Trabajo Práctico Nro. 5 / 2025

Materia: **Redes III**

Fecha inicio:	Calificación:
Fecha de fin:	Inicial profesor: Ing. Gustavo Pérez Ares
Unidades: VIII	Integrantes:
Tema: Seguridad de redes, ruteo y firewalling.	

Objetivo:

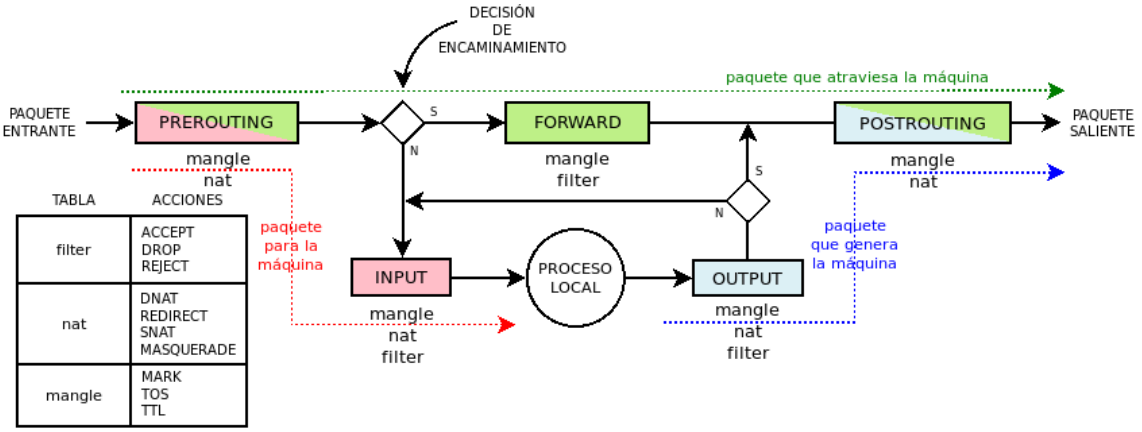
Comprender y aplicar modelos de instalaciones y configuraciones de seguridad para redes corporativas de pequeño y mediano porte.

NOTA IMPORTANTE:

1. Fedora tiene por defecto implementado para el manejo de firewalling el servicio FIREAWLLD, el cual no utilizaremos en el trabajo práctico.
2. Por razones de disponibilidad y experiencia registrada en el uso del conocido servicio IPTABLES, el cual utilizaremos en el trabajo.
3. Para preparar el sistema para su uso debemos detener y desactivar el FIREWALLD e intalar y activar el IPTABLES-SERVICE, según https://es.linux-console.net/?p=1275#gsc.tab=0 (Hay muchos otros documentos que lo explican por lo cyual el presente es solo una referencia.
4. Finalmente debemos asegurarnos la desactivación del sistema SELINUX (Ver https://www.redhat.com/es/topics/linux/what-is-selinux#:~:text=SELinux%20define%20los%20controles%20de,qu%C3%A9%20elementos%20se%20puede%20acceder.) se puede utilizar el siguiente instructivo: https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/8/html/using_selinux/enabling_and_disabling_selinux-disabling_selinux_changing-selinux-states-and-modes .

5. Completadas estas tareas queda el sistema listo para utilizar el sistema IPTABLES.

¿Cómo funciona el complejo IPTABLES?



Actividades:

- Investigar y comprender:
  - Modelos de firewalling (https://es.wikipedia.org/wiki/Cortafuegos_(inform%C3%A1tica)
  - Operatoria del complejo IPTABLES de LINUX (https://www.acens.com/wp-content/images/2014/07/wp-acens-iptables.pdf)
  - Comprender y aplicar los comandos interactivos de IPTABLES (https://www.youtube.com/watch?v=gkALH_iuaeI)
  - Elaborar scriptings de seguridad y configuración de seguridad en tiempo de booteo.(https://computernewage.com/2019/03/09/scripting-linux-bash-ejecutar-script-arranque/)
  - Comprender los logs y el sistema de loging de LINUX: /var/log, logrotate. (Ver https://www.ochobitshacenunbyte.com/2021/12/17/logrotate-administra-los-registros-de-tu-sistema-linux/#:~:text=Permite%20auto%20rotaci%C3%B3n%2C%20compresi%C3%B3n%2C%20borrado,como%20un%20trabajo%20cron%20diario.)
- Instalar y configurar los servicios de:
  - Ruteo
  - NAT
  - Proxy
  - Firewall bastión y perimetral
- Ruteo estático:
  - Simular en un entorno real la estructura de dos redes y un router de vinculación.
  - Cargar las tablas de ruteo
  - Verificar la conectividad mediante comandos ping y por conexión a puertos.
  - Reformular la red y configurar una tercera red en cascada de una de las anteriores, reconfigurar las tablas de rutas.
  - Verificar la capacidad de acceso mediante los comandos de prueba ping y telnet.
- Proxy / NAT /PAT:

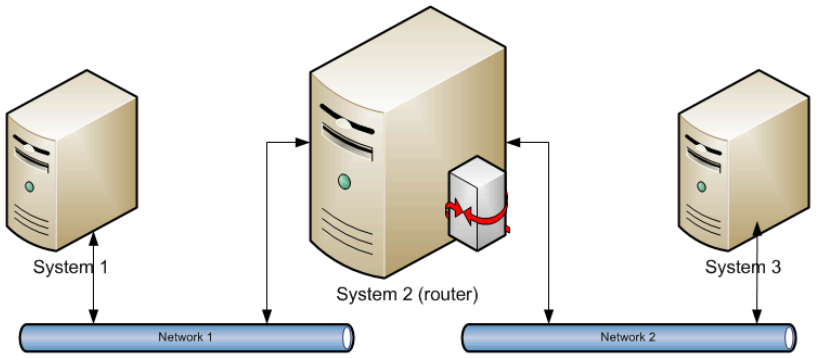
    - Configurar una topología de red que contemple un firewall (iptables), con servicio de proxy transparente (SQUID+IPTABLES), de tres zonas: LAN, INTERNET Y DMZ.
    - Investigar y proponer una configuración al firewall para esta topología, bajo las premisas básicas de este tipo de topologías.
- Bastión host:
    - Configurar un servidor LINUX, en la DMZ que brinde servicios de SSH, HTTP y HTTPS solamente, eliminando los paquetes que no cumplan con esta premisa.
    - Ídem con un servidor MS Windows 2003/2008 Server.
- Verificación de seguridad:
    - Para este fin utilizaremos la aplicación NMAP: https://es.wikipedia.org/wiki/Nmap#:~:text=Nmap%20es%20un%20programa%20de,Linux%20aunque%20actualmente%20es%20multiplataforma
    - Comandos más frecuentes: https://www.redeszone.net/tutoriales/configuracion-puertos/nmap-escanear-puertos-comandos/
    - Interfaz gráfica: https://www.redeszone.net/tutoriales/configuracion-puertos/zenmap-interfaz-grafica-escanear-puertos/
    - Instalación en Windows: https://www.youtube.com/watch?v=3LH0URMz7lU

Topologías:

A. **Host Bastión:**
    a. Sistema que publica servicios y debe segurizar sus conexiones
    b. Puertos permitidos: SSH (TCP 22), HTTP (80 TCP) y HTTPS (443 TCP)
    c. Conexiones permitidas: Solo las relativas y derivadas, no debe permitirse conexiones iniciales salientes, excepto ICMP.
    d. La regla por defecto es DROP.

