IPTables

IPTables is a user-space utility program that allows a system administrator to configure the IP packet filter rules of the Linux kernel firewall, implemented as different Netfilter modules. The rules defined by IPTables are used for packet filtering, network address translation (NAT), and port translation, which can be applied to packets at different points in the routing process.

IPTables

- **INPUT:** Filters incoming packets destined for the local system, controlling access to local services like SSH or web servers.
- **FORWARD:** Handles packets being routed through the system, typically used in routers or firewalls to filter traffic between different network segments.
- **OUTPUT:** Filters outgoing packets generated by local processes before they are sent out, controlling outbound connections from the system.

These chains enable granular control over network traffic at different stages: incoming, forwarding, and outgoing.

Vamos a ver cómo configurar iptables con la terminal usando dos máquinas ubuntu, ésta es la máquina 1.

```
user@singular1:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
user@singular1:~$
```

```
user@singular1:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:1c:42:d0:02:5f brd ff:ff:ff:ff:ff:ff
    inet 10.211.55.5/24 brd 10.211.55.255 scope global dynamic eth0
       valid_lft 1225sec preferred_lft 1225sec
    inet6 fdb2:2c26:f4e4:0:21c:42ff:fed0:25f/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 2591933sec preferred_lft 604733sec
    inet6 fe80::21c:42ff:fed0:25f/64 scope link
       valid_lft forever preferred_lft forever
user@singular1:~$
```

Ésta es la máquina 2.

```
user@singular2:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:1c:42:84:e7:e2 brd ff:ff:ff:ff:ff:ff
    inet 10.211.55.17/24 brd 10.211.55.255 scope global dynamic eth0
       valid_lft 1199sec preferred_lft 1199sec
    inet6 fdb2:2c26:f4e4:0:21c:42ff:fe84:e7e2/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 2591909sec preferred_lft 604709sec
    inet6 fe80::21c:42ff:fe84:e7e2/64 scope link
       valid_lft forever preferred_lft forever
user@singular2:~$
```

Comprobamos que están conectadas haciendo un ping.

```
user@singular2:~$ ping 10.211.55.5
PING 10.211.55.5 (10.211.55.5) 56(84) bytes of data.
64 bytes from 10.211.55.5: icmp_seq=1 ttl=64 time=0.767 ms
64 bytes from 10.211.55.5: icmp_seq=2 ttl=64 time=0.403 ms
64 bytes from 10.211.55.5: icmp_seq=3 ttl=64 time=0.648 ms
64 bytes from 10.211.55.5: icmp_seq=4 ttl=64 time=0.547 ms
64 bytes from 10.211.55.5: icmp_seq=5 ttl=64 time=0.403 ms
64 bytes from 10.211.55.5: icmp_seq=6 ttl=64 time=1.83 ms
^C
--- 10.211.55.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5133ms
rtt min/avg/max/mdev = 0.403/0.766/1.828/0.492 ms
user@singular2:~$
```

Empezamos la configuración de iptables en la máquina 2, bloqueando el tráfico.

```
user@singular2:~$ sudo iptables -A INPUT -s 10.211.55.5 -j DROP
[sudo] password for user:
user@singular2:~$ sudo iptables -A OUTPUT -d 10.211.55.5 -j DROP
user@singular2:~$
```

Vemos que funciona porque ya no podemos hacer ping.

```
user@singular2:~$ ping 10.211.55.5
PING 10.211.55.5 (10.211.55.5) 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
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- 10.211.55.5 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9206ms
```

Ahora habilitamos el protocolo ssh para una máxima seguridad y encriptamiento en el input y en el output. Permitimos así el tráfico desde y hacia la máquina 1, pero solo el puerto ssh. Puerto 22 = SSH.

```
user@singular2:~$ sudo iptables -A INPUT -p tcp --dport 22 -s 10.211.55.5 -j ACCEPT
user@singular2:~$ sudo iptables -A OUTPUT -p tcp --sport 22 -d 10.211.55.5 -j ACCEPT
user@singular2:~$
```

Intentamos conectar con la segunda maquina usando el protocolo SSH. Encontramos un problema de jerarquia, porque cómo vemos en el screenshot de debajo, se queda colgado. El problema es que hemos bloqueado el tráfico antes de habilitar SSH, hay que hacerlo al contrario, para que la última regla que se aplique sea la de bloquear pero que ya herede desde la anterior que es la de SSH.

```
user@singular1:~$ ssh user@10.211.55.17
```

