELATED,ESTABLISHED -j
ACCEPT

```

Editamos y recargamos.

```

nano fw-rules.sh
./fw-rules.sh

```

```

root@PC_LAN:~# ping -c3 10.10.10.30
PING 10.10.10.30 (10.10.10.30) 56(84) bytes of data.
64 bytes from 10.10.10.30: icmp_seq=1 ttl=63 time=0.429 ms

--- 10.10.10.30 ping statistics ---
3 packets transmitted, 1 received, 66% packet loss, time 2024ms
rtt min/avg/max/mdev = 0.429/0.429/0.429/0.000 ms
root@PC_LAN:~#

```

Como podemos ver 66% packet loss, porque? Solo hacer 1 echo-reply. Como lo solucionamos:

```

# ICMP LAN → DMZ (request) y DMZ → LAN (reply)
$IPTABLES -A FORWARD -i eth1 -o eth2 -s 192.168.1.0/24 -d 10.10.10.0/24 \
-p icmp --icmp-type echo-request \
-m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i eth2 -o eth1 -s 10.10.10.0/24 -d 192.168.1.0/24 \
-p icmp --icmp-type echo-reply \

```

```

-m state --state ESTABLISHED,RELATED -j ACCEPT

# ICMP DMZ → LAN (request) y LAN → DMZ (reply) – opcional si se requiere ping
inverso
$IPTABLES -A FORWARD -i eth2 -o eth1 -s 10.10.10.0/24 -d 192.168.1.0/24 \
-p icmp --icmp-type echo-request \
-m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i eth1 -o eth2 -s 192.168.1.0/24 -d 10.10.10.0/24 \
-p icmp --icmp-type echo-reply \
-m state --state ESTABLISHED,RELATED -j ACCEPT

```

Una vez aplicamos:

```

root@PC_LAN:~# ping -c3 10.10.10.30
PING 10.10.10.30 (10.10.10.30) 56(84) bytes of data.
64 bytes from 10.10.10.30: icmp_seq=1 ttl=63 time=2.15 ms
64 bytes from 10.10.10.30: icmp_seq=2 ttl=63 time=0.450 ms
64 bytes from 10.10.10.30: icmp_seq=3 ttl=63 time=0.449 ms

--- 10.10.10.30 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2028ms
rtt min/avg/max/mdev = 0.449/1.017/2.154/0.804 ms
root@PC_LAN:~#

```

d) ¿Es necesario añadir alguna(s) otra(s) regla(s) respecto al comando ping?

No, las dos reglas anteriores son suficientes porque el módulo *state* etiqueta automáticamente la respuesta (*echo-reply*) como ESTABLISHED/RELATED y la segunda regla ya acepta ese tráfico.

Sólo sería necesaria una regla adicional si se quisiera permitir ping en el sentido inverso (DMZ → LAN) o limitar otros tipos ICMP (timestamp, address-mask) con reglas específicas.

Ejercicio 4: Añadir las nuevas reglas del fichero fw-ejerc4.txt

Editamos y aplicamos ejer4.txt

```
root@FW:~# iptables -L -n -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target  prot opt in     out    source          destination
  0   0 ACCEPT   all  --  lo      *       127.0.0.1        0.0.0.0/0
  0   0 ACCEPT   all  --  lo      *       127.0.1.1        0.0.0.0/0
  0   0 ACCEPT   all  --  lo      *       192.168.1.1      0.0.0.0/0
  0   0 ACCEPT   all  --  lo      *       10.10.10.1       0.0.0.0/0
  0   0 ACCEPT   all  --  lo      *       203.0.113.10     0.0.0.0/0
  0   0 ACCEPT   icmp --  eth1   *       192.168.1.0/24    0.0.0.0/0           icmp-type 8 state NEW,RELATED,ESTABL
HED
  0   0 ACCEPT   icmp --  eth2   *       10.10.10.0/24     0.0.0.0/0           icmp-type 8 state NEW,RELATED,ESTABL
HED
  0   0 LOG      all  --  *      *       0.0.0.0/0        0.0.0.0/0           LOG flags 0 level 4 prefix "INPUT D
PPED:"
```