B. **Ruteo de dos redes:**
    a. Generar dos instancias virtuales copia del Host Bastión, del paso A
    b. Desactivar los IPTABLES de todas las instancias
    c. Definir dos redes IP y configurar los dispositivos y rutas DEFAULT

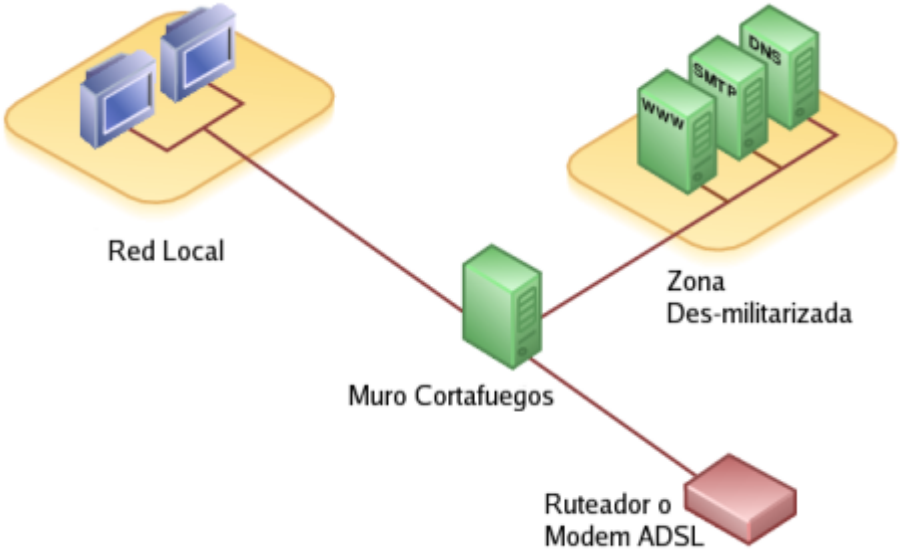
d. Activar el IP forwarding del kernel: https://www.imd.guru/sistemas/linux/ip-forwarding.html



e. Verificarla conectividad, puede ser cómodo habilitar transitoriamente el ICMP hasta completar las configuración y luego eliminar o comentar esas reglas.

C. **Firewall de dos zonas (Internet – DMZ):**
   a. Básicamente hay que convertir el Router del paso anterior en Firewall – Router, mediante comandos IPTABLES pero de reglas NAT.
   b. Debe publicarse los servicios del System 3 en las IP del System 2 usando dichos comandos y configuración:
      i. Internet a DMZ: PREROUTING Y DNAT
      ii. DMZ a Internet: MASQUERADE o POSROUTING y SNAT
      iii. Paso de paquetes entre interfaces del firewall: FORWARD.
      iv. Debe verificarse la imposibilidad de conexión directa entre las redes vía el firewall
   c. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/4/html/security_guide/s1-firewall-ipt-fwd y https://www.karlrupp.net/en/computer/nat_tutorial .

D. **Firewall de tres zonas:**
   a. Finalmente debe sumarse una tercera red de usuarios: LAN
   b. Esto implica agregar al esquema de dos zonas la red local, que debe tener acceso a la DMZ, sin usar la interfaz de Internet sino una tercera que debe agregarse al Firewall.
   c. Las opciones de permitir la salida de la LAN a Internet puede resolverse mediante el uso de MASQUERADE, **opción que debe implementarse en el TP,** o la implementación de un Proxy como SQUID. (https://devopscube.com/setup-and-configure-proxy-server/).
   d. NOTA: Solamente se recomienda el uso de SQUID cuando desea implementarse algún modo de filtrado soportado por este tipo de aplicación: https://mi.certerus.com/knowledgebase/124/iQue--es-un-Proxy-y-para-que-sirve-.html y http://recursostic.educacion.es/observatorio/web/es/software/servidores/589-elvira-mifsud



Referencias documentales:

- http://redesdecomputadores.umh.es/iptables.htm
- https://sio2sio2.github.io/doc-linux/08.redes/07.cortafuegos/index.html
- https://drive.google.com/uc?export=download&id=0B2t-Tmujl-IbcnIzMnNsVDY3Zkk (Manual de iptables)
- http://es.tldp.org/Manuales-LuCAS/doc-iptables-firewall/doc-iptables-firewall.pdf
- https://fp.josedomingo.org/seguridadgs/u03/iptables.html
- https://red-orbita.com/?p=7959

Comandos a comprender:

- Iptables, con sus opciones: https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/s1-iptables-options.html
- Telnet y SSH
- Route
- Ping
- Traceroute
- netstat

NOTA:
LO QUE HACE DEBE ENTENDERLO POR LO CUAL NO DEBE DUDAR EN RECURRIR A LAS REFERENCIAS DE BIBLIOGRAFÍA Y SITIOS WEB, ASÍ COMO EXPLORAR LA MISMA.
INSISTA HASTA COMPRENDER LO QUE HACE Y PORQUÉ ES DE ESE MODO.

Bibliografía básica:

- Building Internet Firewalls, Elizabeth D. Zwicky, Simon Cooper & D. Brent Chapman, Second Edition, June 2000 (BIF)
- LINUX: Network Administrator guide de Olaf Kirch (Second edition)(LAG)

Bibliografía complementaria:

- https://docs.fedoraproject.org/en-US/Fedora/24/html/System_Administrators_Guide/ (FED24)
- http://www.tldp.org/LDP/sag/sag.pdf (SAG)
- http://www.oreilly.com/openbook/linag2/book/ (LAG)
- IT Books, http://it-ebooks.info/
- https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/pdf/Security_Guide/Red_Hat_Enterprise_Linux-6-Security_Guide-en-US.pdf
- https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security_Guide/
- https://books.google.com.ar/books?id=rIWkyUYBsCwC&pg=PR21&lpg=PR21&dq=linux+red+hat+iptables+oficial+book&source=bl&ots=AGOMnEyAVc&sig=_6KUWvYWEoG0jzppw66yqGTIwdE&hl=es-419&sa=X&ved=0ahUKEwiZqri1yMTPAhXJQ5AKHab6BGsQ6AEISjAG#v=onepage&q&f=false

Sitios de referencia

- http://fedoraproject.org/es/
- http://www.linuxguruz.com/
- Como buscador preferido : www.google.com, especialmente por el servicio de traducción.

1. Objetivo

Comprender, configurar y aplicar técnicas de **seguridad de red, ruteo y firewalling** en entornos Linux, utilizando **iptables** como herramienta principal. A través de distintas topologías (DMZ, LAN, bastión, NAT, ruteo estático), se busca desarrollar la capacidad profesional de diseñar, implementar y verificar políticas de seguridad para redes corporativas pequeñas y medianas, asegurando control de acceso, filtrado de tráfico y protección perimetral.

2. Situación

2.1. ¿Por qué necesitamos un firewall?

Porque un firewall permite **controlar y filtrar el tráfico que entra y sale** de una red o equipo, evitando accesos no autorizados, ataques externos y movimientos laterales dentro de la red. Es esencial para proteger servicios expuestos, segmentar zonas (LAN, DMZ, Internet) y aplicar políticas de seguridad que garanticen el funcionamiento seguro de la infraestructura.

