# Teoría y práctica con Firewalls

## Centrados en iptables/nftables de Linux

www.darFe.es

Alejandro Corletti Estrada

## INDICE



## Presentación


Para profundizar en los conceptos teóricos sobre lo que es un firewall, te invitamos a que lo hagas en el punto 6.3. Firewalls de nuestro libro "**Seguridad por Niveles**"


También puedes verlo en el punto 6.2.1. Firewalls de nuestro libro "**Seguridad en Redes**"

Para no ser redundantes con lo escrito en estos libros, en esta artículo comenzaremos directamente con los conceptos básicos de **iptables**, dando por supuesto que ya has leído todo lo que nos indican los libros anteriormente presentados.

**iptables** es un programa que permite configurar el conjunto de reglas de filtro de paquetes proporcionadas por el núcleo Linux (implementado como diferentes módulos **Netfilter**).

**https://www.netfilter.org**

**Netfilter** (o Proyecto Netfilter) es una comunidad de desarrolladores de software e ingenieros, conocidos principalmente por el entorno disponible en el núcleo de **Linux** que permite interceptar y manipular paquetes de red.

Netfilter/iptables, como todo el núcleo Linux, es **software libre** (Open Source), distribuido bajo los términos de GNU GPLv2.

# 1. iptables

El concepto más importante para trabajar con **iptables** es que debemos tener en cuenta tres cosas:

**- tablas**

**- cadenas**

**- acciones**

**iptables** emplea un sistema de estructuración en tablas. (Se llama con "**-t**")

**Tablas**:

MANGLE (permite alterar parámetros de los paquetes)

NAT (se utiliza para la traducción de direcciones de red)

RAW (filtra los paquetes antes que cualquier otra tabla)

FILTER (es la tabla por defecto)

**Cadenas** (se llama, o las "Añade" con "**-A**")

Prerouting (Todo el tráfico de entrada, se utiliza para modificar paquetes antes de ser enrutados)

Postrouting (Todo el tráfico de salida, se utiliza para modificar paquetes después de ser enrutados)

Input (dirigido a la máquina antes de entrar a la misma)

Output (generado por la máquina, al salir de la misma)

Forward (todo tráfico que no sea generado por local, todo lo que pasa por la máquina)

**Acciones** (se llaman con "**-j**")

ACCEPT (acepta la trama)

DROP (Rechaza/descarta sin ICMP)

REJECT (Rechaza/descarta con ICMP)

QUEUE (envía a nivel Aplicación)

RETURN (deja de ser evaluado por la "cadena" correspondiente)

LOG (deja registro del evento)

| Tablas (-t) | Cadenas (-A) | Acciones (-j) |
|---|---|---|
| MANGLE | Prerouting | ACCEPT |
| NAT | Postrouting | DROP |
| RAW | Input | REJECT |
| FILTER | Output | QUEUE |
| | Forward | RETURN |
| | | LOG |

Comencemos de forma práctica nuestro trabajo.

Comenzaremos la práctica, trabajando siempre desde nuestra instalación de "**Kali**". Si aún no lo tienes instalado, será imprescindible que lo hagas. A continuación te dejamos el enlace hacia el video que lo explica y también a un documento "PDF" del que puedes estudiar el tema

Aprendiendo Ciberseguridad paso a paso - Charla 35: Kali - Instalación y conceptos básicos

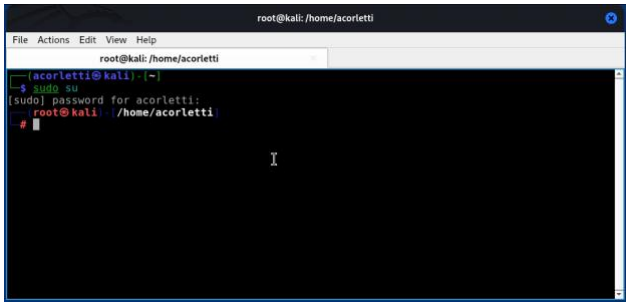

DOCUMENTOS

**Artículo: Instalación de Kali en VirtualBox (paso a paso)**

instalacion_de_Kali_en_VirtualBox.pdf

Como acabamos de indicar, todos los ejercicios de esta práctica los realizaremos desde **Kali**. Comenzaremos abriendo una interfaz de comandos, y a continuación es necesario escalar privilegios, pues justamente operaremos desde el mismo Shell (corazón) de Linux, por lo que es necesario ser un usuario con máximos privilegios.

Desde la interfaz de comandos ejecutaremos $ **sudo su** para convertirnos en súper usuario (root).




Con **iptables**, se recomienda SIEMPRE borrar todas las posibles reglas que tenga previamente creadas.

**# iptables -S** (Show, muestra el estado de iptables)

**# iptables -L** (List, lista el estado de iptables)

**# iptables -F** (Flush, si no se especifica una tabla específica, equivale a borrar todas las reglas una por una)

**# iptables -X** (Delete, borra cualquier cadena opcional especificada por el usuario)

**# iptables -Z** (Zero, pone a cero los paquetes y los contadores de byte de todas las cadenas)

## Ejercicio 1 :

**# iptables -S** (Show: nos muestra la configuración que actualmente tiene iptables

**# iptables -F** (Flush: borra toda la configuración previa)

## Ejercicio 2 :

Primero probamos desde nuestro host anfitrión, o desde donde lo prefieras, hacer "**ping**" a la dirección IP de **Kali**. Debería respondernos.

Recordemos que el formato de toda regla de **iptables** responde a la siguiente secuencia:

*iptables -t **tabla** -A **cadena** regla_con_parámetros -j **Acción***

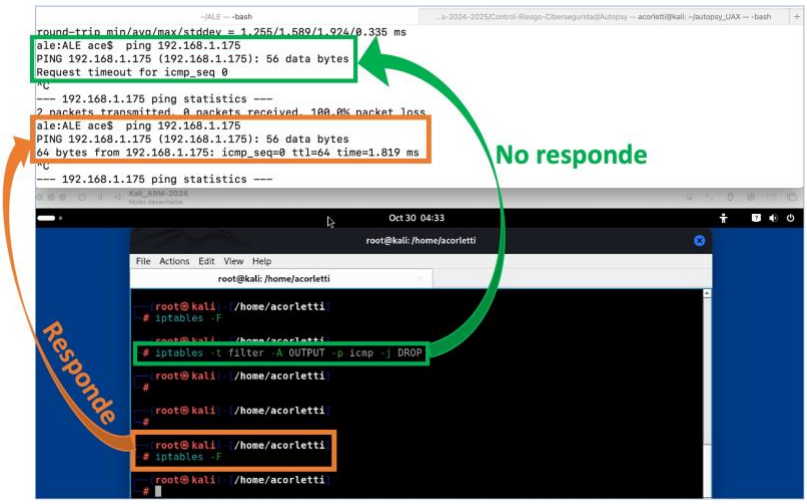
Con este formato probaremos nuestra primer regla, considerando que con la opción "**-p**" puedes definir de qué protocolo se trata, lo haremos escribiendo desde la interfaz de comandos:

**#iptables -t filter -A OUTPUT -p icmp -j DROP**

Le estamos diciendo a **iptables** que "filtre" (tabla filter), en la cadena de salida (OUTPUT) el protocolo "**icmp**", y que como acción, lo descarte (DROP)

Probamos nuevamente desde nuestro host anfitrión, o desde donde lo prefieras, hacer "**ping**" a la dirección IP de **Kali**. **NO** debería respondernos.

Para ver esta regla, puedes ejecutar el comando: **# iptables -L**




Si lo deseáis podéis, desde **Kali**, nuevamente ejecutar: **# iptables -F** para borrar esta nueva regla que acabáis de borrar y prueba nuevamente desde nuestro host anfitrión, o desde donde lo prefieras, hacer "**ping**" a la dirección IP de **Kali**. Debería volver a respondernos.

La tabla por defecto es "**filter**" por lo que si no la ponemos, la interpretará automáticamente. También debes tener en cuenta que el hardware, o software si es virtual, en el que operará **iptables**, puede, y en general debe, tener más de una interfaz de red, con lo cual puedes especificar la misma para que una regla aplique únicamente a esta interfaz.

Cuando tienes más de una interfaz, también es importante considerar que cierto tipo de tráfico puedes filtrarlo antes de ingresar a la interfaz, o al salir de la misma con la opción "**-i**" (input) "**-o**" (oputput), con lo que puedes probar los siguientes comandos:

```
#iptables -A INPUT -i eth0 -p icmp -j DROP
#iptables -A OUTPUT -o eth0 -p icmp -j DROP
```

Para ver las reglas que estás aplicando, tienes dos opciones:

```
#iptables -S
#iptables -L
```


## Ejercicio 3:

En general, el protocolo **icmp**, suele ser muy útil para lo que en redes se denomina "troubleshoutting", que en español sería "problemas" y, justamente, sirve para detectar incidentes (problemas) de comunicación. A través del comando "ping", se puede determinar rápidamente si hay o no comunicación con un recurso, por lo que, si bien el resto de los "tipos" y códigos" del protocolo **icmp** pueden ser brechas de seguridad, el "**ping**" es una buena práctica dejarlo activo, al menos en los segmentos de red internos o donde deba realizarse gestión de red y sistemas.

Recomendamos que veáis los siguientes videos si deseáis profundizar en este tema:

- Charla 57: Protocolo ICMP - Presentación - Aprendiendo Ciberseguridad paso a paso


- Protocolo ICMP - Práctica túnel ICMP - Charla 58 - Aprendiendo Ciberseguridad paso a paso


Por lo tanto, si queremos se más específicos con este protocolo, podemos ajustar más aún esta regla anterior, con las reglas que presentamos a continuación:

```
#iptables -A OUTPUT -o eth0 -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
#iptables -A INPUT -i eth0 -p icmp -m icmp --icmp-type echo-request -j ACCEPT
```

**#iptables -A INPUT -i eth0 -p icmp -j DROP**

Para verificar su correcto funcionamiento, recordemos el empleo del comando "hping3" que vimos en el video:

ejercicio de fragmentación IP empleando HPING3 y Wireshark

A través de este comando y capturando tráfico con **Wireshark**, podremos verificar , que si desde nuestro host anfitrión, o desde cualquier otro, una vez activadas estas reglas, hacemos "**ping**" (icmp tipo 8 y 0) a la IP de **Kali**, nos responderá y **Wireshark** capturará este tráfico, pero si desde nuestro host anfitrión, o desde cualquier otro ejecutamos por ejemplo:

**#hping3 192.168.1.177 --icmptype 13** (*siendo **192.168.1.177** en nuestro caso la IP de Kali*)

Este tráfico no será respondido, pues con la excepción del "ping" todo lo demás de icmp está con la acción: **DROP**.

Por último, si borramos todas las reglas: **#iptables -F**

Y volvemos a ejecutar desde el anfitrión, o desde cualquier otro host:

**#hping3 192.168.1.177 --icmptype 13**

Con **Wireshark**, veremos que este tráfico ahora sí será respondido, como en nuestro caso se puede apreciar en la imagen de abajo.

| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 2024-10-31 13:00:03.359231 | 192.168.1.177 | 192.168.1.175 | ICMP | 54 | Timestamp reply id=0x095a, seq=0/0, ttl=64 |
| 2 | 2024-10-31 13:00:04.367918 | 192.168.1.177 | 192.168.1.175 | ICMP | 54 | Timestamp reply id=0x095a, seq=256/1, ttl=64 |
| 3 | 2024-10-31 13:00:05.366380 | 192.168.1.177 | 192.168.1.175 | ICMP | 54 | Timestamp reply id=0x095a, seq=512/2, ttl=64 |
| 4 | 2024-10-31 13:00:06.370230 | 192.168.1.177 | 192.168.1.175 | ICMP | 54 | Timestamp reply id=0x095a, seq=768/3, ttl=64 |
| 5 | 2024-10-31 13:00:07.374627 | 192.168.1.177 | 192.168.1.175 | ICMP | 54 | Timestamp reply id=0x095a, seq=1024/4, ttl=64 |
| 6 | 2024-10-31 13:00:08.376264 | 192.168.1.177 | 192.168.1.175 | ICMP | 54 | Timestamp reply id=0x095a, seq=1280/5, ttl=64 |

## Ejercicio 4:

Supongamos que deseamos rechazar los segmentos TCP que entran con destino a un servidor Web, el comando sería:

**#iptables -t filter -A INPUT -p tcp --dport 80 -j DROP**

Fijaros que ahora estamos implementando una opción nueva que es "**-dport**" para seleccionar el puerto de nivel de transporte que más nos guste. En este caso, como hemos seleccionado el protocolo TCP (**-p tcp**) se tratará del puerto TCP 80.

Como la tabla por defecto es "filter", sería igual poner:

**#iptables -p tcp --dport 80 -j DROP**

¿Qué cambiarías a esta regla para que en vez de rechazar estos paquetes, los deje pasar?

Supongamos que en esta misma regla deseamos rechazar el puerto fuente TCP 80, y también el puerto destino TCP 80 ¿Cómo lo harías?

¿Si en vez de poner el número “80” pones “http”, lo aceptará?

## Ejercicio 5:

Vamos a mejorar nuestra práctica empleando otra herramienta que también trae embebida, o integrada, esta maravilla que es “**Kali**”.