```
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target  prot opt in     out    source          destination
  0   0 ACCEPT   icmp --  eth1   eth2   192.168.1.0/24    10.10.10.0/24      icmp-type 8 state NEW,RELATED,ESTABL
HED
  0   0 ACCEPT   icmp --  eth2   eth1   10.10.10.0/24     192.168.1.0/24      icmp-type 0 state RELATED,ESTABLISHED
  0   0 ACCEPT   icmp --  eth2   eth1   10.10.10.0/24     192.168.1.0/24      icmp-type 8 state NEW,RELATED,ESTABL
HED
  0   0 ACCEPT   icmp --  eth1   eth2   192.168.1.0/24    10.10.10.0/24      icmp-type 0 state RELATED,ESTABLISHED
  0   0 ACCEPT   tcp  --  eth1   eth2   192.168.1.0/24    10.10.10.30       multiport dports 80,443
  0   0 ACCEPT   tcp  --  eth2   eth1   10.10.10.30       192.168.1.0/24      multiport sports 80,443
  0   0 LOG      all  --  *      *       0.0.0.0/0        0.0.0.0/0           LOG flags 0 level 4 prefix "FORWARD
ROPED:"
```



```
Chain OUTPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target  prot opt in     out    source          destination
  0   0 ACCEPT   all  --  *      *       127.0.0.1        0.0.0.0/0
  0   0 ACCEPT   all  --  *      *       127.0.1.1        0.0.0.0/0
  0   0 ACCEPT   all  --  *      *       192.168.1.1      0.0.0.0/0
  0   0 ACCEPT   all  --  *      *       10.10.10.1       0.0.0.0/0
  0   0 ACCEPT   all  --  *      *       203.0.113.10     0.0.0.0/0
  0   0 ACCEPT   icmp --  *      eth1   0.0.0.0/0       192.168.1.0/24      icmp-type 0 state RELATED,ESTABLISHED
  0   0 ACCEPT   icmp --  *      eth2   0.0.0.0/0       10.10.10.0/24      icmp-type 0 state RELATED,ESTABLISHED
  0   0 LOG      all  --  *      *       0.0.0.0/0        0.0.0.0/0           LOG flags 0 level 4 prefix "OUTPUT
OPPED:"
```

a) ¿Cuál es la finalidad de estas reglas?

Sabemos que las reglas son las siguientes:

```
$IPTABLES -A FORWARD -i $LAN_IFACE -s $LAN_IP \
-o $DMZ_IFACE -d $DMZ_WEB_IP \
-p tcp -m multiport --dport 80,443 -j ACCEPT
$IPTABLES -A FORWARD -o $LAN_IFACE -d $LAN_IP \
-i $DMZ_IFACE -s $DMZ_WEB_IP \
-p tcp -m multiport --sport 80,443 -j ACCEPT
```

Estas reglas sirven para habilitar la navegación web de la LAN hacia el servidor DMZ. En el primer caso el origen es LAN hacia DMZ con los perros HTTP y HTTPS de forma que nos permite que cualquier host de la LAN abra este tipo de conexiones al servidor web de la DMZ.

EN el segundo caso el origin sería DMZ y destino lan hacia los mismos puertos, autorizando el tráfico de respuesta del servidor hacia la LAN.

b) Comprobar la respuesta del apartado anterior.

Para probarlo vamos a PC_LAN, y probamos curl sobre la IP de DMZ, debería dar HTTP 200:

```
curl http://10.10.10.30
curl -I http://10.10.10.30
```

```
root@PC_LAN:~# curl http://10.10.10.30
<!DOCTYPE html>
<HTML>
<HEAD>
    <TITLE> WEB PC DMZ </TITLE>
</HEAD>