2.2. ¿Por qué elegimos iptables?

Elegimos **iptables** porque es una herramienta **madura, estable y ampliamente utilizada** en entornos Linux para el control detallado del tráfico. Permite crear reglas precisas, entender el flujo de paquetes y diseñar políticas de seguridad desde cero. Además, es ideal para aprender los fundamentos del firewalling, ya que muestra de forma explícita cómo funcionan las tablas, cadenas y reglas, algo clave para formarse como administrador o ingeniero de redes.

2.3. ¿Por qué necesitamos proteger también el OUTPUT de las reglas de firewalling?

Tal como lo hablado en clase, nuestro servidor puede ser víctima de ser usado como puente para atacar otros dispositivos. Tal es por esto que se pone en DROP las reglas de OUTPUT

3. Desarrollo

3.1. Introducción a iptables y firewalling

Un **firewall** es un componente fundamental de seguridad que controla el flujo de paquetes entre redes o dentro de un mismo sistema, aplicando políticas que permiten o bloquean tráfico según criterios definidos. Su función principal es proteger equipos y servicios limitando qué conexiones pueden establecerse, desde dónde y hacia dónde.

En Linux, uno de los mecanismos clásicos para implementar firewalling es **iptables**, un sistema que opera directamente sobre la pila de redes del kernel. Iptables utiliza una estructura basada en **tablas**, **cadenas** y **reglas**:

- **Tablas**: definen el tipo de procesamiento (filter, nat, mangle).

- **Cadenas**: representan puntos del flujo de paquetes (INPUT, OUTPUT, FORWARD, PREROUTING y POSTROUTING).

- **Reglas**: condiciones que evalúan cada paquete (IP origen/destino, puertos, protocolos, interfaces), asociadas a una acción (ACCEPT, DROP, REJECT, MASQUERADE, DNAT/SNAT).

El firewall evalúa los paquetes **secuencialmente**, aplicando la primera regla que haga coincidencia, lo que permite granularidad muy fina. Esto facilita implementar políticas como filtrado de puertos, control de rutas, NAT, protección de servicios expuestos y segmentación de redes (LAN, DMZ, Internet).

En suma, iptables ofrece un modelo poderoso y transparente para comprender cómo el kernel decide qué tráfico se permite, qué se bloquea y cómo se modifican los paquetes. Esa claridad es clave para formar criterio profesional en seguridad, ruteo y diseño de arquitecturas de red.

3.2. **Etapa A:** Configuración en Host Bastión

3.2.1. Desactivación de SELinux y firewalld

Para que no tengamos problemas, lo primero que debemos hacer es desactivar **firewalld**, que viene por defecto en linux.
En trabajos anteriores hubo que hacer esto por lo tanto solamente verificamos que este deshabilitado.
Ahora verificamos que este deshabilitado **SElinux**, el cual es un módulo de seguridad que controla el acceso a archivos y procesos. Verificamos el estado ejecutando el comando sestatus

```
joaquinvm@fedora:~$ systemctl status firewalld
○ firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled;
    Drop-In: /usr/lib/systemd/system/service.d
             └─10-timeout-abort.conf
     Active: inactive (dead)
       Docs: man:firewalld(1)
joaquinvm@fedora:~$ sestatus
SELinux status:                 disabled
```

3.2.2. Instalación y configuración de iptables

Iptables se puede usar de dos maneras:

- **Mediante scripts**
- **Como servicio/daemon**

Iptables mediante scripts:
Podemos correr iptables desde la terminal y las reglas entran en vigor **en ese momento**, pero se pierden cuando se reinicia el sistema a no ser que hagamos un save.

```
#instalacion de iptables
sudo dnf install iptables

#de esta manera indicamos reglas por esta via
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
#aqui estamos habilitando la cadena de entrada al puerto 22, (ssh)

#ver reglas actuales
sudo iptables -L -v -n

#asi guardamos las reglas
sudo service iptables save
```

Iptables como servicio
Las reglas se manejan a través de un daemon, **iptables.service**. Esto asegura que las reglas se carguen automáticamente en cada arranque.

```
#instalacion del servicio ip tables
sudo dnf install iptables-services

#arrancamos el servicio
sudo systemctl start iptables
sudo systemctl enable iptables
sudo systemctl status iptables
```

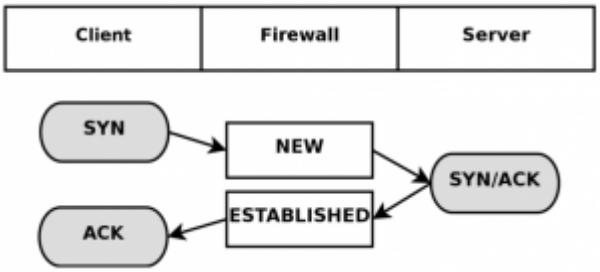
el archivo de configuración esta en **/etc/sysconfig/iptables**

La configuración inicial de políticas de entrada y salida lo hicimos en clase utilizando iptables como servicio, modificando el archivo iptables. Luego debido a que existe mayor documentación utilizando scripts, utilizaremos la otra manera

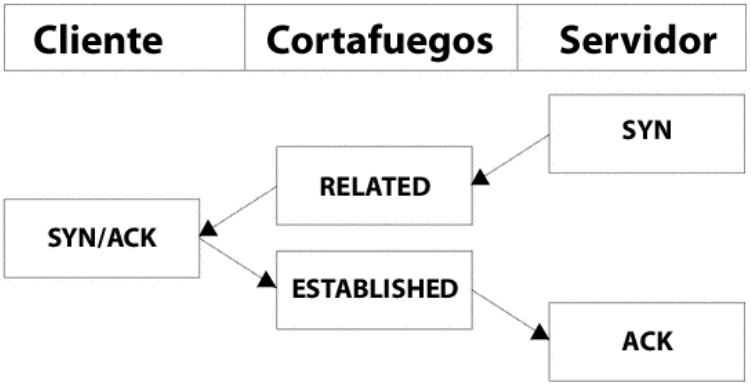
```
:INPUT DROP [0:0] #bloquea todo lo que entra
:FORDWARD DROP [0:0] #Fordward maneja la comunicaicon entra las placas
de red
:OUTPUT ACCEPT [0:0] #permite que pueda salir trafico desde nuestro
server

-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
# -A = append, se le aplica a los paquetes que entran al server
```

```
# -m modulo de coincidencia state, filtra conexiones según su estado
# -state RELATED, ESTABLISHED:
# ESTABLISHED: paquetes que forman parte de una conexión ya establecida.
#Ejemplo: abrís Firefox → hacemos una petición HTTP a Google → la
respuesta de #Google entra como ESTABLISHED.
#RELATED: paquetes relacionados con una conexión ya establecida.
#Ejemplo: cuando descargamos un archivo por FTP, el servidor abre un
canal de #datos adicional; eso se considera RELATED a la conexión FTP
inicial.
# -j jump, significa accept
-A INPUT -p icmp -j ACCEPT
# -p=protocol
#permitimos que se permita el protocolo icmp

 -A INPUT -p icmp -j ACCEPT
#permitimos tráfico desde interfaz de red localhost

-A INPUT -p tcp -m state -state NEW -m tcp -dport 22 -j ACCEPT
-A INPUT -p tcp -m state -state NEW -m tcp -dport 80 -j ACCEPT
-A INPUT -p tcp -m state -state NEW -m tcp -dport 443 -j ACCEPT
# permitimos las nuevas conexiones tcp a esos puertos

-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
# lineas que vienen por defecto, en vez de ignorar rechaza activamente
enviando una #respuesta al remitente
# --reject-with icmp-host-prohibited  la respuesta que se envía es un
mensaje ICMP "host prohibido"
```