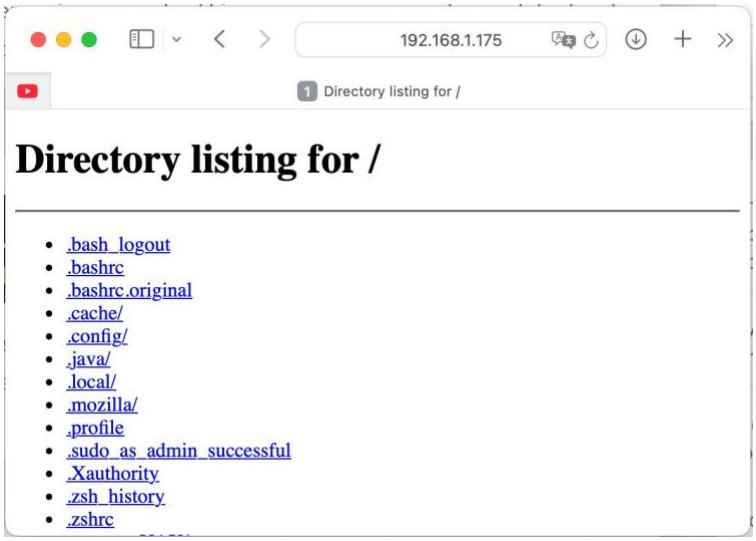
Uno de los tantos lenguajes de programación que existen en el universo de la informática es “**Python**”. Dentro da los millones de posibilidades que ofrece Python y que también está pre-configurado en nuestro fabuloso e insuperable “**kali**”, es un sencillo servidor Web que nos puede ser de muchísima utilidad, por ejemplo para compartir ficheros rápida y sencillamente. Se trata de “**http.server**” y desde nuestra interfaz de comandos podemos ejecutarlo con la siguiente instrucción:

```
#python3 -m http.server 80
```

Fijaros que si aún sigue en vigor la regla que hemos creado en el ejercicio anterior e intentaremos conectarnos a este servidor Web en el puerto 80, no nos lo permitiría, pero si ejecutáramos:

```
#iptables -F
```

Y nuevamente abrimos un navegador y colocamos: http://IP de Kali se nos presentaría un servidor Web similar al que se presenta en la imagen de la derecha.




Si en vez de abrir este servidor Web en el puerto 80, deseáramos hacerlo en otro, pues podemos volver a ejecutar: **#python3 -m http.server 9000** y lo tendríamos ahora ejecutando en el puerto 9000.

Si quisiéramos filtrar el acceso ahora a ese nuevo servidor, la regla sería:

```
#python3 -m http.server 9000
```

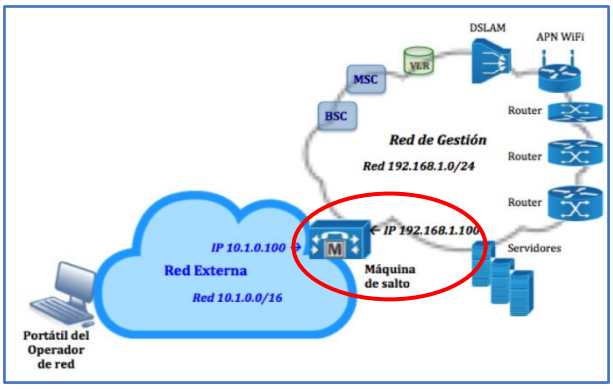
## Ejercicio 6:

Cuando gestionamos los recursos de nuestras redes, en general accedemos a los mismos por protocolo **SSH** (Secure Shell), recordad el NO uso de “telnet”. Un detalle muy importante es el concepto de “**máquina de salto**”, es decir un host muy bien bastionado que por medio de dos tarjetas de red, puede ser la verdadera puerta de entrada de la “**red de gestión**”, es decir una red exclusiva para poder acceder a los recursos de forma segura y con permisos de administración de los mismos.

La idea de esta "**máquina de salto**" es, justamente, poder segmentar o segregar esta red (gestión) del resto del planeta, y de todas las demás redes de mi organización. La máquina de salto, es la puerta de entrada a esta red y, si hacemos las cosas bien, debería ser la **única** puerta de entrada.

Es decir, si alguien intenta hacer SSH desde un equipo de la empresa hacia cualquier otro, jamás debería ver abierto este puerto (TCP 22 = SSH), y le resultaría imposible realizar ningún tipo de conexión SSH desde cualquiera de los recursos de la red, con excepción de la dirección IP de la máquina de salto.

En resumen, la máquina de salto tiene una tarjeta de red visible desde dentro de mi empresa, o desde una conexión segura a través de Internet (para poder ingresar a la misma, en la imagen de la derecha sería la IP ***10.1.0.100***), y otra tarjeta de red conectada a la red de gestión, la cual alcanza todos los dispositivos y servidores de la organización a través del puerto TCP 22 (para poder "saltar" desde allí a cualquier otro recurso por medio del protocolo SSH, en la imagen sería la IP ***192.168.1.100***), tal cual podemos ver ambas en la imagen de la derecha.




Este ejercicio, consiste en diseñar una regla de **iptables**, que se aplicará a todos los recursos de mi organización, para que únicamente permita el acceso SSH desde la dirección IP de la máquina de salto.

Supongamos que la dirección IP de la máquina de salto es: 192.168.1.100 (como se muestra en la imagen). Las reglas a aplicar en TODOS los recursos de mi red sería:

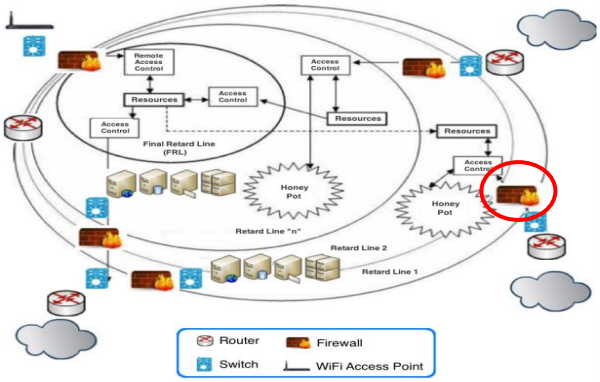
```
#iptables -A INPUT -s 192.168.1.100 -p TCP --dport 22 -j ACCEPT
# iptables -A INPUT -p TCP --dport 22 -j DROP
```

Una vez aplicada esta regla, probaremos la conexión desde dos ordenadores diferentes, uno con la dirección IP que figura en la regla (en nuestro caso 192.168.1.100) el cuál **sí** nos dejará conectar, y el segundo con cualquier otra dirección IP, el cual **NO** nos dejará conectar.

La aplicación de esta regla en todos los recursos, evita lo que se conoce como "salto lateral", es decir que desde un recurso, pueda saltar libremente a otro, sin respetar la jerarquía de "**control de accesos**" que debe tener una red de gestión.

**Ejercicio 7:** Proteger toda nuestra red privada.


En el punto 2.2. Planificación de la Seguridad, de nuestro libro "**Seguridad en Redes**" (y también en varios videos y ponencias), presentamos la imagen que podemos ver a la derecha, en la que concretamente se refiera a la "**defensa en profundidad**"



pero para este ejercicio, nos centraremos solamente en uno de sus firewalls. Concretamente, supongamos que queremos definir las reglas para el que resaltamos, en la imagen anterior, con el **círculo rojo**.

Este firewall, por ejemplo, puede ser el que regula la **salida** de los empleados hacia Internet. Vemos que se encuentra en la zona más externa de la red (**DMZ**), y que no está protegiendo ningún recurso específico (*como podrían ser los servidores que se ven debajo de la figura o en su centro, o el punto de acceso **WiFi** de la parte superior izquierda, etc.*).

Comenzando desde las reglas más simples, con las que figuran a continuación, permitirías que a través de este firewall, todos los usuarios de la organización puedan navegar por internet, pero desde el exterior, nadie podría ingresar a tu empresa.

Este firewall ahora analizará los paquetes que pasan a través de él, por lo que nuestra recomendación es comenzar a emplear la cadena "**FORWARD**". En nuestro ejemplo, esta firewall tiene la interfaz "**eth0**" que se conecta a la organización y la interfaz "**eth1**" que es la salida hacia Internet. Estas reglas quedarían:

```
#iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
#iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 443 -j ACCEPT
#iptables -A FORWARD -i eth1 -o eth0 -j DROP
```

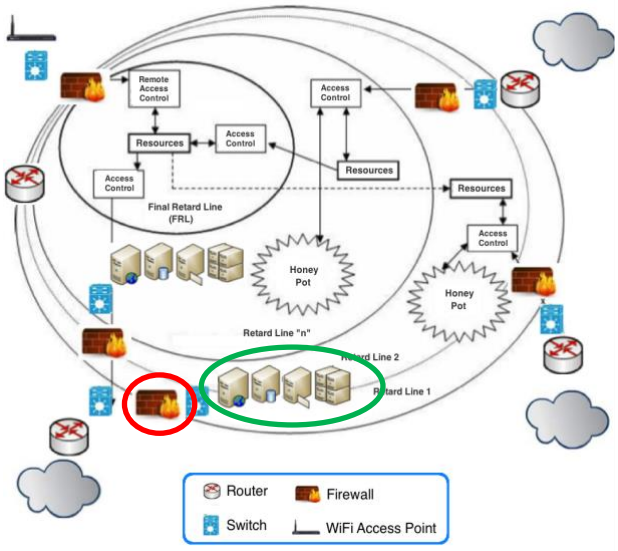
También podríamos haber puesto las dos primeras reglas como una sola:

```
#iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80, 443 -j ACCEPT
```

## Ejercicio 8:

Sobre el mismo diagrama de red anterior, supongamos ahora que nuestra empresa tiene los servidores que hemos remarcado con el **óvalo verde** (en la imagen de abajo) y tienen expuestas las direcciones 200.200.200.1, 200.200.200.2 y 200.200.200.3, ofreciendo los siguientes servicios a sus clientes que acceden desde Internet:

- Servidor de correo electrónico (200.200.200.1, puertos 25 y 110)
- Servidor Web (200.200.200.2, puerto 443)
- Servidor de ficheros (200.200.200.3, puerto 22 para sftp o scp)




El firewall sobre el que aplicaremos las reglas será el que hemos remarcado con el **círculo rojo**.

*Antes de pasar a la página siguiente ¿te atreves a definir tu solo las reglas?*

Como ya has pasado de página, aquí abajo te dejamos una propuesta de estas reglas:

```
#iptables -A FORWARD -i eth1 -o eth0 -d 200.200.200.1 -p tcp --dport 25, 110  -j ACCEPT
#iptables -A FORWARD -i eth1 -o eth0 -d 200.200.200.1 -p tcp --dport 25, 110 -j ACCEPT
#iptables -A FORWARD -i eth1 -o eth0 -d 200.200.200.1 -p tcp --dport 25, 110 -j ACCEPT
#iptables -A FORWARD -i eth1 -o eth0 -j DROP
```

## Ejercicio 9:

Si recordamos el tema del control de sesiones del protocolo TCP y queremos mejorar la seguridad de estos accesos, para evitar ataques relacionados con el control de estados.

Para refrescar estos conceptos, os recomendamos que miréis el video:


- El nivel de Transporte (Segmentación TCP) - Charla 71 - Aprendiendo Ciberseguridad paso a paso

  (*y también los anteriores de TCP, a partir de la charla 64*)

Para que estas reglas, permitan también realizar el control de estados, deberían ser como sigue a continuación:

```
#iptables -A FORWARD -i eth1 -o eth0 -d 200.200.200.1 -p tcp --dport 25, 110  -m state --state NEW,ESTABLISHED -j ACCEPT
#iptables -A FORWARD -i eth1 -o eth0 -d 200.200.200.1 -p tcp --dport 25, 110 -m state --state NEW,ESTABLISHED -j ACCEPT
#iptables -A FORWARD -i eth1 -o eth0 -d 200.200.200.1 -p tcp --dport 25, 110 -m state --state NEW,ESTABLISHED -j ACCEPT
#iptables -A FORWARD -i eth1 -o eth0 -j DROP
```

## Ejercicio 10:

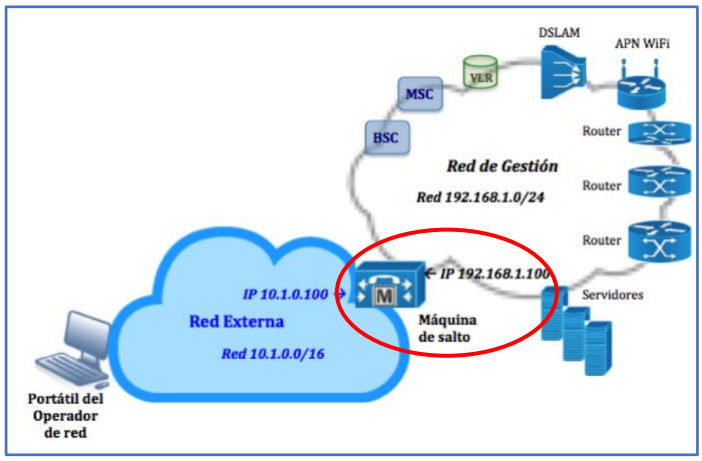
Como el control de estados, puede congestionar o recargar considerablemente el rendimiento de un firewall, o inclusive ser aprovechado maliciosamente para llegar a este tipo de situaciones, siempre es conveniente restringir este tipo de reglas, tal cual se presentan a continuación.

```
#iptables -A FORWARD -i eth1 -o eth0 -d 200.200.200.1 -p tcp --dport 25, 110  -m state --state NEW,ESTABLISHED -j ACCEPT -m connlimit --connlimit-above 15 -j REJECT --reject-with tcp-reset
#iptables -A FORWARD -i eth1 -o eth0 -d 200.200.200.1 -p tcp --dport 25, 110 -m state --state NEW,ESTABLISHED -j ACCEPT -m connlimit --connlimit-above 15 -j REJECT --reject-with tcp-reset
#iptables -A FORWARD -i eth1 -o eth0 -d 200.200.200.1 -p tcp --dport 25, 110 -m state --state NEW,ESTABLISHED -j ACCEPT -m connlimit --connlimit-above 15 -j REJECT --reject-with tcp-reset
#iptables -A FORWARD -i eth1 -o eth0 -j DROP
```

¿Te atreves a aplicar también un control de estados, pero basado en tiempos de conexión?