<BODY>
    <H1> WEB PC DMZ </H1>
</BODY>

</HTML>
root@PC_LAN:~# curl -I http://10.10.10.30
HTTP/1.1 200 OK
Date: Sun, 04 May 2025 23:46:25 GMT
Server: Apache/2.4.10 (Debian)
Last-Modified: Sat, 23 Mar 2019 23:58:36 GMT
ETag: "75-584cbbfe16300"
Accept-Ranges: bytes
Content-Length: 117
Vary: Accept-Encoding
Content-Type: text/html
```

Ejercicio 5:

La DMZ es una red expuesta siendo conveniente aislar la red LAN mediante la traducción de todas las direcciones provenientes de la LAN y que van a la DMZ (no es estrictamente necesario si DMZ también trabaja con direcciones privadas).

- Comprobar que las direcciones de la LAN son visibles en la DMZ. Analizar las tramas.

En FW abrimos una captura:

```
tcpdump -i eth2 -nn host 10.10.10.30 and tcp port 80
```

Y una vez esta escuchando desde PC_LAN:

```
curl -I http://10.10.10.30
```

```
root@FW:~# tcpdump -i eth2 -nn host 10.10.10.30 and tcp port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
23:53:49.810970 IP 192.168.1.10.44370 > 10.10.10.30.80: Flags [S], seq 2958950596, win 64240, options [mss 1460,sackOK,TS val 2286501561 ecr 0,nop,wscale 7]
23:53:49.811097 IP 10.10.10.30.80 > 192.168.1.10.44370: Flags [S.], seq 121851645, ack 65160, options [mss 1460,sackOK,TS val 1979933136,nop,wscale 7]
23:53:49.811184 IP 192.168.1.10.44370 > 10.10.10.30.80: Flags [.], ack 1, win 502, options [nop,nop,TS val 2286501562 ecr 1979933136], length 0
23:53:49.811485 IP 192.168.1.10.44370 > 10.10.10.30.80: Flags [P.], seq 1:77, ack 1, win 502, options [nop,nop,TS val 2286501562 ecr 1979933136], length 76
23:53:49.811562 IP 10.10.10.30.80 > 192.168.1.10.44370: Flags [.], ack 77, win 509, options [nop,nop,TS val 1979933136 ecr 2286501562], length 0
23:53:49.811990 IP 10.10.10.30.80 > 192.168.1.10.44370: Flags [P.], seq 1:252, ack 77, win 509, options [nop,nop,TS val 1979933137 ecr 2286501562], length 251
23:53:49.812068 IP 192.168.1.10.44370 > 10.10.10.30.80: Flags [.], ack 252, win 501, options [nop,nop,TS val 2286501563 ecr 1979933137], length 0
23:53:49.838092 IP 192.168.1.10.44370 > 10.10.10.30.80: Flags [F.], seq 77, ack 252, win 501, options [nop,nop,TS val 2286501587 ecr 1979933137], length 0
23:53:49.838715 IP 10.10.10.30.80 > 192.168.1.10.44370: Flags [F.], seq 252, ack 78, win 509, options [nop,nop,TS val 1979933163 ecr 2286501587], length 0
23:53:49.839065 IP 192.168.1.10.44370 > 10.10.10.30.80: Flags [.], ack 253, win 501, options [nop,nop,TS val 2286501590 ecr 1979933163], length 0
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

- Realizar la traducción de las direcciones entre LAN y DMZ.

Para ello editamos fw-rules.sh y añadimos la siguiente linea antes de el ejercicio 6 junto con el resto de -t nat.

```
$IPTABLES -t nat -A POSTROUTING -s $LAN_IP -o $DMZ_IFACE -j SNAT --to-source
$DMZ_IFACE_IP
```

- Contrastar que las direcciones de la LAN ya no son visibles en la DMZ. Analizar tramas.