Graficamente podemos distinguir las conexiones **NEW** y **ESTABLISHED**



Y de esta manera entendemos las **RELATED:**



Link util sobre el tema: https://rlworkman.net/howtos/iptables/spanish/iptables-tutorial.html

3.2.3. Comprobación de funcionamiento de las reglas establecidas

Una vez establecidas estas reglas, tenemos que probar que efectivamente el firewall este funcionando, lo que esperamos es lo siguiente:

- Tenemos que poder conectarnos solamente a los puertos 80,443 y 22
- Se deben permitir las conexiones mediante icmp

Para listar las reglas de firewall que estan cargadas hasta este momento con un formato detallado ejecutaremos el siguiente comando

```
iptables -L -nv
# -L lista de reglas de la tabla filter
#-n numeric, muestra direcciones IP sin resolverlos a nombres de host
#-v verboso, muestra numero de paquetes y bytes que coincidieron
```

Obtenemos lo siguiente:

```
joaquinvm@fedora:~$ sudo iptables -L -nv
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    5   546 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
    1    67 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:22
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:80
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:443
   19  3472 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           reject-with icmp-host-prohibited

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           reject-with icmp-host-prohibited
```

Ahora por ejemplo hacemos un ping desde mi maquina local y vemos lo siguiente:

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   35  3173 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
    2   120 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
    1    67 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:22
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:80
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:443
   96 16815 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           reject-with icmp-host-prohibited
```

Aumentaron los paquetes recibidos en la linea de ACCEPT icmp, lo cual es esperable.

Ahora, por que aumentaron los de la ultima linea?

Esto sucede debido a que no todos los mensajes ICMP son echo request/reply, durante un **ping**, pueden aparecer otros tipos de icmp relacionados.
Si esos no coinciden con la regla especifica de icmp caen en la regla de reject all

Ahora probamos conectarnos al puerto 22 mediante ssh. Para ver en tiempo real la cantidad de paquetes entrantes para cada regla podemos ejecutar el comando **watch -n 1 “**sudo iptbales -L -nv”

```
joaquincernik@dell:~$ ssh joaquin@192.168.0.2
The authenticity of host '192.168.0.2 (192.168.0.2)' can't be established.
ED25519 key fingerprint is SHA256:/aP6UPxtxCKRilwTFzmSPLAhJ7EknWpTir/Jg9NXxcg.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:16: [hashed name]
    ~/.ssh/known_hosts:21: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.2' (ED25519) to the list of known hosts.
Last login: Wed Sep 10 18:21:48 2025 from 172.16.3.119
joaquin@fedora:~$
```

```
  1    60 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:22
```

3.2.4. Uso de NMAP

Uno de los comandos más útiles a la hora de escanear los puertos de un dispositivo es el comando nmap. Lo ejecutamos desde el host:

```
joaquincernik@dell:~$ nmap 192.168.0.2
Starting Nmap 7.80 ( https://nmap.org ) at 2025-09-16 11:02 -03
Nmap scan report for 192.168.0.2
Host is up (0.94s latency).
Not shown: 997 filtered ports
PORT    STATE  SERVICE
22/tcp  open   ssh
80/tcp  closed http
443/tcp closed https

Nmap done: 1 IP address (1 host up) scanned in 91.33 seconds
```

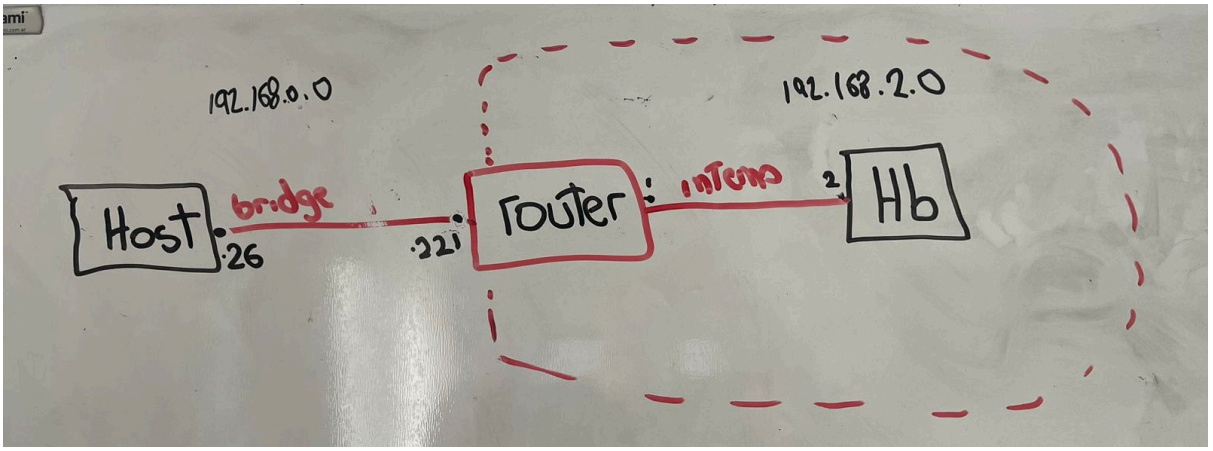
Esto nos da el indicio que hicimos nuestro trabajo correctamente, los únicos tres puertos que abrimos en el firewall son los que vemos, el puerto 22 está open debido a que esta ssh abierto del otro lado, en cambio el 80 y 443 están closed ya que todavía no pusimos a ningún servicio a trabajar en esos puertos

En el ámbito de la ciberseguridad nmap es un comando sumamente importante, debido a tiempo y alcance de este trabajo practico no puedo hablar mas sobre el, pero para mas información acerca de su uso y demas posibilidades de testeo se recomienda el siguient elink: https://nmap.org/man/es/index.html

3.3. **Etapa B:** Ruteo de dos redes

3.3.1. Requerimientos del trabajo y cómo lo voy a implementar

Lo que veníamos haciendo aquí, lo hacíamos en el host bastión, ahora vamos a trabajar sobre una clonación realizada con anterioridad, este funcionara como router, y para entenderlo realice el siguiente esquema:

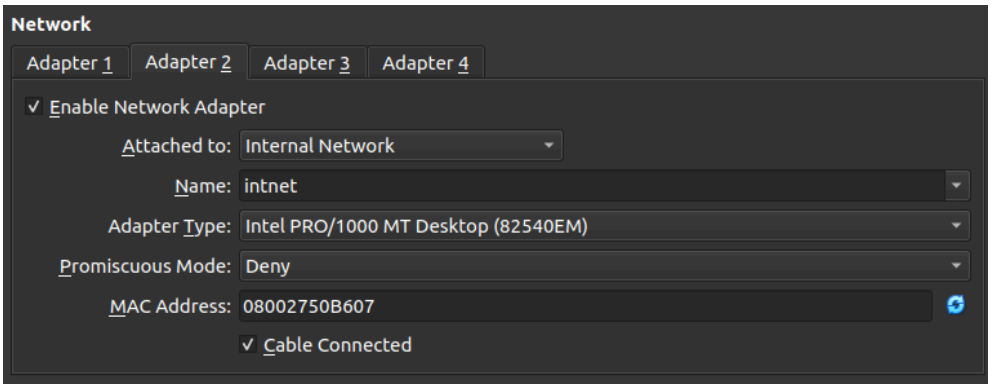


- Tenemos que cambiar el adaptador de red del **host bastión** para que solamente pueda conectarse al router y no desde afuera (internal network).
- Configurar la nueva máquina virtual **router**, agregando una interfaz de red, haciendo que una interfaz de red esté conectada al host, formando parte de su red, mediante la conexión bridge.
- Desde el host tenemos que poder hacer un **ping** al host bastión pero pasando por el router, el **objetivo es lograr que el router pueda redirigir todo su tráfico al host bastión**.

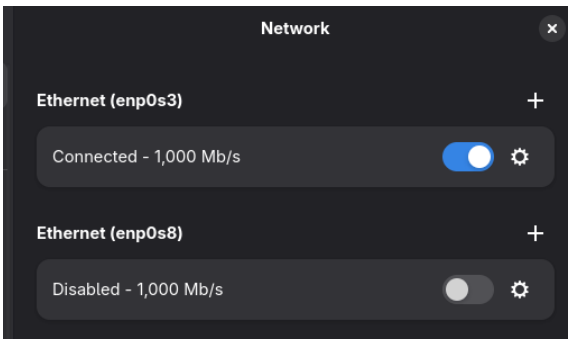
3.3.2. Configuración de red

Host bastión:
Vamos a la configuración de virtualbox y cambiamos la interfaz de red, que antes estaba en bridge a internal network.

Router:
Luego en la máquina virtual que hará de router, nos aseguramos que la interfaz de red este modo bridge. También, le agregamos un adaptador de red, clickeando adapter 2, y configurandolo en modo internal network



Para chequear que la maquina virtual reconoce la interfaz vamos a la interfaz grafica>configuración> network: (enp0s3 ->bridge, enp0s8 -> internal)



La interfaz enp0s8 no tiene salida a internet, solamente puede hablar con otras máquinas virtuales de la misma red interna. VirtualBox no levanta un servidor DHCP para redes internas, por lo tanto tenemos que asignar la IP nosotros:
IP: 192.168.2.1
Mask: 255.255.255.0

Le asignamos la IP:

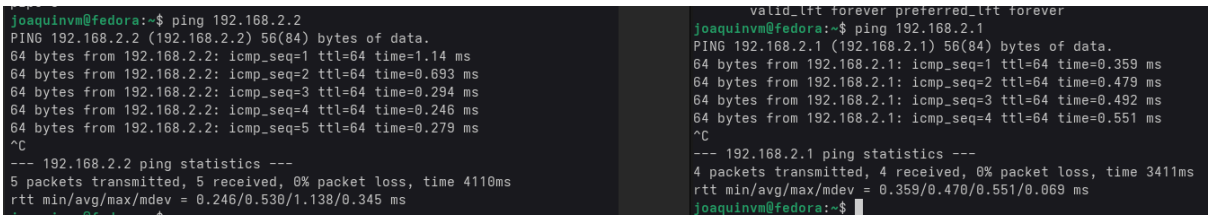
```
sudo nmcli con mod "Wired connection 2" ipv4.method manual \
ipv4.addresses 192.168.2.1/24
# nmcli es la herramienta para manejar network manager
# con mod modifica una conexión existente
# "Wired connection 2" es el nombre de la conexion, para saberlo tenemos
que ejecutar #nmcli device status
# ipv4.method manual indica que le vamos a pasar manualmente una ip
# ipv4.adresses le asignmos la ip