Investiga y practica con las siguientes opciones: **-m time --timestart**

**Ejercicio 11:** Empleo de "Log"




Volvamos a nuestro ejemplo de la **máquina de salto**. En este tipo de situaciones, y por supuesto en muchísimas otras, nos interesaría que nos quede registrado si alguien está intentando acceder a nuestros dispositivos por medio de SSH. En estas situaciones **iptables** nos ofrece la posibilidad de "loguear" (registrar) estos eventos por medio de la acción "LOG".

Recordemos que la dirección IP de la máquina de salto de nuestro ejemplo era: 192.168.1.100. Las reglas a aplicar en TODOS los recursos de mi red, si deseo que las mismas generen registros (Logs) en caso de accesos serían:

```
#iptables -A INPUT -s 192.168.1.100 -p TCP --dport 22 -j ACCEPT
#iptables -A INPUT -p tcp -m tcp --dport 22 -j LOG --log-prefix 'acceso SSH '
# iptables -A INPUT -p TCP --dport 22 -j DROP
```

Para profundizar sobre el sistema de "**Syslog**" de Linux, os recomendamos que le deis una mirada a nuestro video:

Syslog (ejercicio, teoría y práctica)


**Ejercicio 12:** Guardar y restaurar reglas

**Iptables** posee parámetros para almacenar las reglas que hemos creado, y también restaurarlas en el momento en que lo deseemos. Tengamos en cuenta que hasta ahora solo hemos creado reglas "volátiles", es decir que si se apaga el ordenador, las mismas se perderán y si se vuelve a encender estas reglas no existirán.

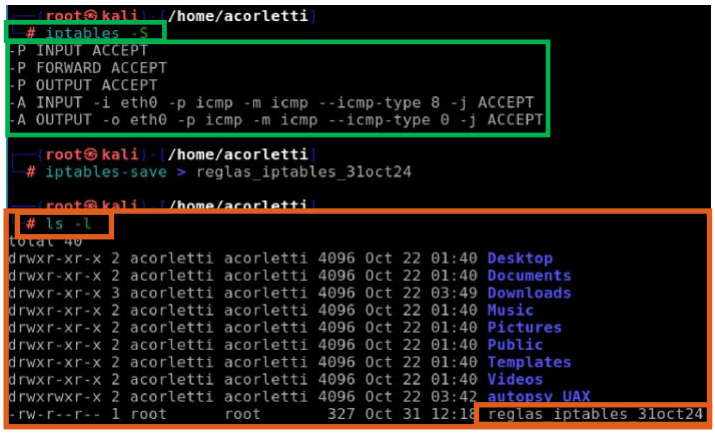
Más adelante veremos cómo hacerlas permanentes, pero en el ejercicio actual, solamente nos interesa que conozcas las opciones que nos ofrece **iptables** para el almacenamiento y recuperación de los trabajos que estamos realizando.

Las opciones son:

**iptables-save** e

### iptables-restore

Tomemos cualquiera de los ejercicios que hemos realizado hasta ahora, en nuestro caso, como podéis ver en la imagen de abajo, con el comando **iptables -S** nos muestra la configuración actual que hemos **resaltado en verde**.




A continuación ejecutamos:

**#iptables-save > reglas_iptables_31oct24**

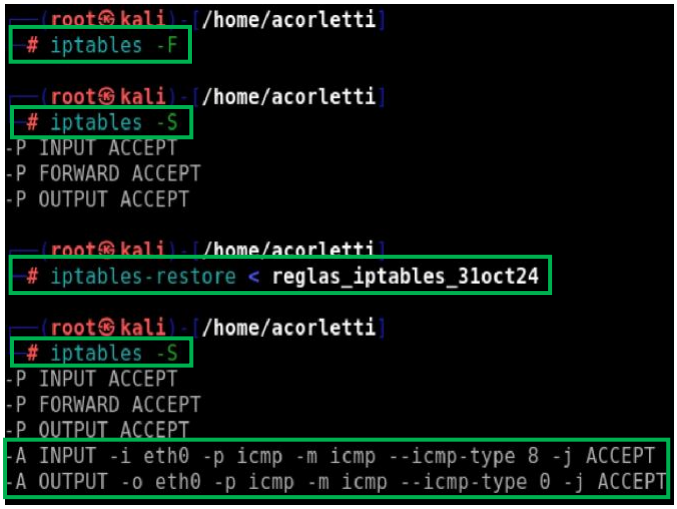
Hemos guardado estas reglas, tal cual se muestra en la imagen al ejecutar "**ls -l**" y hemos resaltado en **color naranja**.

Para verificar su recuperación, vamos a borrar todas las reglas con el comando **iptables -F**, verificamos que hayan sido borradas con el comando **iptables -S** y a continuación procedemos a restaurarlas con el comando:

**#iptables-restore > reglas_iptables_31oct24**

Si ahora ejecutamos nuevamente **iptables -S** veremos que las reglas están aplicadas nuevamente.

Estos pasos los hemos resaltado en verde en la imagen de la derecha.



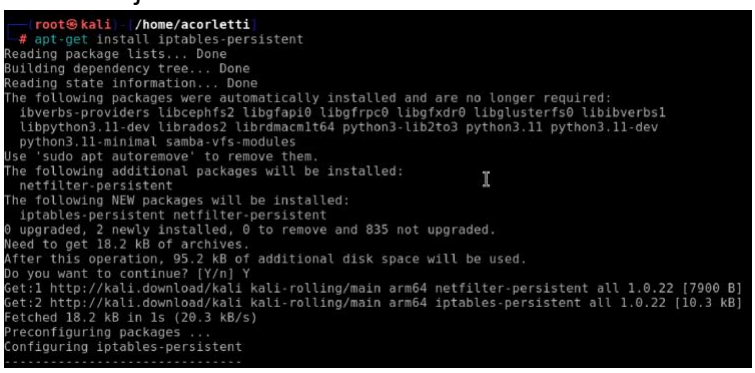

## Ejercicio 13:

Otra herramienta muy útil para guardar y restaurar configuraciones es "iptables-persistent", te invitamos a que la emplees, y en este ejercicio trabajaremos con ella.

Para instalarla en nuestro "**Kali**" es bastante sencillo, solo debes ejecutar:

**# apt-get install iptables-persistent**

Como puedes ver en la imagen de la derecho solo te pedirá si deseas continuar , ante lo que respondes "**Y**".



```
Save current IPv4 rules? [yes/no] yes


Current iptables rules can be saved to the configuration file /etc/iptables/rules.v6.
will then be loaded automatically during system startup.

Rules are only saved automatically during package installation. See the manual page of
ip6tables-save(8) for instructions on keeping the rules file up-to-date.

Save current IPv6 rules? [yes/no] yes


Selecting previously unselected package netfilter-persistent.
(Reading database ... 384482 files and directories currently installed.)
Preparing to unpack .../netfilter-persistent_1.0.22_all.deb ...
Unpacking netfilter-persistent (1.0.22) ...
Selecting previously unselected package iptables-persistent.
Preparing to unpack .../iptables-persistent_1.0.22_all.deb ...
Unpacking iptables-persistent (1.0.22) ...
Setting up netfilter-persistent (1.0.22) ...
update-rc.d: We have no instructions for the netfilter-persistent init script.
update-rc.d: It looks like a non-network service, we enable it.
netfilter-persistent.service is a disabled or a static unit, not starting it.
Setting up iptables-persistent (1.0.22) ...
Processing triggers for man-db (2.12.1-2) ...
┌──(root㉿kali)-[/home/acorletti]
└─#
```

Durante la instalación, te preguntará también si deseas guardar las reglas que actualmente tienes en tu **iptables**. Lo hará tanto para IP versión 4 como para IP versión 6 (aunque en nuestro caso no las tengamos).

Si respondes "yes" automáticamente te guardará las reglas que tengas configuradas en el directorio "/etc/iptables" con los nombres: "**rules.v4**" y "**rules.v6**". Como puedes ver en la imagen de abajo, ambos son los ficheros en texto plano de las configuraciones.

```
┌──(root㉿kali)-[/home/acorletti]
└─# ls -l /etc/iptables
total 4
-rw-r--r-- 1 root root 327 Nov  2 12:36 rules.v4
-rw-r--r-- 1 root root   0 Nov  2 12:36 rules.v6
```

Si queremos visualizar nuestro fichero con el comando "**cat**" podemos hacerlo, tal cual se muestra en la siguiente imagen.

```
┌──(root㉿kali)-[/home/acorletti]
└─# cat /etc/iptables/rules.v4
# Generated by iptables-save v1.8.10 (nf_tables) on Sat Nov  2 12:36:52 2024
*filter
:INPUT ACCEPT [6720:1928683]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [1819:187794]
-A INPUT -i eth0 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 0 -j ACCEPT
COMMIT
# Completed on Sat Nov  2 12:36:52 2024
```

Para verificar su funcionamiento, a continuación crearemos una nueva regla.

```
┌──(root㉿kali)-[/home/acorletti]
└─# iptables -A INPUT -s 192.168.1.100 -p TCP --dport 22 -j ACCEPT

┌──(root㉿kali)-[/home/acorletti]
└─# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -s 192.168.1.100/32 -p tcp -m tcp --dport 22 -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 0 -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 8 -j ACCEPT
```

Para guardar estas nuevas reglas, con la nueva que acabamos de crear, el comando es:

```
#netfilter-persistent save
```

En la imagen que sigue podemos ver la ejecución de este comando y en la misma, unas líneas más abajo, el "**cat**" para visualizar el nuevo fichero "**rules.v4**" que se ha creado

```
┌──(root㉿kali)-[/home/acorletti]
└─# netfilter-persistent save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save

┌──(root㉿kali)-[/home/acorletti]
└─# cat /etc/iptables/rules.v4
# Generated by iptables-save v1.8.10 (nf_tables) on Sat Nov  2 12:46:47 2024
*filter
:INPUT ACCEPT [38:1809]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [1:72]
-A INPUT -s 192.168.1.100/32 -p tcp -m tcp --dport 22 -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 0 -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 8 -j ACCEPT
COMMIT
# Completed on Sat Nov  2 12:46:47 2024
```

El paso siguiente, será borrar nuevamente todas las reglas, mostrar cómo ha quedado **iptables** y finalmente reiniciar el sistema con el comando: **#reboot**

```
┌──(root㉿kali)-[/home/acorletti]
└─# cat /etc/iptables/rules.v4
# Generated by iptables-save v1.8.10 (nf_tables) on Sat Nov  2 12:50:57 2024
*filter
:INPUT ACCEPT [69:3304]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [1:72]
-A INPUT -s 192.168.1.100/32 -p tcp -m tcp --dport 22 -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 0 -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 8 -j ACCEPT
COMMIT
# Completed on Sat Nov  2 12:50:57 2024

┌──(root㉿kali)-[/home/acorletti]
└─# iptables -F

┌──(root㉿kali)-[/home/acorletti]
└─# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT

┌──(root㉿kali)-[/home/acorletti]
└─# reboot
```

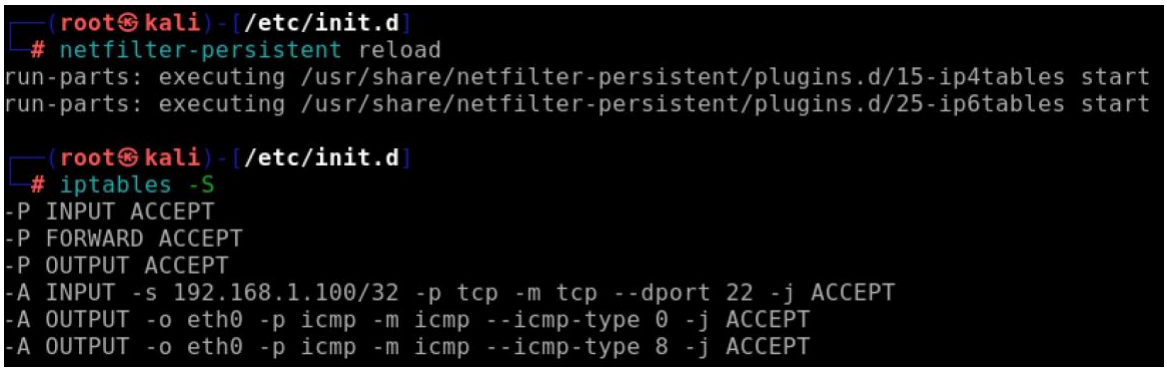
Una vez reiniciado, si miramos nuevamente cómo se encuentra **iptables**, podemos verificar que está vacío, como se ve en la imagen de la derecha.

```
┌──(acorletti㉿kali)-[~]
└─$ sudo su
[sudo] password for acorletti:
┌──(root㉿kali)-[/home/acorletti]
└─# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

Para retornar a nuestra última configuración guardada el comando será:

```
#netfilter-persistent reload
```

Como podemos ver en la imagen siguiente, hemos recuperado la configuración de nuestro firewall.




Como conclusión final, hemos logrado guardar y rescatar nuestras configuraciones de **iptables**, pero aún así, acabamos de comprobar que si el sistema se reinicia, aspecto común en cualquier recurso informático, al arrancar nuevamente **iptables** lo hace desde cero. Es decir, se han perdido los cambios y debemos recuperarlos manualmente… esto no nos termina de convencer, pues si nosotros instalamos un firewall, justamente debemos confiar en que las reglas que hemos configurado sean permanentes, sucediera lo que sucediese… ¿podremos hacerlo?...

## Ejercicio 14:

En el ejercicio anterior, hemos visto e instalado la herramienta "**iptables-persistent**" para ayudarnos a guardar y recuperar nuestras configuraciones de **iptables**.

Hemos dedicado varias horas de trabajo con la misma para intentar que la configuración se recupere al reiniciar nuestros sistemas. Se supone que con solo instalarla e iniciarla, a medida que vamos guardando (save) nuestras configuraciones, automáticamente queda guardado para cualquier reinicio del sistema. Siendo totalmente sincero con vosotros, no sabemos por qué razón en Ubuntu esto no funciona de manera confiable. Es decir, al reiniciar no recupera el fichero de configuración de **iptables**.

Para que esto no genere ningún tipo de duda, hemos recurrido a la vieja solución de hacerlo por medio de un sencillo script que se aplicará a las interfaces de red al arranque del sistema operativo, por lo que os invitamos a que practiquéis los siguientes pasos.

Tal cual se ejecutó en el ejercicio anterior cuando ejecutamos el comando:

```
#netfilter-persistent save
```

Se almacena la configuración actual en **/etc/iptables/rules.v4**.

Nuestra recomendación es que a medida que realizamos cambios SIEMPRE vayáis haciendo copias temporales del fichero de **iptables**, esto podemos ejecutarlo, como ya hemos visto con el comando:

```
#iptables-save > reglas_iptables_31oct24 (en este ejemplo la fecha es 31 de octubre del 24)
```

Una vez que estamos seguros que estamos seguros que es este el fichero correcto y que funciona adecuadamente, entonces sí ejecutamos:

```
#netfilter-persistent save
```

Con lo que la presente configuración se almacenará en **/etc/iptables/rules.v4**.

Ahora que vamos comprendiendo la dinámica parcial y final de creación e reglas en iptables, vamos a proceder a dejarlas permanentes para que cuando arranque nuestro sistema operativo, se cargue esta última configuración, es decir: /etc/iptables/**rules.v4**.

El primer paso es situarnos en el directorio: **/etc/network/if-pre-up.d** que es desde donde se configuran las interfaces de red al iniciar el sistema operativo.

Desde este directorio, crearemos un fichero llamado "iptables" con las dos líneas que figuran a continuación:

**#!/bin/sh**

**/sbin/iptables-restore < /etc/iptables/rules.v4**

A este fichero "iptables" que acabamos de crear, le damos permiso de ejecución con el siguiente comando:

**chmod +x /etc/network/if-pre-up.d/iptables**

En la imagen siguiente resaltamos en rojo, cada uno de estos pasos.

```
┌──(root㉿kali)-[/home/acorletti]
└─# cd /etc/network

┌──(root㉿kali)-[/etc/network]
└─# cd if-pre-up.d

┌──(root㉿kali)-[/etc/network/if-pre-up.d]
└─# pwd
/etc/network/if-pre-up.d

┌──(root㉿kali)-[/etc/network/if-pre-up.d]
└─# ls -l
total 32
-rwxr-xr-x 1 root root  348 May 25 17:17 ethtool
-rwxr-xr-x 1 root root 7214 Dec  9  2023 ifenslave
-rwxr-xr-x 1 root root   58 Nov  4 11:46 iptables
lrwxrwxrwx 1 root root   28 Jun 17  2018 macchanger -> ../../macchanger/ifupdown.sh
-rwxr-xr-x 1 root root 4224 Jul 16 10:33 vlan
-rwxr-xr-x 1 root root 4191 Apr 16  2024 wireless-tools
lrwxrwxrwx 1 root root   32 Apr 28  2024 wpasupplicant -> ../../wpa_supplicant/ifupdown.sh

┌──(root㉿kali)-[/etc/network/if-pre-up.d]
└─# cat iptables
#!/bin/sh
/sbin/iptables-restore < /etc/iptables/rules.v4