Verificamos en la segunda maquina cómo lo hemos configurado. Primero tenemos el DROP dónde se bloquea todo y luego tenemos el ACCEPT dónde se aceptan las conexiones SSH, está al revés.

```
user@singular2:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  ubuntu-linux-1.shared  anywhere
ACCEPT     tcp  --  ubuntu-linux-1.shared  anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             ubuntu-linux-1.shared
ACCEPT     tcp  --  anywhere             ubuntu-linux-1.shared  tcp spt:ssh
user@singular2:~$
```

Decidimos cambiar la jerarquia para que funcione. Primero reseteamos las reglas, luego las listamos de nuevo para ver que efectivamente se han reseteado y no haya ninguna regla aplicada.

```
user@singular2:~$ sudo iptables -F
user@singular2:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
user@singular2:~$
```

Ahora empezamos habilitando SSH antes de bloquear el tráfico y listamos las reglas para verificar que ambas reglas ACCEPT para SSH están activas.

```
user@singular2: ~
user@singular2:~$ sudo iptables -A INPUT -p tcp --dport 22 -s 10.211.55.5 -j ACCEPT
user@singular2:~$ sudo iptables -A OUTPUT -p tcp --sport 22 -d 10.211.55.5 -j ACCEPT
user@singular2:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  ubuntu-linux-1.shared  anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             ubuntu-linux-1.shared  tcp spt:ssh
user@singular2:~$
```

Ahora hacemos el bloqueo de la maquina 1 aplicando la regla DROP con el input y con el output.

```
user@singular2:~$ sudo iptables -A INPUT -s 10.211.55.5 -j DROP
user@singular2:~$ sudo iptables -A OUTPUT -d 10.211.55.5 -j DROP
user@singular2:~$
```

Ahora listamos las reglas y vemos que ahora está con la buena jerarquía.

```
user@singular2:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  ubuntu-linux-1.shared  anywhere             tcp dpt:ssh
DROP       all  --  ubuntu-linux-1.shared  anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             ubuntu-linux-1.shared  tcp spt:ssh
DROP       all  --  anywhere             ubuntu-linux-1.shared
user@singular2:~$
```

En definitiva, la última regla que hay que aplicar es la que lo bloquea todo con DROP, antes ponemos las especificaciones más concretas y las excepciones cómo serían en este caso las conexiones SSH desde la maquina 1.

Ahora vemos que si queremos conectar por ssh una maquina a la otra, ahora si nos deja, porque aunque haya el drop puesto, hemos especificado que las ssh se autoricen.

```
user@singular1:~$ ssh user@10.211.55.17
The authenticity of host '10.211.55.17 (10.211.55.17)' can't be established.
ECDSA key fingerprint is SHA256:8Iiv/8aBCprlS3ohWLFH9iiofaIpwXFjxaRKAT6pZvc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.211.55.17' (ECDSA) to the list of known hosts.
user@10.211.55.17's password:

user@singular1:~$
```

Y vemos que un ping normal no va a hacerse porque está bloqueado con el drop.

```
user@singular1:~$ ping 10.211.55.17
PING 10.211.55.17 (10.211.55.17) 56(84) bytes of data.
```

Vale ahora vamos a ser aún más específicos y aumentar el filtro, para que las conexiones sean sólo SSH pero que sean como máximo 4 conexiones cada minuto, no más.

Primero hay que eliminar el drop para poner las limitaciones extra.

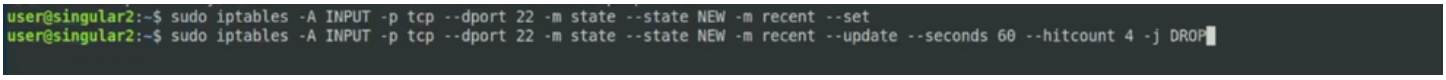
```
user@singular2: ~
user@singular2:~$ sudo iptables -D OUTPUT -d 10.211.55.5 -j DROP
```

```
user@singular2: ~
user@singular2:~$ sudo iptables -D INPUT -s 10.211.55.5 -j DROP
```