sudo nmcli con up "Wired connection 2"
#levantamos la conexión sin tener que reiniciar la vm
```

Controlamos que este levantada la conexión de la interfaz:

```
joaquinvm@fedora:~$ nmcli device status
DEVICE  TYPE      STATE                   CONNECTION
enp0s3  ethernet  connected               Wired connection 1
enp0s8  ethernet  connected               Wired connection 2
lo      loopback  connected (externally)  lo
joaquinvm@fedora:~$
```

Ahora le asignamos la ip al host bastión de igual manera, pero agregando la linea que aclara el default gateway: `ipv4.gateway 192.168.2.1`
IP: 192.168.2.2
Mask: 255.255.255.0
Default gateway: 192.168.2.1

Probamos haciendo un ping en ambas direcciones:



3.3.3. Configuración de ip forwarding en el router

Para que las placas de red puedan compartir conexiones entre sí tenemos que configurar un par de cosas en el router.

```
net.ipv4.ip_forward = 1
# en el archivo ubicado en /etc/sysctl.conf
# Para guardar los cambios: sudo sysctl -p
```

Chequeamos

```
joaquinvm@fedora:/etc$ sudo nano /etc/sysctl.conf
joaquinvm@fedora:/etc$ sudo sysctl -p
net.ipv4.ip_forward = 1
joaquinvm@fedora:/etc$ cat /proc/sys/net/ipv4/ip_forward
1
```

3.3.4. Configuración de red en el host

El host no sabe como llegar a la red 192.168.2.0, se lo tenemos que decir nosotros:

```
joaquincernik@dell: ~
joaquincernik@dell:~$ sudo ip route add 192.168.2.0/24 via 192.168.0.221
```

Siendo 192.168.0.221 la ip que el router tiene en la red compartida por el host
Probamos que funciona:

```
joaquincernik@dell:~$ ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=63 time=0.964 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=63 time=0.953 ms
64 bytes from 192.168.2.2: icmp_seq=3 ttl=63 time=0.959 ms
```

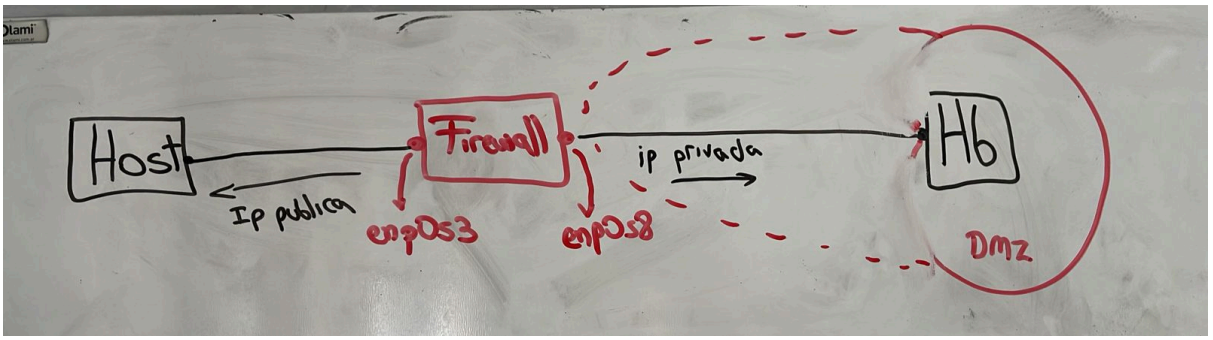
```
joaquincernik@dell:~$ traceroute 192.168.2.2
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 60 byte packets
 1  192.168.0.221 (192.168.0.221)  0.426 ms  0.378 ms  0.374 ms
 2  192.168.2.2 (192.168.2.2)  0.967 ms  0.976 ms  0.965 ms
joaquincernik@dell:~$
```

Funciona como esperábamos

3.4. **Etapa C:** Firewall de dos zonas

3.4.1. Requerimientos, configuraciones previas y como lo vamos a implementar

Nos solicitan lo siguiente

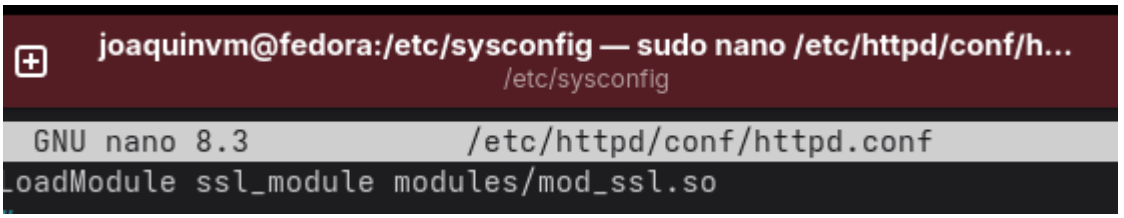


Configuraciones previas: Apache en el host bastión

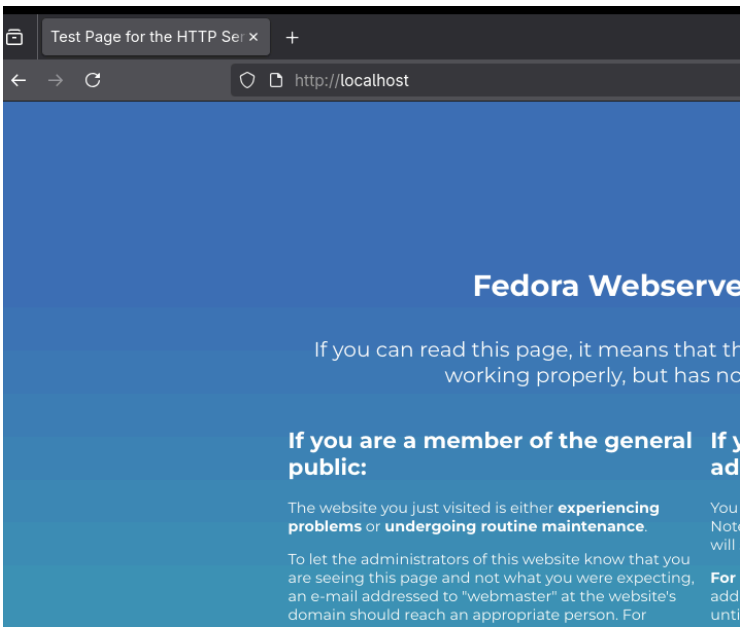
Antes de empezar tenemos que levantarnos en los puertos 80 y 443 apache en el host bastión. Para esto, momentáneamente agregamos una interfaz bridge para que se pueda conectar internet y poder descargar apache

```
sudo dnf install httpd -y

#habilitamos y arrancamos el servicio
sudo systemctl enable httpd --now

#para que el puerto 443 funcione necesitamos un  certificado
sudo dnf install mod_ssl -y

#esto genera un certificado autofirmado en
# /etc/pki/tls/certs/localhost.crt
#/etc/pki/tls/private/localhost.key
```

Luego en el archivo de configuración de apache le decimos que vamos a usar ese certificado



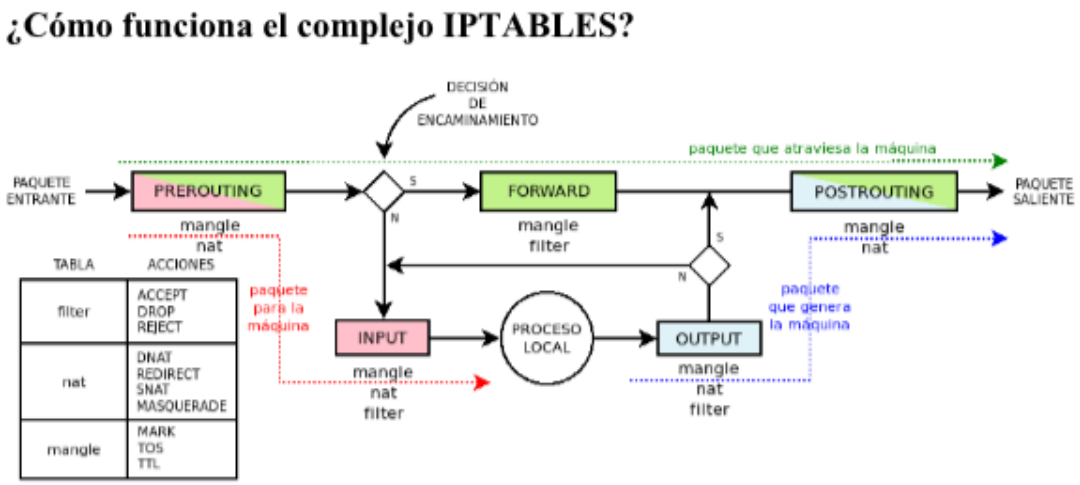
Comprobamos que funciona:



Ahora removemos la interfaz de red que creamos para esta tarea

3.4.2. Configuración en el router

Debemos tener en cuenta lo siguiente



Instalaremos iptables en el router y tenemos que logras lo siguiente

- Desactivar el forward para que desde el host no pueda conectarme al host bastion tal como decíamos antes, es decir, no tengo que poder hacer *ping 192.168.2.2,* ya que ahora vamos a usar NAT.
- El objetivo es que yo desde el host pueda acceder al servidor http levantado en el host bastión pero usando la ip publica que generaremos en el nateo del router.

Creamos las siguientes reglas en el iptables del router para que no se pueda hacer ping desde el host a la red privada:

```
iptables -A FORWARD -i enp0s3 -d 192.168.2.0/24 -j DROP
```

3.4.3. Configuración de reglas NAT, DNAT en iptables

Veamos ahora que comandos utilizaremos para realizar el nat en iptables:
En iptables, **NAT** es la tabla que permite la Traducción de Direcciones de Red, y el comando **DNAT** (Destination NAT o NAT de destino) se usa en la cadena de **PREROUTING** o **OUTPUT** para modificar la dirección IP de destino de los paquetes entrantes, permitiendo que el tráfico público sea dirigido a servidores internos en una red privada.