┌──(root㉿kali)-[/etc/network/if-pre-up.d]
└─# chmod +x if-pre-up.d/iptables
```

Con estos tres pasos, si ahora reinicias el sistema (**reboot**) verás que tus reglas de **iptables**, están exactamente igual a las que guardaste en **/etc/iptables/rules.v4**

Hemos logrado garantizar que ante cualquier caída y reinicio de nuestro sistema operativo, las reglas de **iptables** se mantendrán acorde a la última configuración que hemos guardado en **/etc/iptables/rules.v4**

Con toda humildad, si alguien encuentra la forma que "iptables-persistent" se ejecute automáticamente al reiniciar el sistema operativo en "**Kali**", que nos lo diga, pues para nosotros ha sido ¡¡**imposible**!!.

**Ejercicio 15:** Algunos desafíos

a. Investiga el borrado de líneas de forma puntual. Solo te dejamos un ejemplo:

   El comando: **#iptables -L --line-numbers** te indica la línea correspondiente a cada una de las cadenas.

   Por ejemplo, si ejecutas: **#iptables -D INPUT <Number>** te borrará esa línea específica de la cadena **INPUT**.

   Practica con todo el detalle posible qué otras opciones de borrado posee **iptables**.

   Te invitamos a que profundices en esta tarea, pues puedes hacerlo una a una, de forma selectiva, todas ellas, etc... (dedícale su tiempo).

b. Cada vez que desde una consola ejecutas: **#iptables -..... [enter]**, ingresas una nueva regla a "netfilter" y AUTOMÁTICAMENTE la misma entra en ejecución.

   Es muy probable que en los intentos anteriores, te haya sucedido que aplicabas una regla, luego cambiabas algo y las cosas no respondían como querías. Esto sucede porque cada nueva regla que ingresaste, se "encola" en el sistema netfilter, por lo tanto tú creías que estabas ejecutando "esa" regla, cuando en verdad estabas ejecutando una "lista" de reglas, y en vez de cumplirse esa última se estaba cumpliendo alguna anterior.

   Como ya hemos visto, para poder ver todas las reglas que tienes ejecutándose debes escribir:

   **#iptables – L**

   Allí te listará todo lo que has escrito (siempre que su formato haya sido correcto) y se está ejecutando en tu host.

   Tú que ya estás familiarizado con Linux, ¿Qué opción se te ocurre que puedes colocar inmediatamente después de "-L" para que te ofrezca más detalle este listado?

c. ¿En qué consiste la opción "Masquerade"?

d. ¿Cuál es el resultado de la siguiente regla?:

   iptables -A INPUT -s 172.18.1.100 -p TCP --dport 22 -j ACCEPT

e. ¿Cómo debería ser la regla para permitir que todos los usuarios de la red: 192.168.10.0 con máscara: 255.255.255.0 puedan acceder al servidor web de la empresa, que tiene instalado también **iptables**?

f. Hasta ahora hemos tratado las reglas una a una, pero una potente opción que nos presenta la "programación bash" es, redactar un sencillo archivo de texto plano, colocar todo el conjunto de reglas que queramos, asignarle los permisos correspondientes con "**chmod**" y luego ejecutarlo (./).

¿Te atreves a comenzar con uno de ellos que inicialmente contenga una regla sola y luego continuar sumándole más?

g. Un trabajo con dos hosts:

Te proponemos a continuación una secuencia de tareas para que ejercites parte de lo visto hasta ahora desde dos hosts. Uno de ellos debería ser Linux con **iptables** y el otro podría ser también Linux o cualquier otro sistema operativo. Te invitamos a que sigas esta secuencia paso a paso para que puedas analizar su evolución:

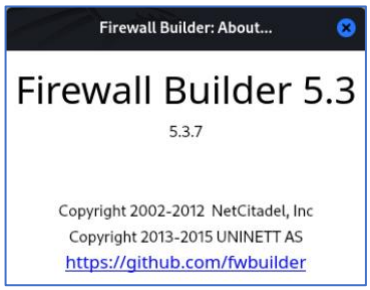
1) Realizar scan con **nmap** hacia los puertos tcp y udp de ambas máquinas.
2) Verificar con **nmap** localhost que los datos sean los correctos, o hacerlo también con **Zenmap**.
3) Verificar desde una consola a través del comando "**netstat –etn",** qué puertos tengo establecidos con conexión (Established).
4) Vamos a instalar una herramienta muy sencilla que permite realizar conexiones remotas hacia ese host, que aún no haremos, pero nos permitirá verificar la apertura y cierre del puerto 23 (telnet). Para esta actividad, desde el host Linux debemos instalar "**telnetd**". Esta aplicación automáticamente al finalizar su instalación se ejecutará y dejará abierto el puerto TCP 23.
5) verificar nuevamente con **nmap** y **zenmap** ambos equipos.
6) Verificar cómo tengo las reglas de mi **iptables**, ¿Qué comando empleo?
7) Borrar todas las reglas de mi **iptables** ¿Qué comando empleo?
8) ¿Con qué regla de **iptables** puedo negar acceso al puerto 23? (en el host que se instaló telnetd).
9) ¿Con qué regla de **iptables** puedo negar acceso al puerto 23 a todo el mundo menos a la dirección IP del otro host?
10) ¿Con qué regla de **iptables** puedo negar la salida a navegar por Internet desde este mismo equipo?
11) ¿Qué sucede si apago este host y la vuelvo a encender?
12) ¿Es posible hacer un programa sencillo para que se niegue el acceso a este puerto TCP 23, excepto al otro host y ejecutar directamente este programa, en vez de regla por regla?
13) Cómo se puede hacer para que este programa se inicie siempre que arranque este host?

# 2. FirewallBuilder

**Ejercicio 16:**

Empleo de la herramienta "**Firewall Builder**". El desarrollo de este proyecto se encuentra en la siguiente URL:

**https://fwbuilder.sourceforge.net**




Para instalarlo en nuestro **Kali**, sencillamente son dos pasos:

1) #**apt-get update**
2) #**apt-get install fwbuilder**.

En nuestra opinión esta herramienta gráfica es una de las más potentes que podemos llegar a emplear en temas de seguridad con Firewalls.

Existen ofertas comerciales de alto coste que no reúnen la potencia que ofrece Firewall Builder. En esta guía de ejercicios trabajaremos con la versión 5.3.

Para su instalación, puede ser que te encuentres con algún nivel de complejidad, dependiendo de la distribución de Linux que emplees, de su actualización y sobre todo de las librerías y paquetes que tengas instalado, no debería ser así con las versiones más recientes de **Kali**. Por nuestra parte, como hemos intentado hacer en todo el artículo, no entraremos en temas de instalación pues suelen cambiar muy periódicamente, pero como siempre estamos seguros que es una muy buena práctica en temas de seguridad, que seas capaz de investigar a través de Internet hasta encontrar la solución a cada uno de estos problemas y desafíos.

En el caso de "**firewallbuilder**" te aconsejamos que no escatimes esfuerzos hasta que lo tengas funcionando al 100% y no te quede ningún aspecto del mismo sin conocer pues será el día a día de todo administrador de sistemas que se quiera preciar de "experto en seguridad".

En pocas palabras, es una herramienta "excelente".

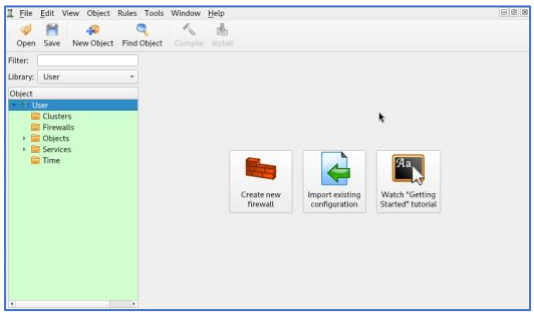
Todo este trabajo se puede ampliar, tomando como referencia la documentación (que es abundante) y se puede descargar gratuitamente la "**Firewall Builder 5 user guide**" desde: https://fwbuilder.sourceforge.net/4.0/docs/users_guide5/UsersGuide5.pdf

A riesgo de quedar desactualizados en poco tiempo, igualmente hemos decidido recomendar la esta guía, la cual citaremos en estos ejercicios.

Una vez instalado, desde consola, puedes ejecutar "**#fwbuilder**", se abrirá su interfaz gráfica y te presentará diferentes opciones de "escenarios" por defecto, te aconsejamos que elijas uno sencillo con no más de dos interfaces. Luego te preguntará sobre qué plataforma se trabajará, en nuestro caso seleccionamos "**iptables**" que es la que venimos tratando.

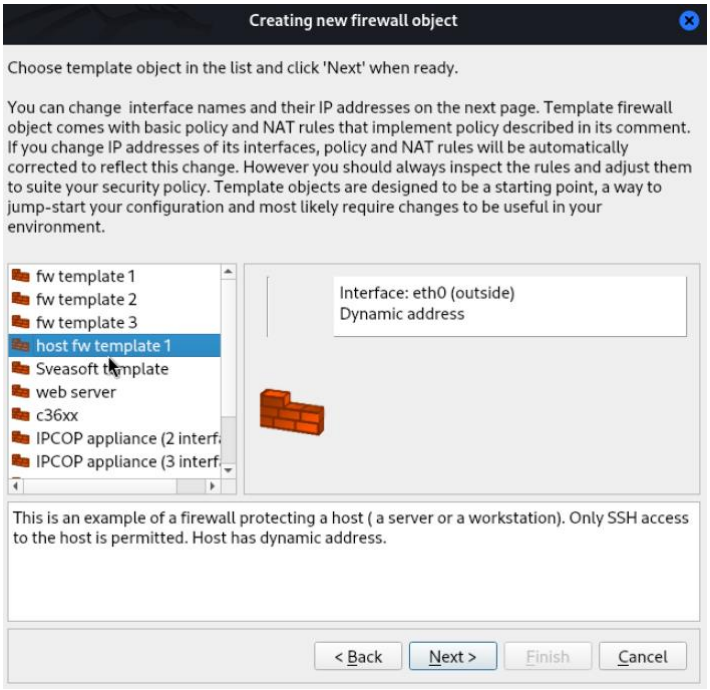

Al abrir la interfaz gráfica por primera vez, te ofrecerá la posibilidad de acceder a una guía, si estás con tiempo, te vendría muy bien que lo hagas.

1) Una vez abierta la consola, el primer paso es crear un nuevo FW (nombre, interfaces, etc. - Activar la opción "Plantilla por defecto"). (*Puede consultarse en el capítulo 4 de la guía).*

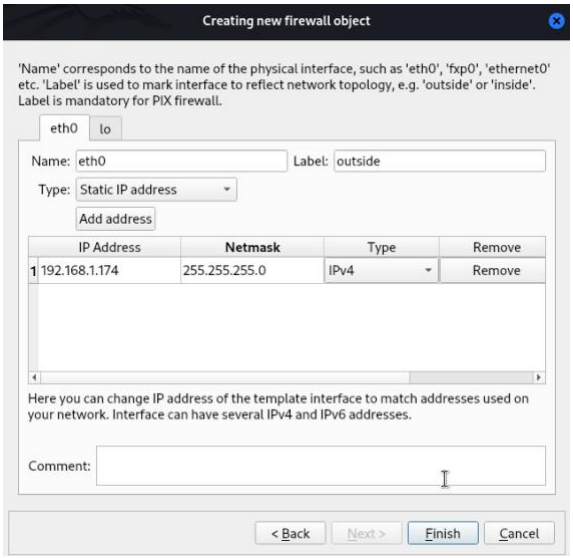



En nuestro caso lo llamaremos "Firewall_ejercicio", elegiremos "iptables" y el SO Linux 2.4/2.6

2) De las plantillas que nos ofrece, en nuestro caso vamos a seleccionar "host fw template 1 (como puedes ver en la imagen de la derecha) por ser la que nos ofrece una configuración sencilla, con una sola interfaz de red.
Por supuesto que puedes seleccionar la que desees, el día que tengas otro tipo de firewall con mayor nivel de complejidad.



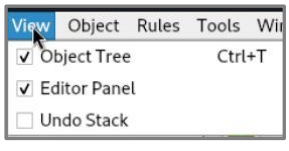

3) Para mantener lo que aprendimos y ejercitamos en las páginas anteriores, vamos a configurar la interface (eth0: con la IP que tenga nuestra máquina virtual **Kali**, que en nuestro caso es la eth0 con la dirección IP "Static" **192.168.1.174,** máscara 255.255.255.0.