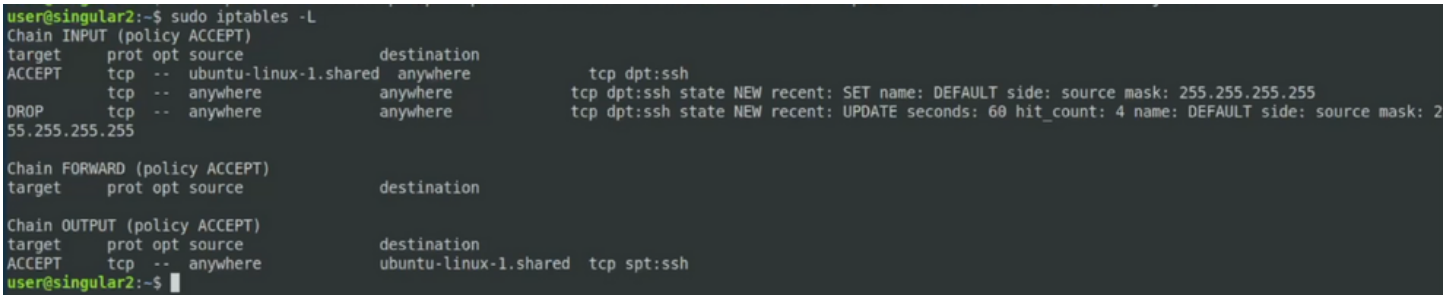
Vemos haciendo el -L que los DROP han desaparecido.

No es necesario eliminar la regla y ponerla de nuevo luego, también podemos moverla en una posición distinta según la jerarquia de la siguiente forma.

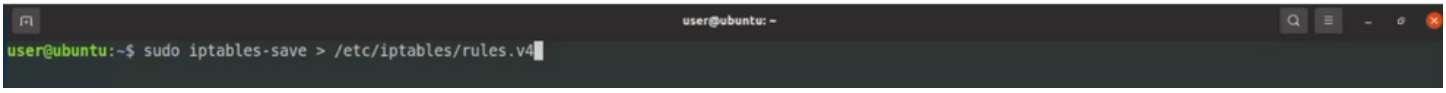
Ahora que no tenemos el drop, sólo tenemos el filtro de ssh, tenemos que añadir el filtro de 4 conexiones por minuto como maximo en el input sólo, porque el filtro es para el input. Se dice que si llega a mas de 4 conexiones en un minuto se haga DROP.



Listamos las reglas y vemos que se ha aplicado la regla en el input.



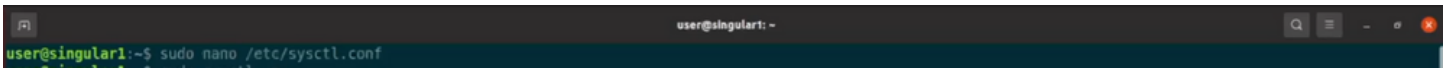
Almacenamos las reglas.