```
iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80  -j DNAT
--to-destination 192.168.2.2:80
```

Con esto hacemos que todos los paquetes que entren por la interfaz bridge del router, la que esta conectada a internet (host) y su puerto 80 sean reenviados al host bastion, cuyo ip es 192.168.2.2 a su puerto 80

3.4.4. Configuración de reglas NAT, MASQUERADE en iptables

Ahora tenemos que “enmascarar” lo que nos provenga de la interfaz de red conectada a la red interna, para que se cambie en el paquete de respuesta el source (que en lugar de que diga su ip original, 192.168.2.2, diga la ip pública del firewall), lo hacemos de la siguiente manera:

```
iptables -A POSTROUTING -t nat -p tcp -o enp0s8 --dport 80 -j MASQUERADE
```

Podemos hacer que sea por la interfaz o bien por su ip:

```
iptables -A POSTROUTING -t nat -p tcp -d 192.168.2.2 --dport 80 -j
MASQUERADE
```

Resumiendo nos queda el siguiente archivo de configuracion:

```
#limpiamos las reglas previas#
iptables -F
iptables -X
iptables -t nat -F

#politicas por defecto
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

#para que no me pueda conectar desde el hpst a la dmz0
#aca bloqueo solo icmp iptables -A FORWARD -i enp0s3 -d 192.168.2.0/24 -p icmp --icmp-type echo-request -j DROP
iptables -A FORWARD -i enp0s3 -o enp0s8 -p tcp -dport 80 -m state -state NEW -j ACCEPT
iptables -A FORWARD -m state -state ESTABLISHED, RELATED -j ACCEPT

#ahora tenemos que relaizar un nateo de lo que viene y redirigirlo al puerto 80 del host bastion
iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80  -j DNAT --to-destination 192.168.2.2:80
#PREROUTING-> captura el trafico antes que el kernel decida a que socket mandarlo
#-p tcp --dport 80 aplicamos solo a los paquetes tcp destinados al puerto 80
```

```
#enmascaramos las respuesstas desde la dmz
#iptables -A POSTROUTING -t nat -p tcp -d 192.168.2.2 --dport 80 -j MASQUERADE   #----> cualquiera de las dos funciona, podemos hacer de todo lo que venga de la interfaz
iptables -A POSTROUTING -t nat -p tcp -o enp0s8 --dport 80 -j MASQUERADE # o de todo lo que venga desde la ip, como arriba
```

3.4.5. Prueba de funcionamiento

3.4.5.1. Funcionamiento de redireccion

Desde el host ejecutamos un curl a la ip pública del firewall

```
Chain PREROUTING (policy ACCEPT 970 packets, 71998 bytes)
 pkts bytes target     prot opt in     out     source               destination
    5   300 DNAT       tcp  --  enp0s3 *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80 to:192.168.2.2:80
```

```
Chain POSTROUTING (policy ACCEPT 320 packets, 23689 bytes)
 pkts bytes target     prot opt in     out     source               destination
    6   360 MASQUERADE  tcp  --  *      enp0s8  0.0.0.0/0            0.0.0.0/0           tcp dpt:80
```

```
joaquincernik@dell:~$ curl 192.168.0.221:80
<!doctype html>
<html>
  <head>
    <meta charset='utf-8'>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <title>Test Page for the HTTP Server on Fedora</title>
    <style type="text/css">
      /*<![CDATA[*/

      html {
        height: 100%;
        width: 100%;
      }
        body {
  background: rgb(60,149,180);
  background: -moz-linear-gradient(180deg, rgba(60,110,180,1) 30%, rgba(60,149,180,1) 90%)  ;
  background: -webkit-linear-gradient(180deg, rgba(60,110,180,1) 30%, rgba(60,149,180,1) 90%) ;
  background: linear-gradient(180deg, rgba(60,110,180,1) 30%, rgba(60,149,180,1) 90%);
  background-repeat: no-repeat;
  background-attachment: fixed;
  filter: progid:DXImageTransform.Microsoft.gradient(startColorstr="#3c6eb4",endColorstr="#3c95b4",GradientType=1);
        color: white;
        font-size: 0.9em;
        font-weight: 400;
        font-family: 'Montserrat', sans-serif;
        margin: 0;
        padding: 10em 6em 10em 6em;
        box-sizing: border-box;

      }
```

Esto nos confirma de que esta redirigiendo correctamente la solicitud http al host bastion y este es el que responde

3.4.5.2. Funcionamiento de filtrado de acceso

Y ahora comprobamos que el host no pueda realizar ningún ping a la red interna:

```
joaquincernik@dell:~$ ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
```

```
Chain FORWARD (policy ACCEPT 203 packets, 47164 bytes)
 pkts bytes target     prot opt in     out     source               destination
   26  2184 DROP       icmp --  enp0s3 *       0.0.0.0/0            192.168.2.0/24       icmptype 8
```

```
Chain FORWARD (policy ACCEPT 203 packets, 47164 bytes)
 pkts bytes target     prot opt in     out     source               destination
   51  4284 DROP       icmp --  enp0s3 *       0.0.0.0/0            192.168.2.0/24       icmptype 8
```

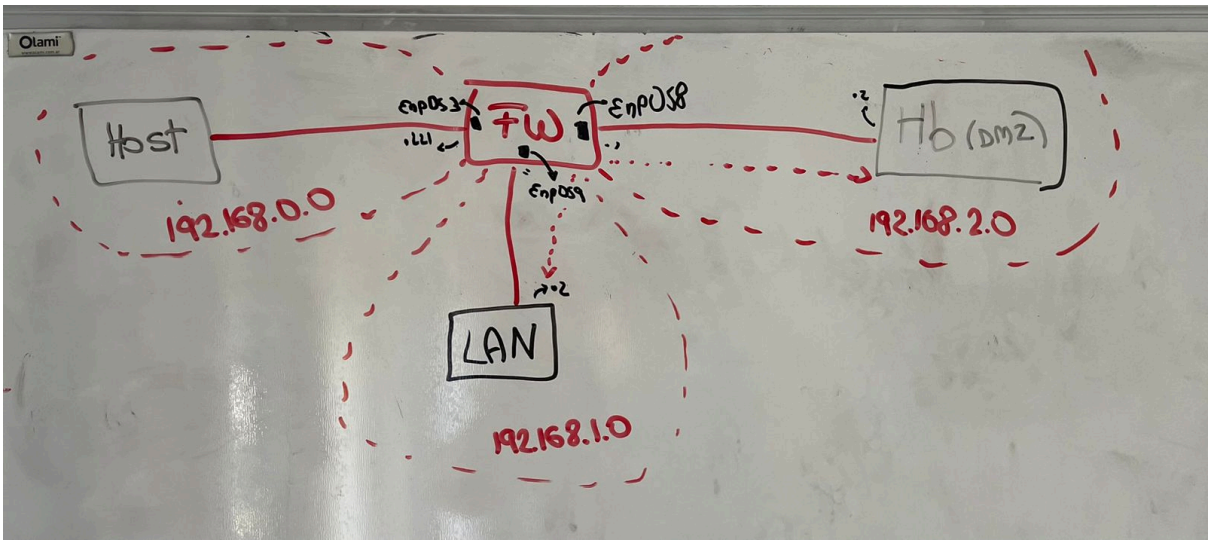
Vemos que efectivamente esta bloqueando los paquetes

Para resolver esta problemática fue util el siguiente link:
https://serverfault.com/questions/140622/how-can-i-port-forward-with-iptables

3.5. **Etapa D:** Firewall de tres zonas

3.5.1. Requerimientos y objetivos de esta etapa

Lo que buscamos es lo siguiente:



Nuestro objetivo es lograr lo siguiente:

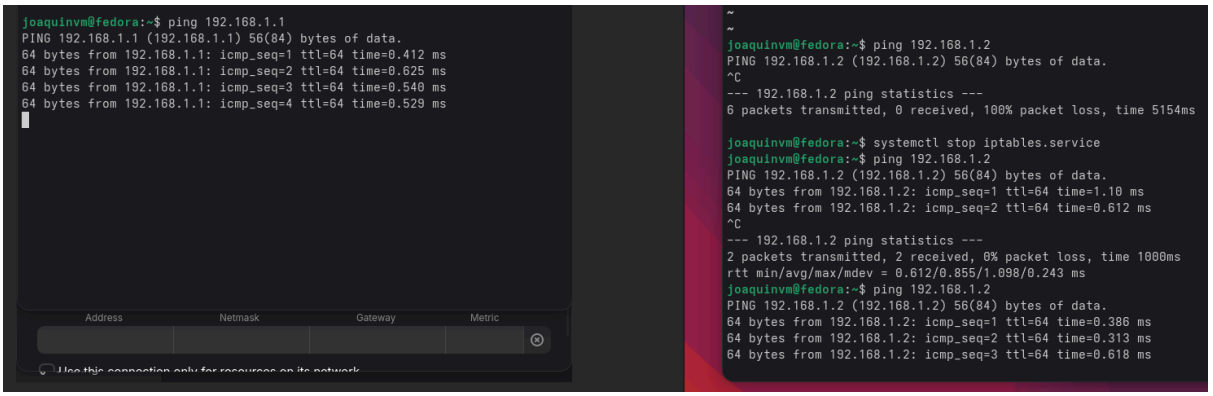
- Desde la **LAN** tenemos que poder acceder a los servicios ofrecidos por la **DMZ** y a usar internet
- No tenemos que poder acceder desde el **host** a la **LAN**

3.5.2. Conexión de LAN al router

Por un momento desactivaremos el iptables
Haremos lo siguiente:

- Agregaremos una interfaz de red nueva al router, en modo **internal network**
- Configuraremos la red 192.168.1.0

Confirmamos que ambos se ven mutuamente:

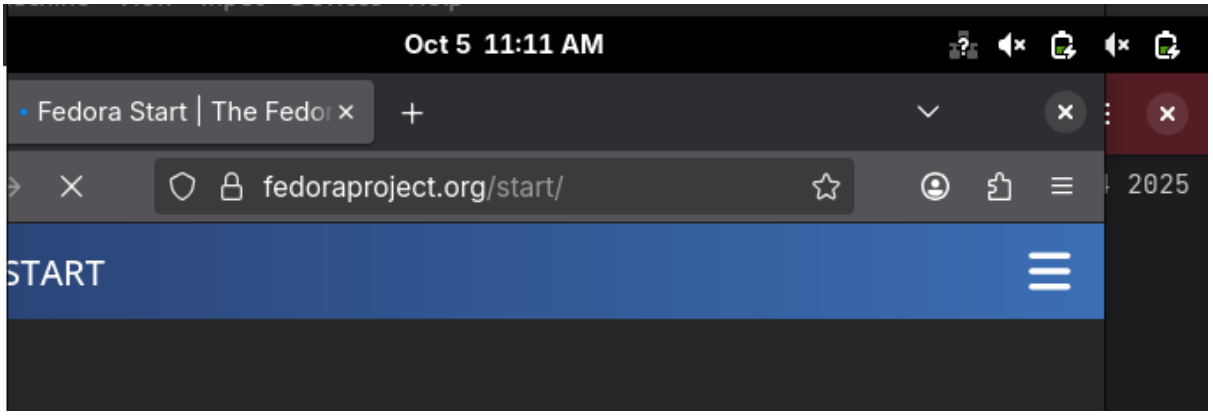


3.5.3. Permitir conexión desde la LAN a internet