4) Al finalizar con la configuración de la interfaz de red, automáticamente **fwbuilder** nos presentará una configuración estándar de reglas con las que podemos empezar a trabajar en este ejercicio.

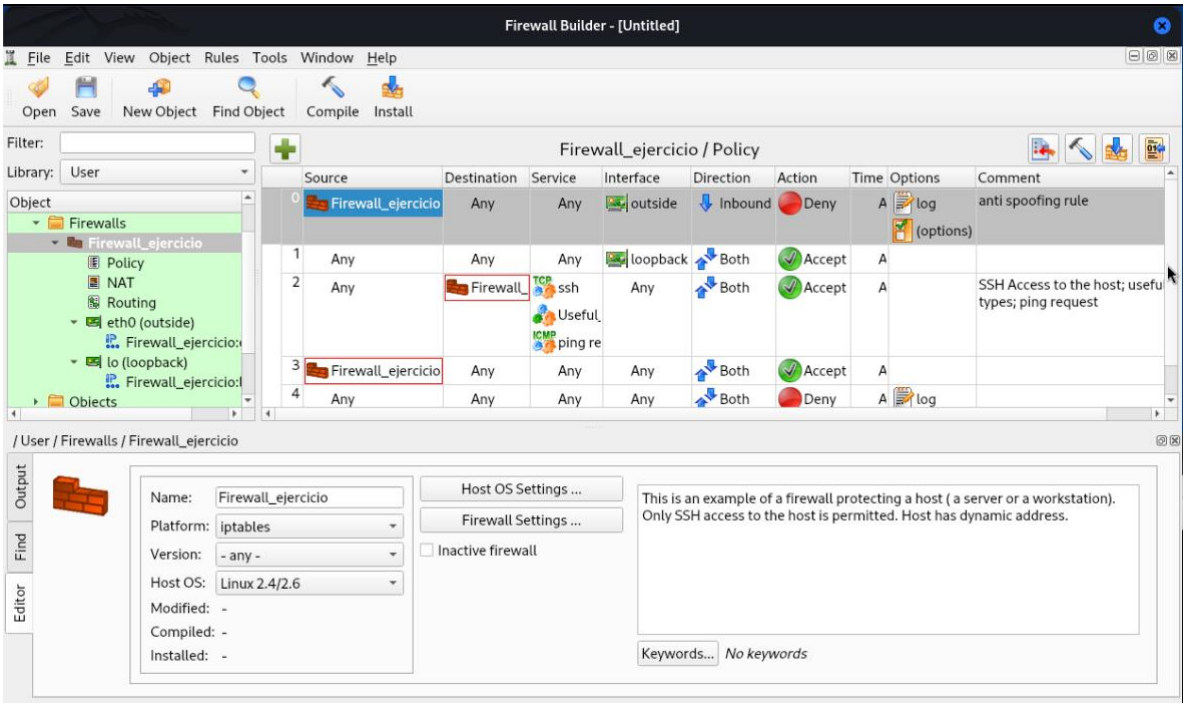
En la imagen que sigue abajo, puedes ver que ya nos ha creado nuestro "**Firewall_ejercicio**" con cuatro reglas que nos propone inicialmente, las cuales por supuesto, podemos modificar con total libertad hasta dejar la configuración que más nos guste.

Pasemos a analizar la imagen que sigue. En la misma, podemos



ver dos paneles (*si seleccionamos "**View**" nos los muestra*), el "**Árbol de objetos**" que es l parte superior, y el "**Panel Editor**" que es la parte inferior.

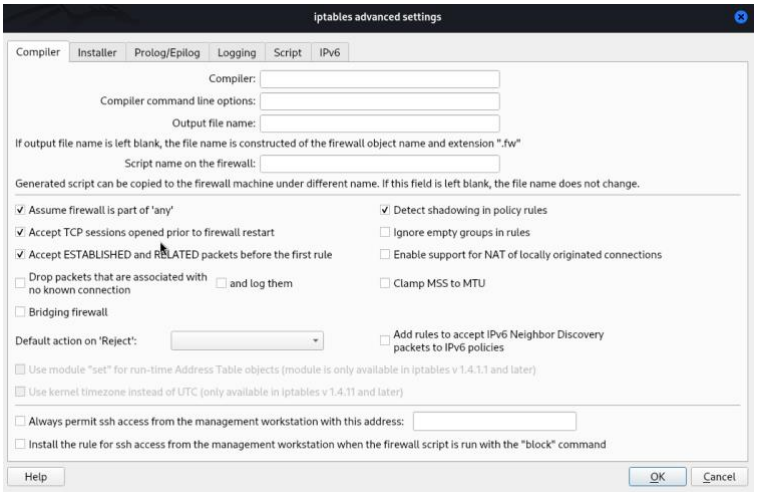
A la derecha del árbol de objetos, podemos ver numeradas las cuarto reglas que nos ha creado automáticamente. El formato de las mismas, no debe sorprendernos pues muy similar a las que ya hemos trabajado.




Las interfaces, puedes editarlas y modificarlas las veces que lo desees, haciendo "doble click" sobre ellas, se despliega la ventana de edición, como puedes ver en la imagen de aquí al lado.

(La ventana de "edición" la puedes abrir sobre cualquier "objeto" haciendo "doble clic" sobre este).

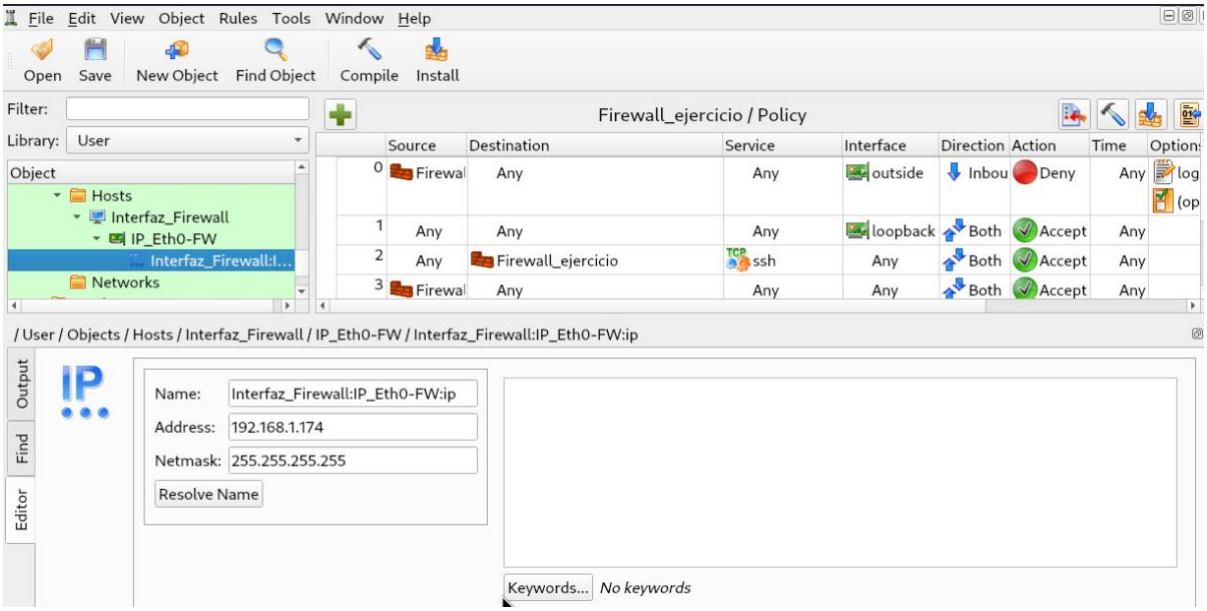
5) Seleccionar el FW recientemente creado, editarlo (con doble "Click") y seleccionar el botón "**Firewall Setting**", se abrirá una nueva ventana "**iptables advanced settings**". En la parte inferior izquierda está el botón de "Help" que describe con todo detalle sus opciones. Analizar brevemente las mismas (como se ve en la primera de las imágenes).



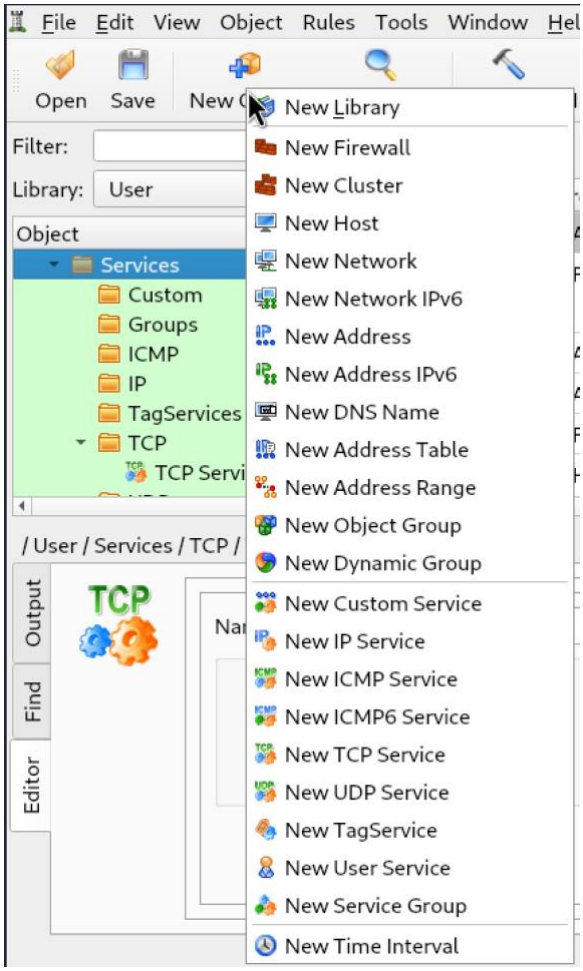

6) Al haber creado un FW, como hemos dicho, nos ofrece un conjunto de reglas por defecto que se corresponderán a la "**política de este FW**". Podemos revisar y ajustar las reglas para la red. Un consejo "Comentar TODO", el realizar comentarios sobre las reglas es fundamental pues nos servirá a futuro a nosotros o a cualquier otro que deba realizar cualquier tarea sobre el mismo. Como dato interesante, en la realidad, hemos visto FWs con cientos o miles de reglas en los cuáles ya era imposible "adivinar" para qué estaban muchas de ellas.

7) Vamos a ir creando y configurando elementos en FWBuilder. Lo primero que haremos es asignar un "nuevo Host", que en nuestro ejemplo será justamente nuestra interfaz eth0, pero el día de mañana desde aquí deberías asignar todos los host importantes de tu arquitectura sobre los que debas aplicar reglas específicas (servidores Web, BBDD, correo, ficheros, etc.).

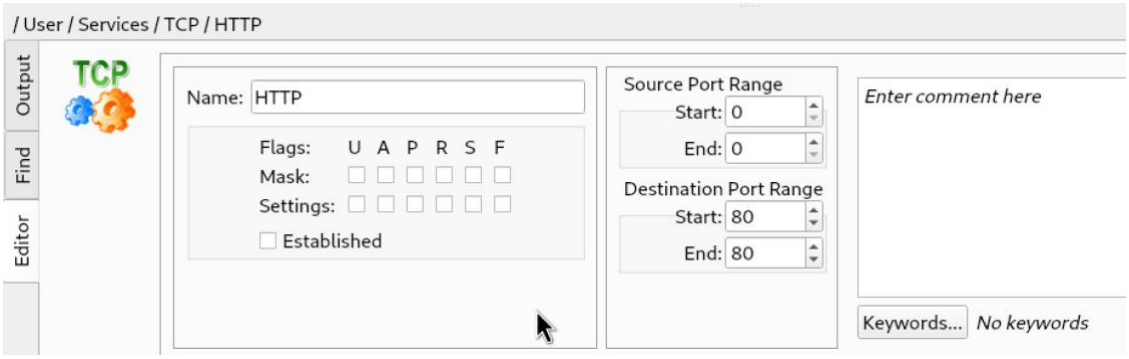
   En la figura de abajo mostramos nuestro ejemplo.



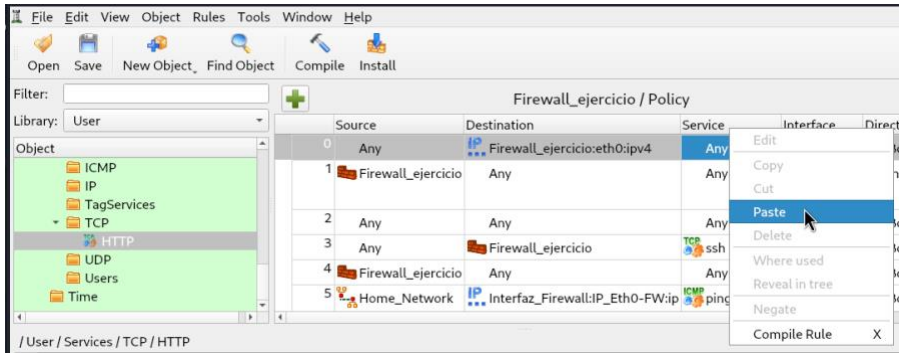

8) La mayoría de las interfaces de gestión de firewalls, suelen emplear la palabra "Objetos" para abarcar un número de elementos que podemos configurar. Si presionamos en el signo "**+**" del botón superior que dice "**New Object**" podemos ver que desde aquí podemos crear nuevos Firewalls, Cluster, Hosts, Networks, etc.
   En la imagen de la derecha podemos ver todo el listado que se pueden crear.



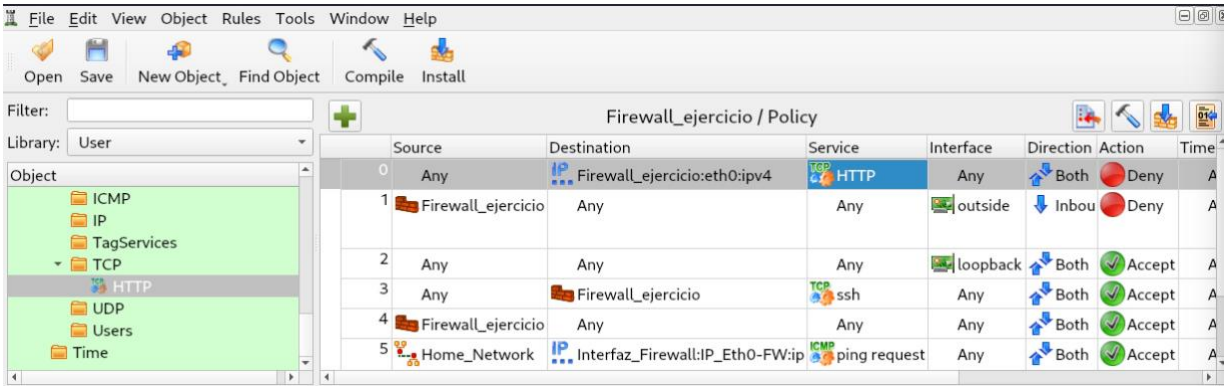

   Como se presenta en la imagen que sigue abajo, crearemos un nuevo Servicio TCP para poder operar sobre el puerto TCP 80, es decir el protocolo HTTP (Hiper Text Transfer Protocol).
   Una vez seleccionado "New TCP Service" se nos abre la ventana que se presenta a continuación, en la que podemos ver que hemos seleccionado el destination por range 80, y de nombre le hemos puesto HTTP.



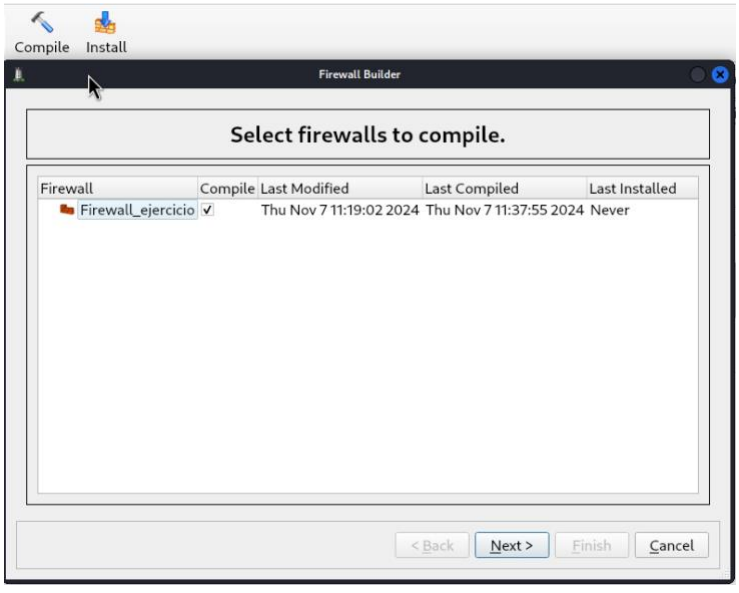

9) Ahora que hemos creado este nuevo objeto como "servicio HTTP" podemos volver a la regla que hemos creado y aplicarle cono servicio destino "paste" HTTP.



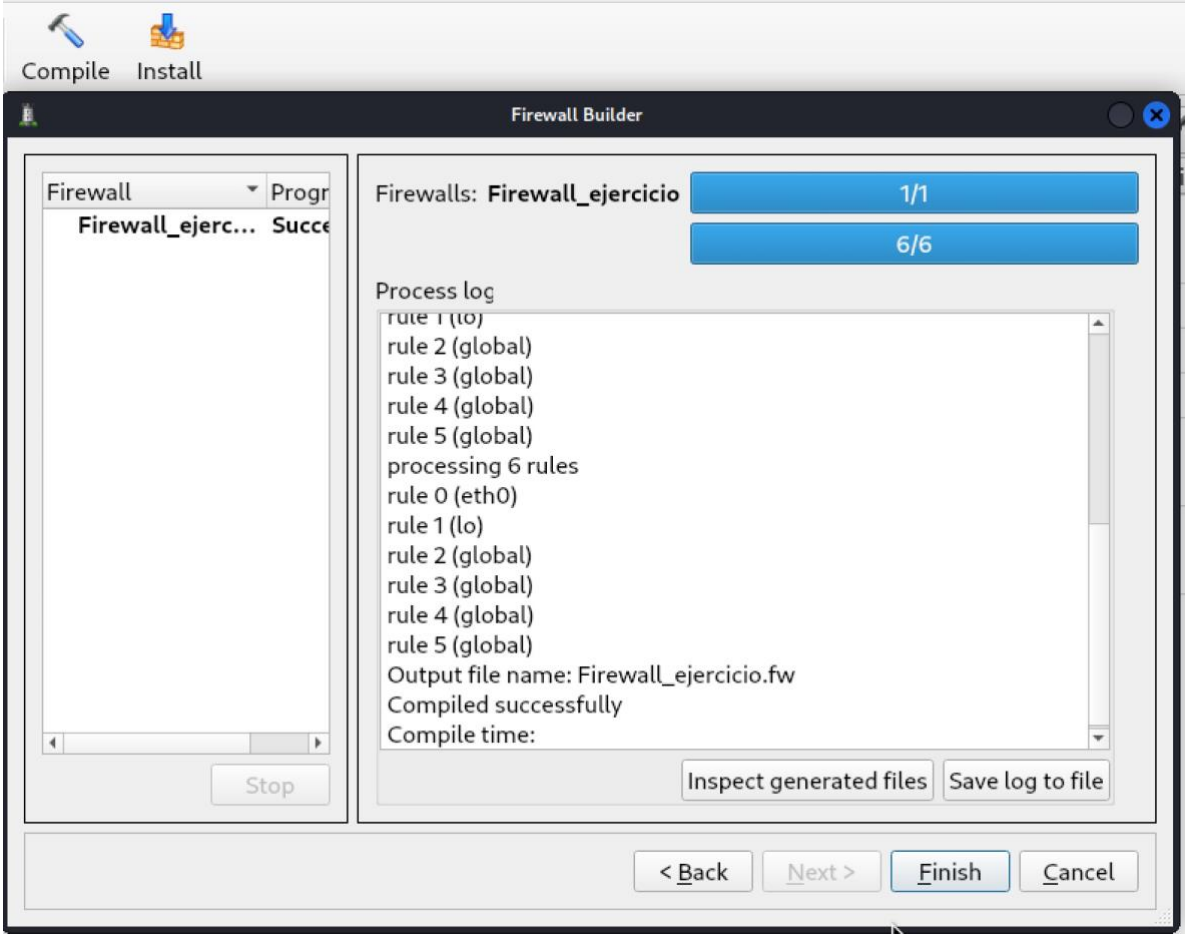

Una vez aplicado, la regla nos quedará como la imagen que figura abajo.



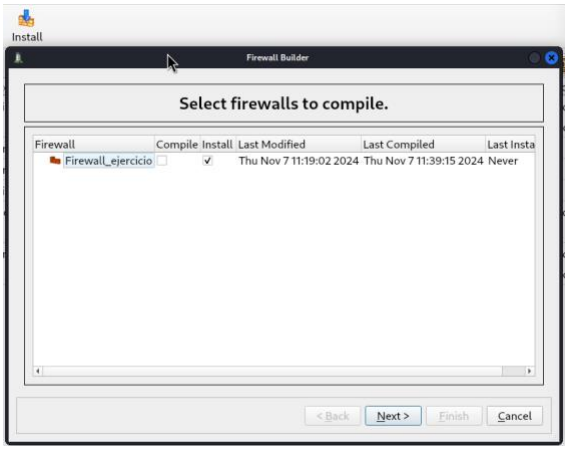

10) Cuando hemos completado el conjunto de reglas que deseamos aplicar a nuestra "policy", el paso siguiente es "compilarlas" para que la interfaz gráfica las traduzca al lenguaje de **iptables**. Para ello, seleccionamos el botón "**Compile**" del menú superior y se nos desplegara la ventana que se presenta a la derecha para que seleccionemos a qué firewall deseamos compilar, en nuestro caso como solo tenemos este primero, solo nos presenta uno.



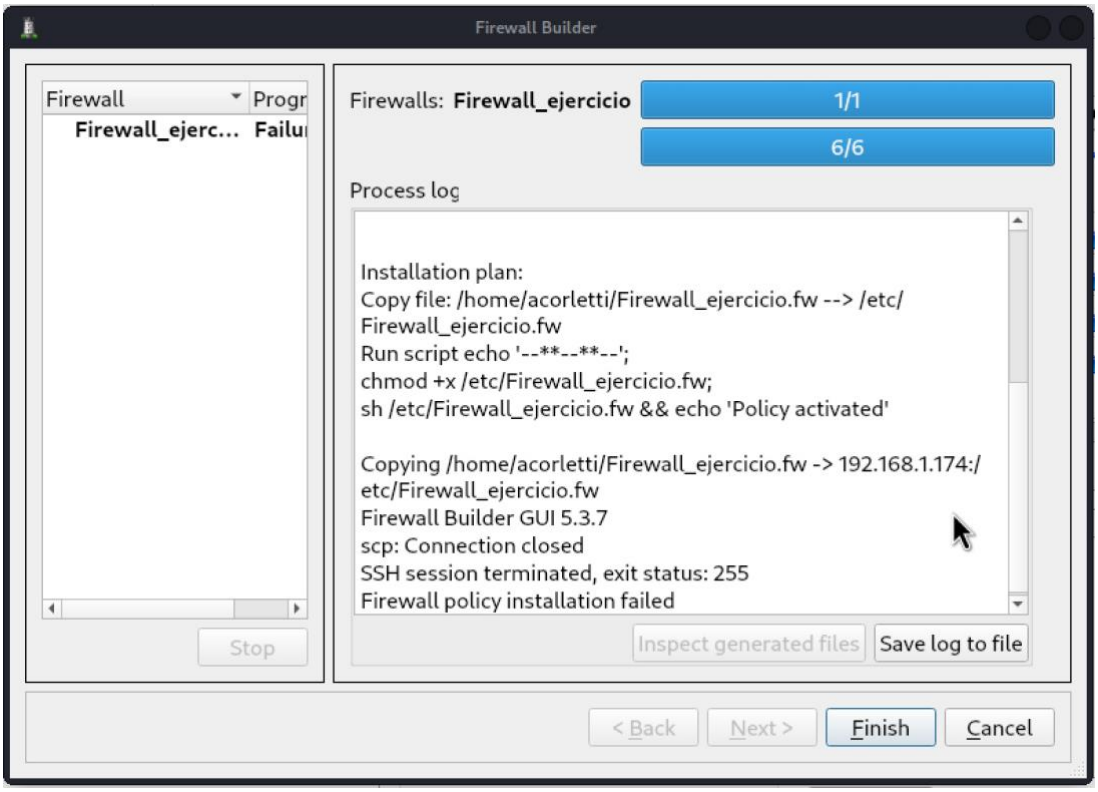
Si presionamos el botón “Next”, comenzará el proceso de compilación y si todo está OK, nos presentará la imagen que figura abajo.




11) El paso final es la “instalación” de esta reglas en el firewall correspondiente. Una vez más, como solo tenemos, nos aparecerá este única opción, pero si fueran más, nos aparecería el listado de firewalls para que seleccionemos en cuál de ellos, o cuáles, se desean instalar estas reglas, pues justamente puede ser una instalación masiva en más de un firewall que operen sobre la misma política, por lo tanto la interfaz gráfica nos permitiría realizar una instalación masiva. Seleccionaremos el botón “**Install**” de la barra superior y se nos presentará la ventana que figura a la derecha.



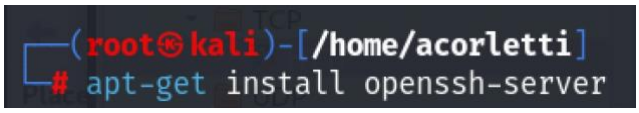

12) Ups… ¿Qué nos ha sucedido, según la imagen de abajo, nos da un error…




13) Pues, resulta ser que por defecto las nuevas versiones de "**Kali**" no traen instalada la posibilidad de conexión remota por medio del protocolo SSH, con lo que debemos instalar este comando y habilitar el acceso SSH, que dicho sea de paso nos servirá también para poder realizar conexiones remotas seguras, así que hagámoslo.

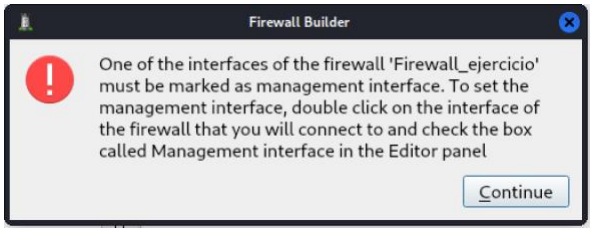
El primer paso es ejecutar el comando:

**#apt-get install openssh-server**




A continuación, se debe reiniciar el servicio con el comando:

**#systemctl restart ssh.service**


Puede que también debamos asociarle una interfaz de red para la gestión (*management*) de este acceso, si es así, nos aparecerá el mensaje que figura a la izquierda.

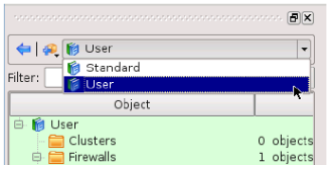
Si nos sucede esto, seleccionaremos la interfaz "**eth0**" y veremos que tiene una casilla para "**Management interface**", con lo que ya tenemos seleccionada por qué interfaz deberá realizarse el acceso de gestión.

14) Una vez resuelto estos temas, solo nos queda volver al botón “Install” y repetir la operación, nos pedirá cuál es la cuenta de acceso por SSH a ese firewall, en nuestro caso es “root”, con lo que seleccionaremos esta cuenta con su respectiva contraseña, y ahora sí al presionar en “Install” debería quedar instalada esta “policy” en el firewall que hemos elegido.


SI queréis profundizar más en esta herramienta, a continuación os dejamos algunos ejercicios adicionales.

15) Cambiar de color las reglas que parezcan más importantes (puede consultarse el capítulo 5 de la guía).

16) Analizar las Librerías "Standard" y "User", ¿Qué servicios y objetos incorpora por defecto?




17) Para profundizar en el empleo de la GUI de FWBuilder, puede consultarse en la figura 5.30 de la guía)

18) Crear nuevas redes (por ejemplo por ciudades: Valencia, Santiago. O zonas: DMZ, etc) (puede consultarse el capítulo 5.19 de la guía). Practicar el “Drag and Drop” arrastrando los objetos que desees.

19) Crear troncales (10.0.0.0, 172.16.0.0).

20) Crear Hosts (Servidores de correo, de DMZ, etc.).

21) Crear Grupos (Directorio, Administradores, etc...) (Asignarle a cada uno algunos hosts, que deberán ser creados).

22) Configurar diferentes opciones en la ventana "Filter" para visualizar únicamente ciertos rangos, nombres, interfaces, etc.

23) Analizar las opciones de "Logs".

24) Analizar "Inspeccionar Filas Generadas" (icono derecho de los 4 del menú). ¿Qué es esto? (Describir brevemente algunas reglas).

25) Si se desea profundizar sobre FWBuilder, una opción interesante y afín al trabajo que se viene realizando en el texto, es la del punto 4.2. de la guía "Configuring Cisco Router ACL", donde se describe la compatibilidad entre estos productos. (en la figura 4.5.3 se ve con claridad el empleo de las ACLs)

Para finalizar todo este ejercicio con FWBuilder, a continuación os pegamos el formato que con sus más o sus menos debería quedar en un archivo de configuración compilado en formato "**iptables**" (hemos recortado con "....." muchas de sus partes, dejando únicamente las que consideramos te pueden servir de referencia en los pasos que acabamos de realizar):