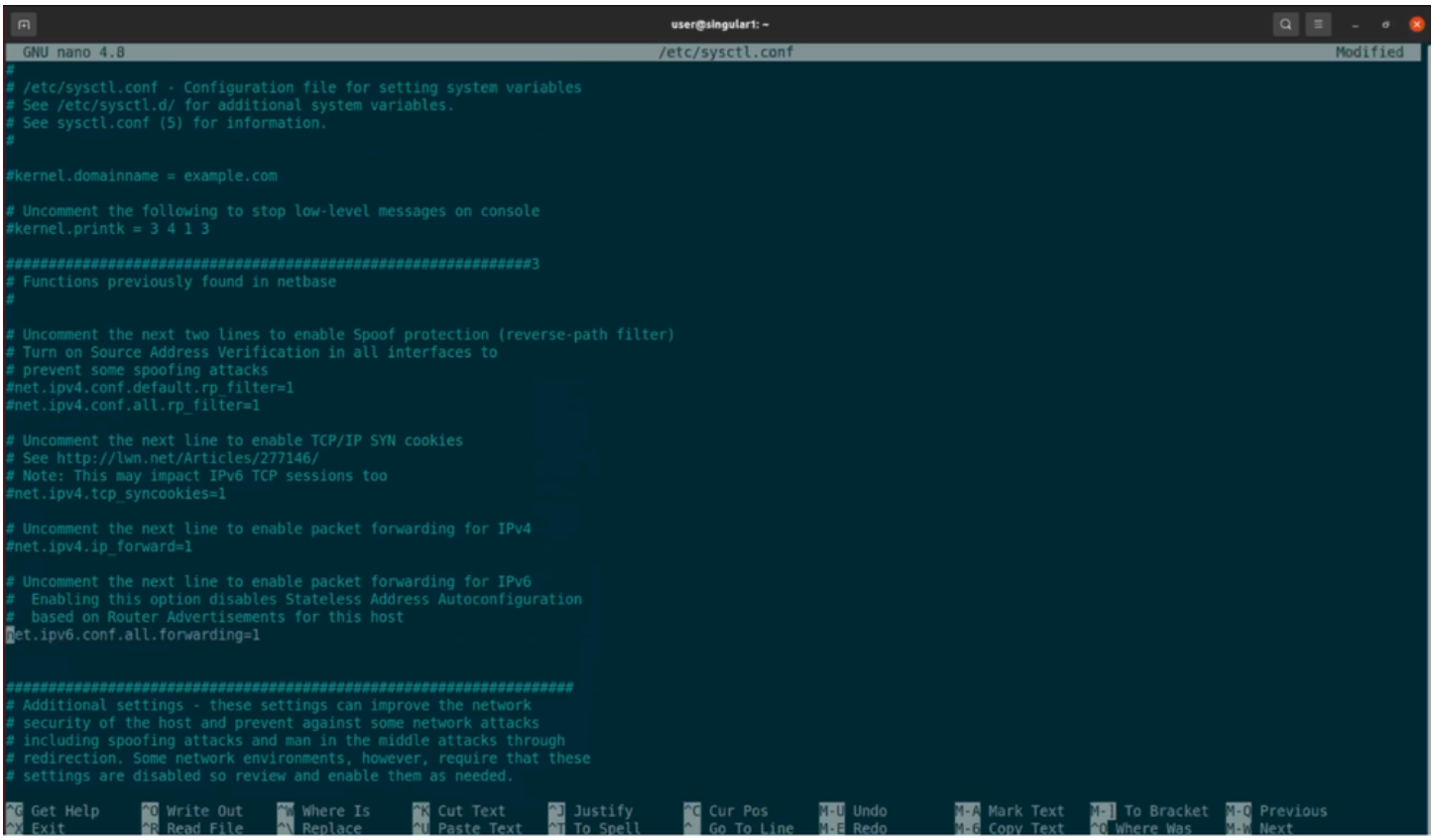


Si queremos restaurarlas hariamos ésto.

Ahora vamos a profundizar en los aspectos más complejos de IPTABLES, primero compartir la conexión a internet de una máquina a otra que no tiene conexión a internet.

Para habilitar el enrutamiento hay que descomentar la línea de éste archivo con sudo nano.





Aplicamos el cambio.

Ocultamos las direcciones ip internas haciendo parecer que todo el tráfico sale desde la máquina 1.

Ahora configuramos la maquina 2 para que utilice la maquina 1 cómo gateway.

Configuramos el archivo de configuración de la dns.

```
 GNU nano 4.8                                                /etc/resolv.conf
# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "resolvectl status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs must not access this file directly, but only through the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a different way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

nameserver 127.0.0.53
options edns0 trust-ad
search localdomain
```