Realiamos lo mimo que en el primer arspartado a):

En FW abrimos una captura:

```
tcpdump -i eth2 -nn host 10.10.10.30 and tcp port 80
```

Y una vez esta escuchando desde PC_LAN:

```
curl -I http://10.10.10.30
```

La diferencia en este caso es el campo origen, que aparece como 10.10.10.1

```

root@FW:~# tcpdump -i eth2 -nn host 10.10.10.30 and tcp port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
23:57:22.521496 IP 10.10.10.1.44374 > 10.10.10.30.80: Flags [S], seq 537346764, win 64240, options [mss 1460,sackOK,TS val
23:57:22.521666 IP 10.10.10.30.80 > 10.10.10.1.44374: Flags [S.], seq 219972657, ack 537346765, win 65160, options [mss 146
23:57:22.521761 IP 10.10.10.1.44374 > 10.10.10.30.80: Flags [.], ack 1, win 502, options [nop,nop,TS val 2286714272 ecr 283
23:57:22.525095 IP 10.10.10.1.44374 > 10.10.10.30.80: Flags [P.], seq 1:77, ack 1, win 502, options [nop,nop,TS val 2286714
23:57:22.525244 IP 10.10.10.30.80 > 10.10.10.1.44374: Flags [.], ack 77, win 509, options [nop,nop,TS val 2835795107 ecr 2
23:57:22.526260 IP 10.10.10.30.80 > 10.10.10.1.44374: Flags [P.], seq 1:252, ack 77, win 509, options [nop,nop,TS val 2835795107 ecr 2
23:57:22.526334 IP 10.10.10.1.44374 > 10.10.10.30.80: Flags [.], ack 252, win 501, options [nop,nop,TS val 2286714277 ecr 2
23:57:22.540093 IP 10.10.10.1.44374 > 10.10.10.30.80: Flags [F.], seq 77, ack 252, win 501, options [nop,nop,TS val 2286714277 ecr 2
23:57:22.541665 IP 10.10.10.30.80 > 10.10.10.1.44374: Flags [F.], seq 252, ack 78, win 509, options [nop,nop,TS val 2835795107 ecr 2
23:57:22.541825 IP 10.10.10.1.44374 > 10.10.10.30.80: Flags [..], ack 253, win 501, options [nop,nop,TS val 2286714292 ecr 2
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel

```

d) Consultar el contenido de la tabla NAT.

```

root@FW:~# iptables -t nat -L -n -v --line-numbers
Chain PREROUTING (policy ACCEPT 1 packets, 60 bytes)
num  pkts bytes target     prot opt in     out      source        destination
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source        destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source        destination
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source        destination
1      1    60 SNAT       all   --  *      eth2    192.168.1.0/24    0.0.0.0/0          to:10.10.10.1

```

No es lo que se pide

Ejercicio 6: Las máquinas LAN no deben de ser directamente visibles de Internet

- Añadir reglas que solo permitan a las máquinas de la LAN realizar: ping a maquinas en Internet y accesos a servidores web situados en Internet.

Añadimos el siguiente bloque dentro de las rules:

```
##### INSERTAR AQUÍ reglas ejercicio 6a (FILTER)
$IPTABLES -A FORWARD -i $LAN_IFACE -o $INTERNET_IFACE -s $LAN_IP -d
$ANY_IP \
    -p icmp --icmp-type echo-request -m state --state NEW,ESTABLISHED,RELATED
    -j ACCEPT
$IPTABLES -A FORWARD -i $INTERNET_IFACE -o $LAN_IFACE -d $LAN_IP -p icmp
--icmp-type echo-reply \
    -m state --state ESTABLISHED,RELATED -j ACCEPT

$IPTABLES -A FORWARD -i $LAN_IFACE -o $INTERNET_IFACE -s $LAN_IP -p tcp -m
multiport --dports 80,443 \
    -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $INTERNET_IFACE -o $LAN_IFACE -d $LAN_IP -p tcp -m
state --state ESTABLISHED,RELATED -j ACCEPT
```

- Realizar la traducción de la dirección para todo tráfico que tenga como dirección fuente la red LAN y como destino INTERNET (o cualquier dirección).