```
#!/bin/sh
#
#  This is automatically generated file. DO NOT MODIFY !
#  Firewall Builder  fwb_ipt v4.2.0.3530
# Compiled for iptables (any version)
.....

FWBDEBUG=""
PATH="/sbin:/usr/sbin:/bin:/usr/bin:${PATH}"
export PATH
LSMOD="/sbin/lsmod"
IPTABLES="/sbin/iptables"
.....

LOGGER="/usr/bin/logger"
log() {
   echo "$1"
   command -v "$LOGGER" >/dev/null 2>&1 && $LOGGER -p info "$1"
}
.....

reset_iptables_v4() {
  $IPTABLES -P OUTPUT  DROP
  $IPTABLES -P INPUT   DROP
  $IPTABLES -P FORWARD DROP
.....

configure_interfaces() {
   :
   # Configure interfaces
   update_addresses_of_interface "eth0 172.16.0.1/16" ""
   update_addresses_of_interface "eth1 192.168.0.1/16" ""
   update_addresses_of_interface "lo 127.0.0.1/8" ""
.....

   # ================ Table 'filter', automatic rules
   # accept established sessions
   $IPTABLES -A INPUT   -m state --state ESTABLISHED,RELATED -j ACCEPT
   $IPTABLES -A OUTPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT
   $IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

   # ================ Table 'nat',  rule set NAT
   #
   # Rule 0 (NAT)
   #
   echo "Rule 0 (NAT)"
   #
   $IPTABLES -t nat -A POSTROUTING -o eth0  -s 192.168.0.0/16  -j SNAT --to-source 172.16.0.1

   # ================ Table 'filter', rule set Policy
   #
```

```
    # Rule 0 (eth0)
    #
    echo "Rule 0 (eth0)"
    #
    # anti spoofing rule (sirve para detectar y negar accesso a IP falsas)
    $IPTABLES -N In_RULE_0
    $IPTABLES -A INPUT -i eth0   -s 172.16.0.1   -j In_RULE_0
    $IPTABLES -A INPUT -i eth0   -s 192.168.0.1   -j In_RULE_0
    $IPTABLES -A INPUT -i eth0   -s 192.168.0.0/16   -j In_RULE_0
    $IPTABLES -A FORWARD -i eth0   -s 172.16.0.1   -j In_RULE_0
    $IPTABLES -A FORWARD -i eth0   -s 192.168.0.1   -j In_RULE_0
    $IPTABLES -A FORWARD -i eth0   -s 192.168.0.0/16   -j In_RULE_0
    $IPTABLES -A In_RULE_0  -j LOG  --log-level info --log-prefix "RULE 0 -- DENY "
    $IPTABLES -A In_RULE_0  -j DROP
.............

    # All other attempts to connect to
    # the firewall are denied and logged
    $IPTABLES -N RULE_7
    $IPTABLES -A OUTPUT  -d 172.16.0.1   -j RULE_7
    $IPTABLES -A OUTPUT  -d 192.168.0.1   -j RULE_7
    $IPTABLES -A INPUT  -j RULE_7
    $IPTABLES -A RULE_7  -j LOG  --log-level info --log-prefix "RULE 7 -- DENY "
    $IPTABLES -A RULE_7  -j DROP
```

# 3. nftables

En nuestra opinión, este desarrollo es la evolución natural de **iptables**, pero ajustado y optimizado para el empleo más directo de todas las librerías del proyecto "**netfilter**". Tiene varias características destinada a optimizar su ejecución y rendimiento, pero para nosotros en este texto, lo que más nos interesa es que posee una sintaxis más coherente y compacta, lo cual hace que sea más sencillo de utilizar.

Si deseáis profundizar más acerca de todas la ventajas que nos ofrece, respecto a **iptables**, en los enlaces que pasamos a continuación, tenéis todas las explicaciones que podáis desear.

Dentro de "**Netfilter**" se encuentra el desarrollo de **nftables**, podemos acceder a la totalidad de su desarrollo en el siguiente enlace:

https://netfilter.org/projects/nftables/

También tienes una Wiki muy buena sobre

https://wiki.nftables.org/

Lo primero que presentamos es, desde esta misma Wiki, los pasos de su instalación, que figuran en las imágenes de aquí abajo.


En nuestro caso, como seguimos trabajando con nuestro "**Kali**" Linux, como ya hemos ejecutado "**update**" en el ejercicio 16, solamente deberemos hacer:

```
#apt install nftables
```

```
┌──(root㉿kali)-[/home/acorletti]
└─# apt install nftables
The following packages were automatically installed and are no longer required:
  ibverbs-providers  libgfrpc0      libibverbs1        librdmacm1t64    python3.11-dev
  libcephfs2         libgfxdr0      libpython3.11-dev  python3-lib2to3  python3.11-minimal
  libgfapi0          libglusterfs0  librados2          python3.11       samba-vfs-modules
Use 'sudo apt autoremove' to remove them.

Upgrading:
  libnftables1  libnftnl11  nftables

Summary:
  Upgrading: 3, Installing: 0, Removing: 0, Not Upgrading: 1683
  Download size: 0 B / 439 kB
  Space needed: 68.6 kB / 47.2 GB available

Continue? [Y/n] 
```

Nos preguntará si deseamos continuar, le responderemos que si “Y”, y listo.

```
Continue? [Y/n] Y
(Reading database ... 384490 files and directories currently installed.)
Preparing to unpack .../libnftnl11_1.2.8-1_arm64.deb ...
Unpacking libnftnl11:arm64 (1.2.8-1) over (1.2.7-1) ...
Preparing to unpack .../nftables_1.1.1-1_arm64.deb ...
Unpacking nftables (1.1.1-1) over (1.1.0-2) ...
Preparing to unpack .../libnftables1_1.1.1-1_arm64.deb ...
Unpacking libnftables1:arm64 (1.1.1-1) over (1.1.0-2) ...
Setting up libnftnl11:arm64 (1.2.8-1) ...
Setting up libnftables1:arm64 (1.1.1-1) ...
Setting up nftables (1.1.1-1) ...
Processing triggers for man-db (2.12.1-2) ...
Processing triggers for libc-bin (2.40-2) ...