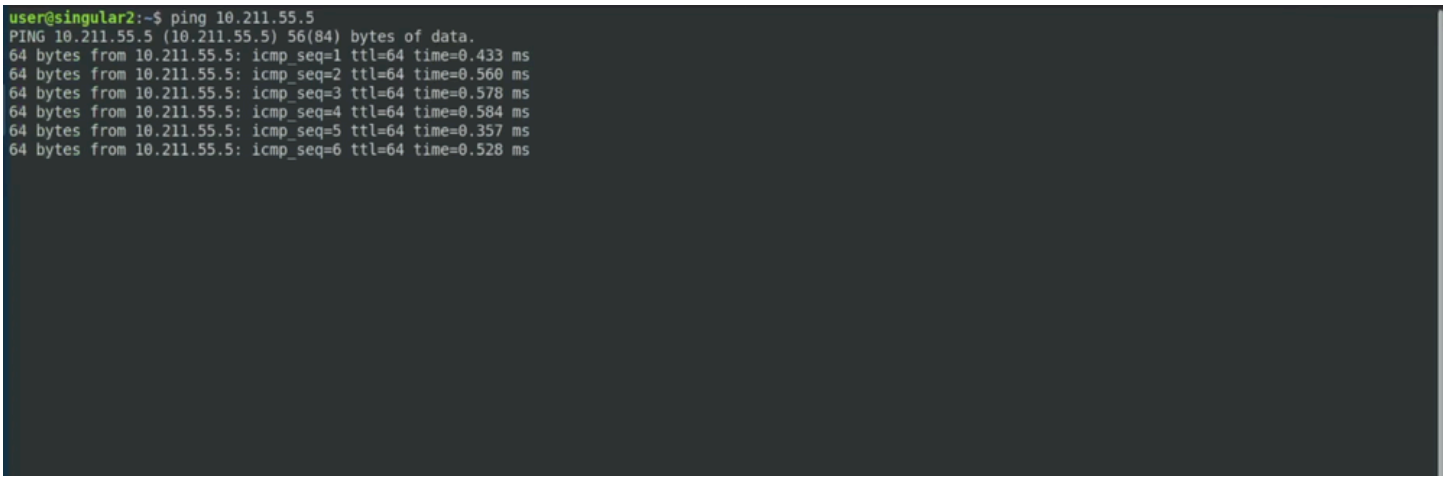
```
 GNU nano 4.8                                /etc/resolv.conf                        Modified
# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "resolvectl status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs must not access this file directly, but only through the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a different way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

nameserver 127.0.0.53
options edns0 trust-ad
search localdomain

nameserver 8.8.8.8
nameserver 8.8.4.4
```

Vemos con el ping que la maquina 2 ahora está conectandose a google usando la dirección ip de la maquina 1.

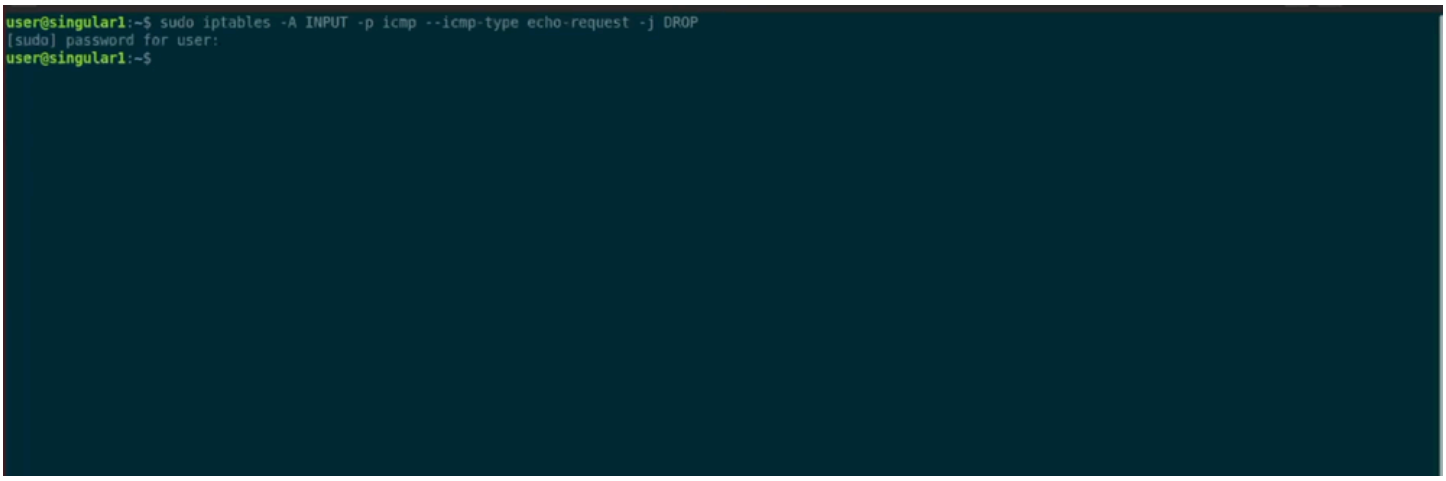
```
user@singular2:~$ ping www.google.es
PING www.google.es (216.58.215.163) 56(84) bytes of data.
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=1 ttl=128 time=9.52 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=2 ttl=128 time=10.1 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=3 ttl=128 time=9.54 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=4 ttl=128 time=9.21 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=5 ttl=128 time=9.49 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=6 ttl=128 time=9.59 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=7 ttl=128 time=10.00 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=8 ttl=128 time=9.48 ms
^C
--- www.google.es ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7026ms
rtt min/avg/max/mdev = 9.214/9.610/10.053/0.261 ms
user@singular2:~$
```

Ahora vamos a bloquear el ping y el icmp entre ambas maquinas para evitar que si alguien se infiltra que no tengas tanta información, icmp da mucha información sensible.