Añadimos el siguiente bloque dentro de las rules

```
$IPTABLES -t nat -A POSTROUTING -s $LAN_IP -o $INTERNET_IFACE -j SNAT
--to-source $INTERNET_IFACE_IP
```

- Verificar as reglas definidas comprobando accesos. Examinar las traducciones realizadas (man conntrack).

Una vez realizados los apartados anteriores realizamos pruebas.

Primero hacemos conntrack en FW y despues hacemos ping desde PC_LAN:

```
root@FW:~# conntrack -E
^Cconntrack v1.4.2 (conntrack-tools): 0 flow events have been shown.
root@FW:~# ^C
root@FW:~# conntrack -E
[NEW] icmp      1 30 src=192.168.1.10 dst=203.0.113.1 type=8 code=0 id=580 [UNREPLIED]
pe=0 code=0 id=580
[UPDATE] icmp      1 30 src=192.168.1.10 dst=203.0.113.1 type=8 code=0 id=580 src=203.0.11
id=580
```

Donde podemos ver las traducciones. [conntrack -L](#)

Desde PC_LAN:

```
[root@PC_LAN:~# ping -c3 203.0.113.1
PING 203.0.113.1 (203.0.113.1) 56(84) bytes of data.
64 bytes from 203.0.113.1: icmp_seq=1 ttl=63 time=1.27 ms
64 bytes from 203.0.113.1: icmp_seq=2 ttl=63 time=0.256 ms
64 bytes from 203.0.113.1: icmp_seq=3 ttl=63 time=0.318 ms

--- 203.0.113.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2012ms
rtt min/avg/max/mdev = 0.256/0.617/1.279/0.469 ms
[root@PC_LAN ~]
```

Ejercicio 7: Filtrado entre la red DMZ e INTERNET

- a) El servidor web de la DMZ debe tener acceso público desde Internet con URL <http://203.0.113.10>. Añadir las reglas necesarias para realizar la traducción de la dirección y el puerto (fw-ejerc7.txt).

```
nano fw-rules.sh
```

```
# INSERTAR AQU ^ reglas auxiliares ejercicio 7a
#
# Reglas
# DNAT HTTP
$IPTables -t nat -A PREROUTING \
-i $INTERNET_IFACE \
-d $INTERNET_IFACE_IP \
-p tcp --dport 80 \
-j DNAT --to-destination $DMZ_WEB_IP:80

# DNAT HTTPS
$IPTables -t nat -A PREROUTING \
-i $INTERNET_IFACE \
-d $INTERNET_IFACE_IP \
-p tcp --dport 443 \
-j DNAT --to-destination $DMZ_WEB_IP:443
```

```
sh fw-rules.sh
```

- b) Testar la configuración, accediendo desde el PC de la LAN-EXT a través de un navegador con la URL <http://203.0.113.10>

Desde el PC de la LAN-EXT, haz : curl -I <http://203.0.113.10>

```
root@PC_EXT:~# curl -I http://203.0.113.10
^C
root@PC_EXT:~#
```

Sin reglas de filtrado no pasará porque la política por defecto de FORWARD está en DROP.