┌──(root㉿kali)-[/home/acorletti]
```

Una vez instalada como casi todas las herramientas de Linux, nos ofrece un manual, que en este caso particular es sumamente completo, y lo podemos ver con el comando:

# man nft

```
┌──(root㉿kali)-[/home/acorletti]
└─# man nft
```

```
NAME
       nft - Administration tool of the nftables framework for packet filtering and classification

SYNOPSIS
       nft [ -nNscaeSupyjtT ] [ -I directory ] [ -f filename | -i | cmd ...]
       nft -h
       nft -v

DESCRIPTION
       nft is the command line tool used to set up, maintain and inspect packet filtering and classification
       rules in the Linux kernel, in the nftables framework. The Linux kernel subsystem is known as
       nf_tables, and ‘nf’ stands for Netfilter.

OPTIONS
       The command accepts several different options which are documented here in groups for better
       understanding of their meaning. You can get information about options by running nft --help.

       General options:

       -h, --help
           Show help message and all options.

       -v, --version
           Show version.
```

Lo primero que verificaremos y ejecutaremos es que el servicio esté habilitado, y estableceremos también que el servicio **nftables** arranque con el sistema, lo haremos con el comando:

```
# systemctl enable nftables.service
```

Podemos verificar su estado con el comando:

```
# service nftables status
```

Esta herramienta, al instalarla ya creo una configuración por defecto que es sobre la que iremos operando permanentemente, la misma se encuentra en el directorio "/etc" y podemos verla con el comando:

```
# cat /etc/nftables.conf
```

Obtenemos el mismo resultado, pero con una vista más amigable con:

```
#nft list ruleset
```

```
┌──(root㉿kali)-[/home/acorletti]
└─# nft list ruleset
table inet filter {
        chain input {
                type filter hook input priority filter; policy accept;
                oifname "eth0" icmp type echo-reply
                oifname "eth0" icmp type echo-reply
                oifname "eth0" icmp type echo-reply drop
        }

        chain forward {
                type filter hook forward priority filter; policy accept;
        }

        chain output {
                type filter hook output priority filter; policy accept;
        }
}
```

El formato básico de nftables será siempre algo similar a:

```
#nft (list | add | delete | flush) table [<family>] <name>
```

Para que no lo olvides desde aquí y en adelante, recordad que siempre podemos borrar todo con un comando muy similar a iptables. En este caso es:

```
#nft flush ruleset
```

**"nftables"** se compone de **tablas**, y las tablas de **cadenas** y las cadenas de **reglas**. Si aún lo recordáis, es muy similar a **iptables**.

Lo primero que tenemos que hacer para comenzar a trabajar con **nftables** es agregar al menos una **tabla**. Luego, podemos agregar cadenas y agregaremos las reglas a esas cadenas.

**Ejercicio 17:** Agregar **cadenas** y **reglas**.

Comenzaremos creando una nueva cadena:

```
┌──(root㉿kali)-[/home/acorletti]
└─# nft add chain inet filter ejemplo-1
```

```
# nft add chain inet filter ejemplo-1
```

Si ejecutamos nuevamente:

```
#nft list ruleset
```

```
# nft list ruleset
table inet filter {
        chain input {
                type filter hook input priority filter; policy accept;
                oifname "eth0" icmp type echo-reply
                oifname "eth0" icmp type echo-reply
                oifname "eth0" icmp type echo-reply drop
        }

        chain forward {
                type filter hook forward priority filter; policy accept;
        }

        chain output {
                type filter hook output priority filter; policy accept;
        }

        chain ejemplo-1 {
        }
}
```

Podemos ver en la imagen de la derecha que ya tenemos esta nueva **cadena** dentro de la tabla "**inet filter**", la cual por ahora está vacía.

Vamos a crear las mismas reglas de filtrado de "ping" que habíamos empleado con "**iptables**" para que podamos compararlas.

En concreto vamos a restringir la respuesta a cualquiera que nos haga **ping** a nuestra máquina virtual **Kali** (para poder verificar de forma sencilla su funcionamiento), el comando sería:

```
#nft add rule inet filter input iifname "eth0" icmp type echo-reply drop
```

Como podemos ver en este primer ejemplo, lo estamos haciendo sobre la cadena de entrada "**input**", y dentro de la misma, designamos la interfaz de entrada "**iifname**" (si fuera la de salida, sería "**oifname**"), y solamente estamos negando la réplica. Lo ideal sería operar la réplica sobre "oifname" y el "request" sobre "iifname" ¿no te parece mejor?, pues hazlo y pruébalo en el ejercicio siguiente.

```
┌──(root㉿kali)-[/home/acorletti]
└─# nft list table inet filter
table inet filter {
        chain input {
                type filter hook input priority filter; policy accept;
                iifname "eth0" icmp type echo-reply drop
        }

        chain forward {
                type filter hook forward priority filter; policy accept;
        }

        chain output {
                type filter hook output priority filter; policy accept;
        }

        chain ejemplo-1 {
        }
}
```

Como se puede apreciar en la imagen anterior, ya tenemos esta una nueva reglas creadas, dentro de la cadena "**input**", la cual a su vez se encuentra dentro de la tabla "**inet filter**". En resumen, hemos llegado al final de la creación de una secuencia lógica de **nftables**:

**Tablas → cadenas → reglas**

## Ejercicio 18:

En el ejercicio anterior, hemos creado esta nueva regla para negar el "ping". En nuestro caso la máquina virtual **Kali** sobre la que estamos trabajando con **nftables** tiene la dirección IP: **192.168.1.177**. Si desde otro ordenador le enviamos un ping… responde ¿qué está sucediendo?

```
ale:~ ace$ ping 192.168.1.177
PING 192.168.1.177 (192.168.1.177): 56 data bytes
64 bytes from 192.168.1.177: icmp_seq=0 ttl=64 time=1.698 ms
64 bytes from 192.168.1.177: icmp_seq=1 ttl=64 time=0.971 ms
```

Pues justamente, que hemos creado esta nueva regla, pero aún no la hemos puesto en funcionamiento. Es decir, en estos momentos **nftables** sigue trabajando con la última configuración que hemos guardado que es la que está almacenada en: **# cat /etc/nftables.conf**

Para poner en producción esta nueva configuración, debemos ejecutar los siguientes pasos:

**#nft list ruleset > /etc/nftables.conf**

**#systemctl restart nftables.service**

Si ahora repetimos el "ping" nuestro resultado será:

```
ale:~ ace$ ping 192.168.1.177
PING 192.168.1.177 (192.168.1.177): 56 data bytes
ping: sendto: No route to host
ping: sendto: No route to host
```

Un detalle importante es que al ejecutar los dos comandos anteriores, también estamos guardando la configuración de arranque de **nftables**, con lo que al reiniciar este ordenador, mantendrá de forma permanente la configuración guardada y aplicará las reglas establecidas en cada reinicio que sufra.

## Ejercicio 19:

Tal cual hemos mencionado en el ejercicio anterior, ¿te atreves a implementar la regla para que boquee el "**echo-reply**" pero en la cadena "**oputput**"?

Cuidado: No te engañes, primero debes borrar la regla que acabamos de crear, pues si no lo haces, seguirá bloqueando por medio de esta regla y no sabrás si la tuya funciona o no, pero…

¿Cómo se borra una regla?

La forma más práctica es aplicando "**handle**" (asa/manija). Este comando nos indica el número (dónde está "anclada") de cada línea que hemos creado nosotros, o el mismo **nftables**.

Si ejecutamos: **#nft --handle list table inet filter**

Tal cual Podemos ver en la imagen de abajo, nos presenta los números que **nftables** le ha asignado a cada una de las reglas y cadenas.

```
┌──(root㉿kali)-[/home/acorletti]
└─# nft --handle list table inet filter
table inet filter { # handle 6
        chain input { # handle 1
                type filter hook input priority filter; policy accept;
                iifname "eth0" icmp type echo-reply drop # handle 5
        }

        chain forward { # handle 2
                type filter hook forward priority filter; policy accept;
        }

        chain output { # handle 3
                type filter hook output priority filter; policy accept;
        }

        chain ejemplo-1 { # handle 4
        }
}
```

En nuestro caso, la que deseamos borrar es la regla del “echo-reply”, la cual podemos ver que se trata del “handle 5”. Para borrar esa línea en concreto, el comando será:

```
#nft delete rule inet filter input handle 5
```

No olvidemos que para que este cambio se aplique al nftables que está ejecutándose en tiempo real en nuestro sistema, deberemos repetir:

```
#nft list ruleset > /etc/nftables.conf
#systemctl restart nftables.service
```

Ahora dejamos en tus manos la creación de una nueva regla que no responda al “ping”, pero desde la cadena “**output**”.

# 4. tests

Test 1: Un FW de red debería poder realizar las siguiente funciones (VERDADERO/FALSO):

a. Poder "escuchar" la totalidad del tráfico que deseemos analizar
b. Estar en capacidad de "desarmar" los encabezados de cada protocolo
c. Capacidad de descifrar tráfico con certificados digitales.
d. Soportar todos los protocolos de nivel de aplicación.
e. Poder detectar colisiones y corregirlas
f. Capacidad de enrutamiento

Test 2: Cuáles de las siguientes operaciones básicas sobre las cadenas que ofrece "iptables" son VERDADERAS:

a. "-t" (tabla): indica qué tabla se empleará.
b. "-i" (interfaz de entrada): mismo formato que ifconfig.
c. "-R" (redistribute) : redistribuye rutas.
d. "-z" (zone): aplica a una zona en concreto
e. "-s, -d": dirección IP fuente o destino (Se debe aclarara máscara, ej:/24)

Test 3: Sobre iptables (VERDADERO/FALSO):

a. Para Loguear las ocurrencias de una regla en iptables se emplea el comando "syslog"
b. Para guardar las reglas de iptables se emplea el comando "iptables-save"
c. Para guardar las reglas de iptables se emplea el comando "iptables -S"
d. Para guardar también puedo ejecutar el comando "iptables-persistent"
e. El comando **iptables -D INPUT <Number>** borra esa línea concreta

Test 4: Sobre Firewalbuilder (VERDADERO/FALSO):

a. Firewalbulider es un proyecto de sourceforge.net
b. Firewalbulider no soporta iptables
c. Cuando instalas Firewalbulider te propone escenarios por defecto
d. Las reglas de Firewalbulider no se pueden comentar
e. Para inyectar la policy en un firewall concreto es necesaria la conexión vía telnet

Test 5: nftables (VERDADERO/FALSO):

a. Los comandos de nftables comienzan con **nft**
b. La configuración por defecto de nftables se almacena en /var/nft.conf
c. El orden de nftables es tabla -> cadena -> regla
d. El Comando --handle nos fuerza una ejecución manual

## Respuestas a los tests

Test 1: Un FW de red Debería poder realizar las siguiente funciones (VERDADERO/FALSO):

a. Poder "escuchar" la totalidad del tráfico que deseemos analizar **(V)**
b. Estar en capacidad de "desarmar" los encabezados de cada protocolo **(V)**
c. Capacidad de descifrar tráfico con certificados digitales **(F)**
d. Soportar todos los protocolos de nivel de aplicación **(V)**
e. Poder detectar colisiones y corregirlas **(F)**
f. Capacidad de enrutamiento **(V)**

Test 2: Cuáles de las siguientes operaciones básicas sobre las cadenas que ofrece "iptables" son VERDADERAS:

a. "-t" (tabla): indica qué tabla se empleará **(V)**
b. "-i" (interfaz de entrada): mismo formato que ifconfig **(V)**
c. "-R" (redistribute) : redistribuye rutas **(F)**
d. "-z" (zone): aplica a una zona en concreto **(F)**
e. "-s, -d": dirección IP fuente o destino (Se debe aclarar máscara, ej:/24) **(V)**

Test 3: Sobre iptables (VERDADERO/FALSO):

a. Para Loguear las ocurrencias de una regla en iptables se emplea el comando "syslog" **(F)**
b. Para guardar las reglas de iptables se emplea el comando "iptables-save" **(V)**
c. Para guardar las reglas de iptables se emplea el comando "iptables -S" **(F)**
d. Para guardar también puedo ejecutar el comando "iptables-persistent" **(F)**
e. El comando **iptables -D INPUT <Number>** borra esa línea concreta **(V)**

Test 4: Sobre Firewalbuilder (VERDADERO/FALSO):

a. Firewalbulider es un proyecto de sourceforge.net **(V)**
b. Firewalbulider no soporta iptables **(F)**
c. Cuando instalas Firewalbulider te propone escenarios por defecto **(V)**
d. Las reglas de Firewalbulider no se pueden comentar **(F)**
e. Para inyectar la policy en un firewall concreto es necesaria la conexión vía telnet **(F)**

Test 5: nftables (VERDADERO/FALSO):

a. Los comandos de nftables comienzan con **nft** **(V)**
b. La configuración por defecto de nftables se almacena en /var/nft.conf **(F)**
c. El orden de nftables es tabla -> cadena -> regla **(V)**
d. El Comando --handle nos fuerza una ejecución manual **(F)**