Primero verificamos si hay ping.

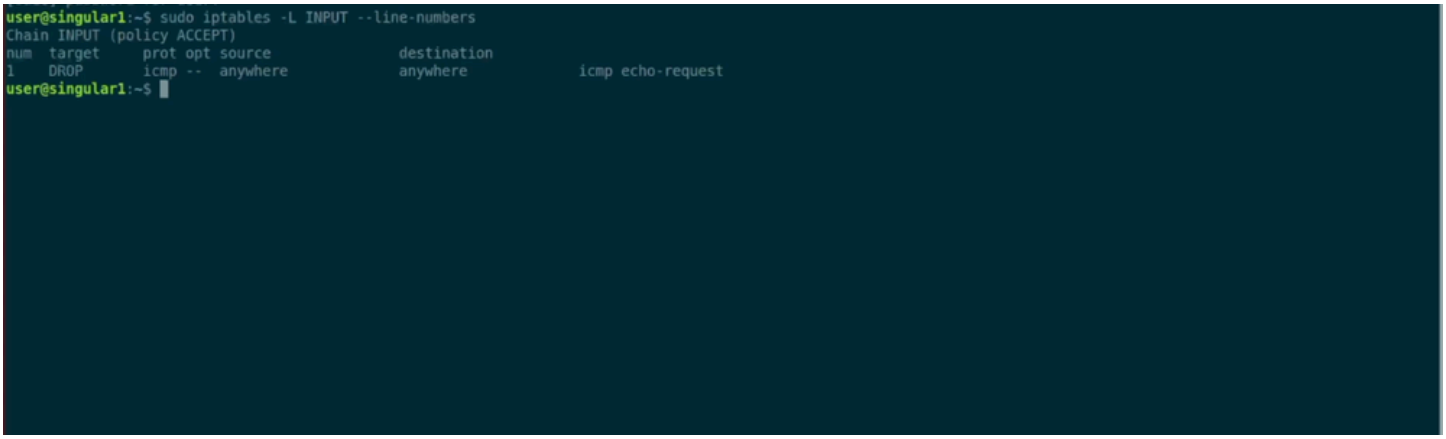


Bloqueamos el icmp o ping desde la maquina 1.



Ping bloqueado.

¿Cómo revertimos el comando usando las posiciones de las reglas, en lugar de los nombres? Aqui vemos el listado de reglas y su posición.