- c) ¿Funciona? Solo hemos habilitado las reglas de traducción, es necesario añadir las reglas de filtrado correspondientes. Volver a testar b)

Hay que añadir estas reglas en la tabla filter antes de retestar:

```

# Permitir HTTP entrante (Internet -> DMZ)
$IPTABLES -A FORWARD \
-i $INTERNET_IFACE -o $DMZ_IFACE \
-p tcp --dport 80 -d $DMZ_WEB_IP \
-m state --state NEW,ESTABLISHED \
-j ACCEPT

# Permitir respuesta (DMZ -> Internet)
$IPTABLES -A FORWARD \
-i $DMZ_IFACE -o $INTERNET_IFACE \
-p tcp --sport 80 -s $DMZ_WEB_IP \
-m state --state ESTABLISHED \
-j ACCEPT
#

```

Faltan para 443

```
sh fw-rules.sh
```

Volvemos a probar:

```

root@PC_EXT:~# curl -I http://203.0.113.10
HTTP/1.1 200 OK
Date: Mon, 05 May 2025 16:33:44 GMT
Server: Apache/2.4.10 (Debian)
Last-Modified: Sat, 23 Mar 2019 23:58:36 GMT
ETag: "75-584cbbfe16300"
Accept-Ranges: bytes
Content-Length: 117
Vary: Accept-Encoding
Content-Type: text/html

root@PC_EXT:~# █

```

Ya carga la página web.

Pruebas y capturas con wireshark?

Ejercicio 8: El servidor web de la DMZ debe tener acceso público desde Internet, pero nunca directamente a su dirección privada 10.10.10.30

- Comprobar el acceso al servidor web de la DMZ desde Internet.

Desde una máquina “externa” (LAN-EXT) prueba a acceder directamente a la IP privada del servidor DMZ:

```
curl -I http://10.10.10.30
```

```
root@PC_EXT:~# curl -I http://10.10.10.30
HTTP/1.1 200 OK
Date: Mon, 05 May 2025 16:50:31 GMT
Server: Apache/2.4.10 (Debian)
Last-Modified: Sat, 23 Mar 2019 23:58:36 GMT
ETag: "75-584cbbfe16300"
Accept-Ranges: bytes
Content-Length: 117
Vary: Accept-Encoding
Content-Type: text/html
```

Funciona, por lo que tenemos que aplicar reglas para bloquearlo

- Añadir las reglas necesarias para impedir el acceso desde Internet con direcciones privadas (fw-ejerc8.txt) y vuelve a testar a).

```
nano fw-rules.sh
```

```
# INSERTAR AQU ^ fw-ejerc7.txt
#
# No permitir paquetes con @IPs privadas como destino
# aunque nunca deberian ser ruteables por Internet
$IPTABLES -t nat -A PREROUTING -i $INTERNET_IFACE \
           -d $DMZ_IP -j DNAT --to $LO_IFACE_IP
$IPTABLES -t nat -A PREROUTING -i $INTERNET_IFACE \
           -d $LAN_IP -j DNAT --to $LO_IFACE_IP
#
```

```
sh fw-rules.sh
```

Estas reglas redirigen al loopback del firewall (127.0.0.1) cualquier paquete que llegue por eth0 dirigido a las IPs privadas de la DMZ o de la LAN, de modo que nunca toquen directamente 10.10.10.0/24 ni 192.168.1.0/24.

Volver a testear a):

```
curl -I http://10.10.10.30
```

```
root@PC_EXT:~# curl -I http://10.10.10.30
^C
```

ya no hay respuesta

Ejercicio 9: Incorporar reglas que registren los intentos de accesos no permitidos incluyendo un comentario descriptivo

Para incorporar nuevas reglas de logging para registrar todos los intentos de acceso no permitidos, incluyendo comentarios descriptivos para identificar fácilmente cada tipo de evento.

Desde la máquina virtual anfitriona ejecutamos de esta forma aseguramos que todos los logs se envian a **/var/logs/messages** en principio.

```
sudo sysctl -w net.netfilter.nf_log_all_netns=1
```

```
user1@demo:~$ sudo sysctl -w net.netfilter.nf_log_all_netns=1
net.netfilter.nf_log_all_netns = 1
user1@demo:~$
```