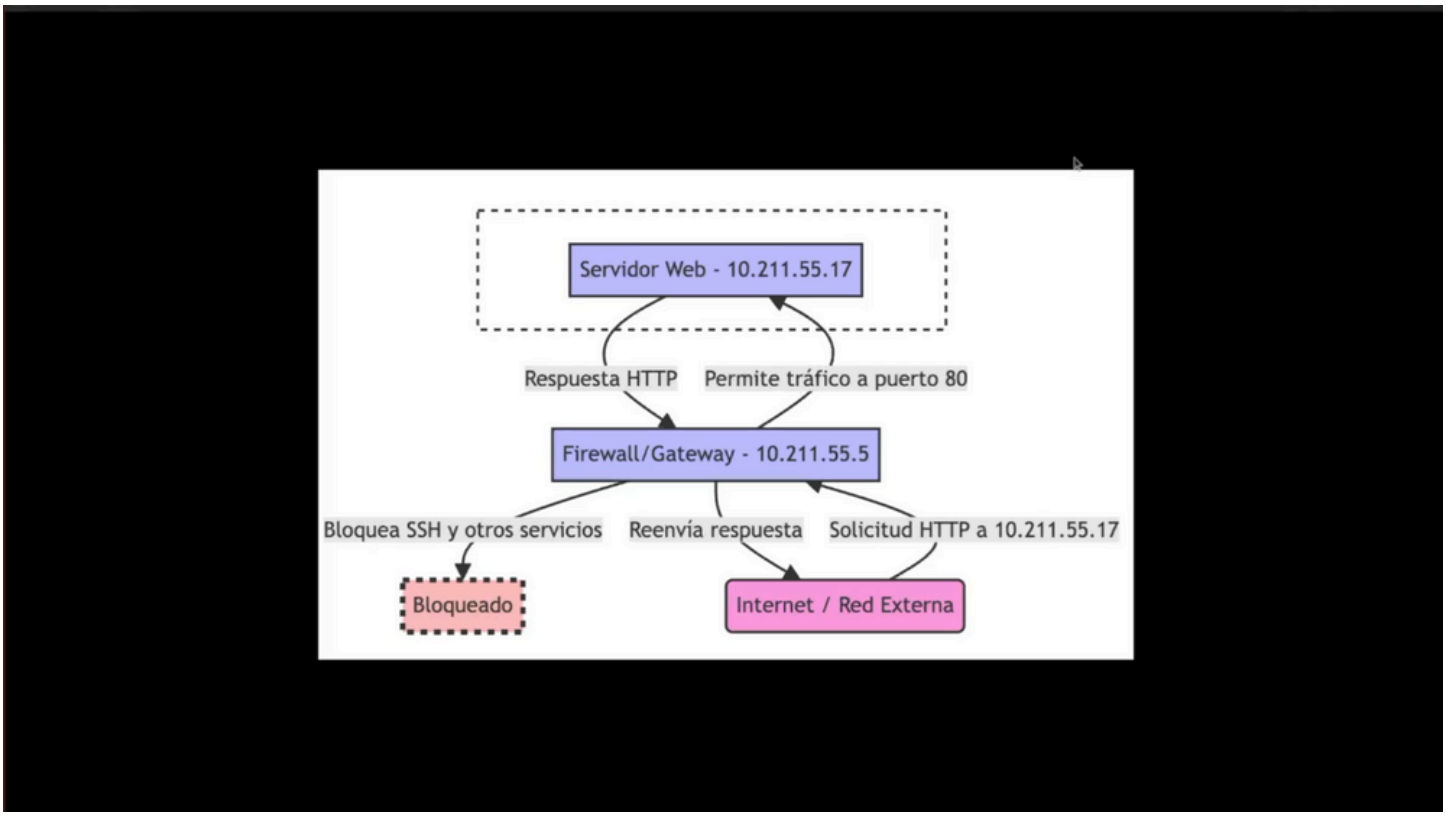
Comando para modificar la posición de la regla usando la posición. Vemos que se ha eliminado de la lista de reglas.

Ahora el ping funciona bien.

Ahora veremos la redirección de un puerto a otro con iptables, en este caso todo el trafico tcp desde un puerto externo que llegue a la maquina1 hacia la ip que vemos abajo en el puerto 80, que es un puerto interno (es lo que hace NAT).

Ahora vamos a crear una DMZ con iptables aplicando la reglas necesarias para que sea cómo el sigueinde diagrama.

Diagrama para entender cómo queremos que funcione la DMZ.



Configuramos una regla en la tabla nat para hacer un enmascaramiento (masquerade).

```
user@singular1:~$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Nueva regla: Permitimos el tráfico http hacia la maquina 2 (la que está en la dmz).

```
user@singular1:~$ sudo iptables -A FORWARD -p tcp --dport 80 -d 10.211.55.17 -j ACCEPT
user@singular1:~$
```

Nuevas reglas: Bloqueamos el tráfico hacia y desde la maquina 2, pero no incluirá el filtro anterior del tráfico http (será la excepción).