Para añadir los comentarios descriptivos editamos el [rules.sh](#) de nuevo en la y añadimos:

```
# Log accesos SSH rechazados (porta 22)
$IPTABLES -A INPUT -p tcp --dport 22 -j LOG --log-prefix "REJECT_SSH:" --log-level 4
$IPTABLES -A INPUT -p tcp --dport 22 -j REJECT --reject-with tcp-reset

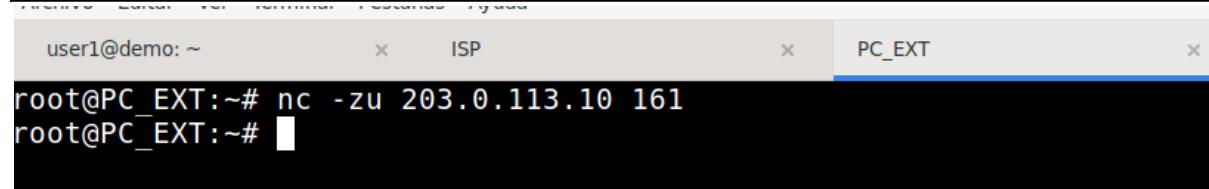
# Log intentos UDP no permitidos (ej.: SNMP)
$IPTABLES -A INPUT -p udp --dport 161 -j LOG --log-prefix "DROP_UDP_161:" --log-level 4

# Log paquetes inválidos (estado INVALID)
$IPTABLES -A INPUT -m conntrack --ctstate INVALID -j LOG --log-prefix "INVALID_PKT:" --log-level 4
```

De esta forma, cada prefijo (REJECT_SSH:, DROP_UDP_161:, INVALID_PKT:) ayudará a identificar el tipo de intento en los logs.

Una vez todo puesto vamos a ponerlo en práctica desde PC_EXT probamos por ejemplo para conexión UDP.

```
nc -zu 203.0.113.10 161 # UDP 161
```



Una vez vemos esto desde FW hacemos:

```
cat /var/log/messages
```

Y vemos que realmente esta el “DROP_UDP_161” de prefijo.

```
May 16 14:56:53 FW kernel: [21283.666321] DROP UDP 161:IN=eth0 OUT= MAC=0a:b2:7a:a3:ae:c2:92:81:cc:23:99:7c:08:0
0.20 DST=203.0.113.10 LEN=29 TOS=0x00 PREC=0x00 TTL=62 ID=41558 DF PROTO=UDP SPT=52027 DPT=161 LEN=9
May 16 14:56:53 FW kernel: [21283.666344] TINPUT_DROPPED:IN=eth0 OUT= MAC=0a:b2:7a:a3:ae:c2:92:81:cc:23:99:7c:08:0
```

Ejercicio 10 [OPC]: Proporner nuevas reglas para bloquear ciertos tipos de paquetes asociados a un posible intento de ataque

Para aparto opcional vamos a añadir bloqueo a patrones de ataques como scans, inundaciones SYN y paquetes inválidos.

Para ellos proponemos las siguientes reglas que se van a introducir en [./fw-rules.sh](#)

```
# INVALID
$IPTABLES -A INPUT -m conntrack --ctstate INVALID -j DROP

# Xmas scan
$IPTABLES -A INPUT -p tcp --tcp-flags ALL ALL -j DROP

# NULL scan
$IPTABLES -A INPUT -p tcp --tcp-flags ALL NONE -j DROP

# NEW que no sean SYN
$IPTABLES -A INPUT -p tcp ! --syn -m conntrack --ctstate NEW -j DROP

# Limitar SYN rate
$IPTABLES -A INPUT -p tcp --syn -m limit --limit 5/sec --limit-burst 10 -j ACCEPT
$IPTABLES -A INPUT -p tcp --syn -j DROP
```

Una vez aplicadas desde PC_LAN haremos nmap y hping3, desgraciadamente se queda en un intento ya que no se puede instalar nmap en las máquinas de GNS3. Pese a eso dejamos la idea.

Tiempo dedicado

El tiempo dedicado total ha sido de 9 horas.

Pasos previos: 1h

Ejercicios: 8h