```
user@singular1:~$ sudo iptables -A FORWARD -p tcp --dport 80 -d 10.211.55.17 -j ACCEPT
user@singular1:~$ sudo iptables -A FORWARD -d 10.211.55.17 -j DROP
user@singular1:~$ sudo iptables -A FORWARD -d 10.211.55.17 -j DROP
user@singular1:~$ sudo iptables -A FORWARD -s 10.211.55.17 -j DROP
user@singular1:~$
```

Ahora hacemos una prueba.

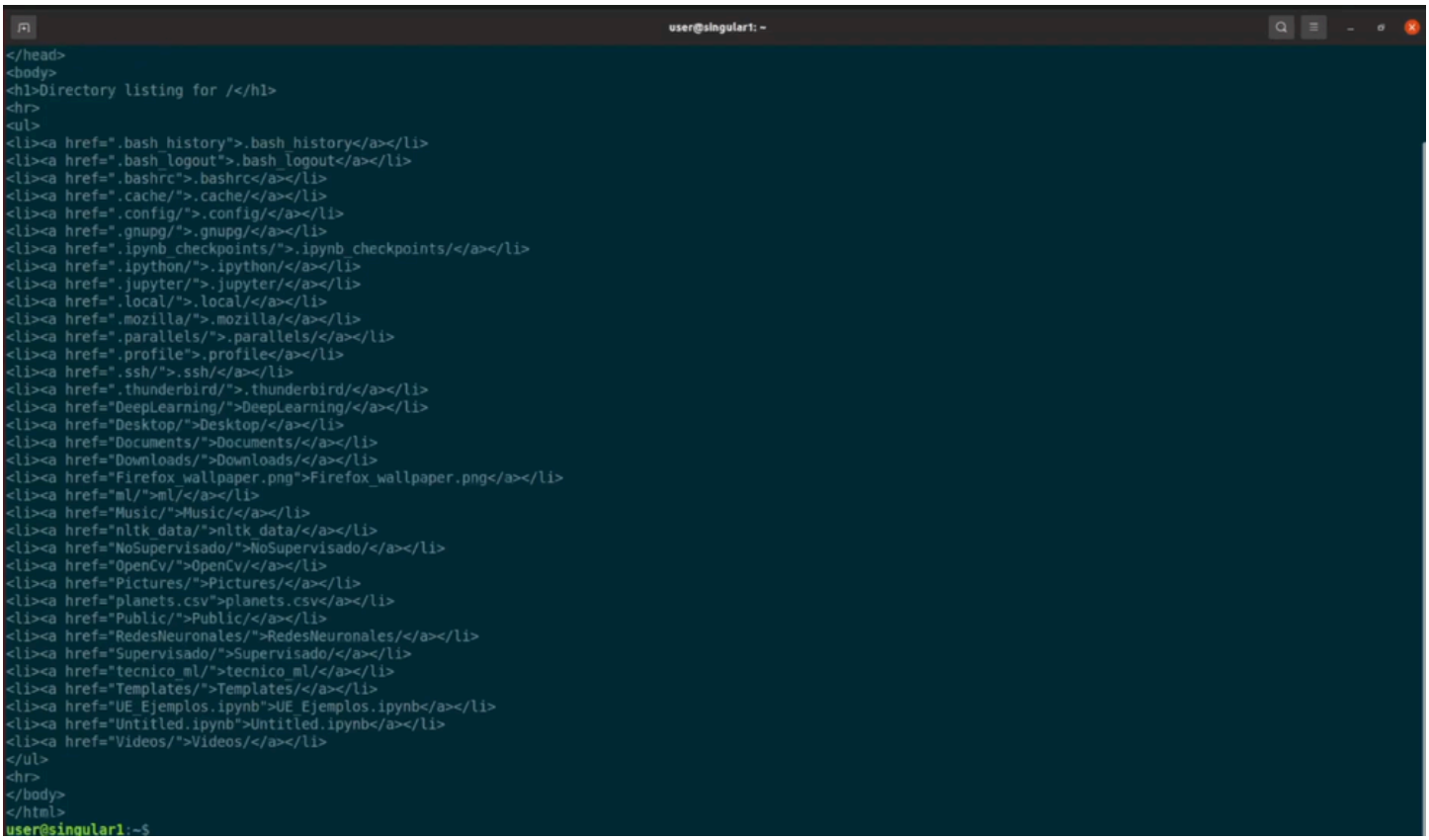
Primero levantamos un proceso en el puerto 80 desde la maquina 2. Para ello lo haremos usando python. Recordar que la maquina 2 esta dentro de la dmz y la hemos configurado para bloquear todo excepto http.

```
user@singular2: ~
user@singular2:~$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

en localhost:80 vemos los directorios que tenemos en esta carpeta.

Hacemos la prueba desde la maquina 1 de acceder a un directorio del puerto 80 (maquina 2) usando http, nos dará el listado porque http está permitido.

```
user@singular1: ~
user@singular1:~$ curl http://10.211.5
```



Si lo hacemos por ssh no nos deja acceder a éste directorio de la maquina 2 desde ésta máquina 1 porque hemos puesto un bloqueo total en la dmz excepto http.