```
Tenemos que lograr que la LAN tenga acceso a internet, para eso vamos a
ejecutar los siguientes comandos en iptables

#Enmascarar todo lo que salga de la interfaz conectada al host
sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

#aceptar todas las conexiones generadas por la interfaz de la LAN
sudo iptables -A FORWARD -i enp0s9 -o enp0s3 \
  -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

#aceptar todas las respuestas generadas que vengan por la interfaz del
host a la LAN
sudo iptables -A FORWARD -i enp0s3 -o enp0s9 \
  -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Hacemos la prueba de que se pueda conectar a internet por ejemplo haciendo ping a google desde un dispositivo de la LAN

```
joaquinvm@fedora:~$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  _gateway (192.168.1.1)  0.407 ms  0.316 ms  0.185 ms
 2  192.168.0.1 (192.168.0.1)  26.654 ms  26.599 ms  26.508 ms
 3  * * *
 4  * * *
 5  host88.186-153-155.telecom.net.ar (186.153.155.88)  74.922 ms  74.867
ms  74.821 ms
 6  * * *
 7  * * *
 8  * host39.181-89-51.telecom.net.ar (181.89.51.39)  35.741 ms host234.18
1-96-113.telecom.net.ar (181.96.113.234)  35.718 ms
 9  72.14.194.198 (72.14.194.198)  35.690 ms  31.621 ms  50.006 ms
10  192.178.80.111 (192.178.80.111)  49.891 ms  72.194 ms 108.170.255.29 (
108.170.255.29)  53.918 ms
11  172.253.71.11 (172.253.71.11)  54.029 ms 142.251.79.117 (142.251.79.11
7)  40.977 ms 142.250.46.111 (142.250.46.111)  40.857 ms
12  dns.google (8.8.8.8)  40.743 ms  34.769 ms  34.901 ms
```



3.5.4. Permitir conexión desde la LAN a la DMZ

Tenemos que tener conexión plena desde la LAN a la DMZ

```
#permitimos conexiones con el bit SYN, establecidas y relacionadas desde
#la lan hacia la DMZ
iptables -A FORWARD -i enp0s9 -o enp0s8 -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT
#Al reves, pero no permitimos que inicie conexiones la dmz a la LAN
iptables -A FORWARD -i enp0s8 -o enp0s9 -m state --state
ESTABLISHED,RELATED -j ACCEPT
```

Ahora probamos conectarnos desde la LAN a la DMZ
Agregamos la ruta en la tabla de ruteo de la LAN:

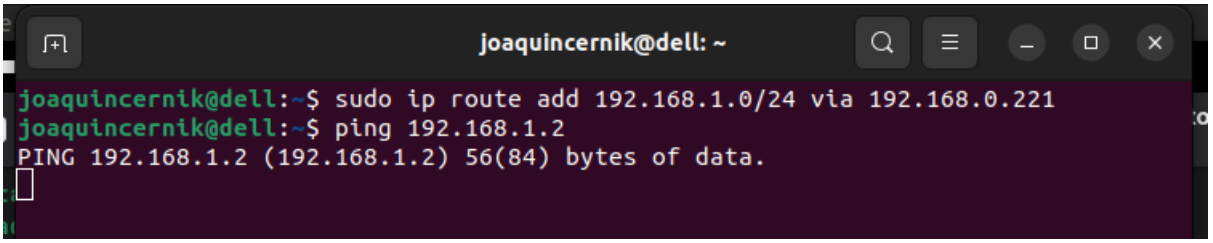
```
joaquinvm@fedora:~$ sudo ip route add 192.168.2.0/24 via 192.168.1.1
```

Realizamos un ping y vemos que funciona correctamente

```
joaquinvm@fedora:~$ ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=63 time=0.644 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=63 time=1.20 ms
64 bytes from 192.168.2.2: icmp_seq=3 ttl=63 time=0.485 ms
64 bytes from 192.168.2.2: icmp_seq=4 ttl=63 time=1.05 ms
64 bytes from 192.168.2.2: icmp_seq=5 ttl=63 time=1.31 ms
64 bytes from 192.168.2.2: icmp_seq=6 ttl=63 time=0.748 ms
64 bytes from 192.168.2.2: icmp_seq=7 ttl=63 time=1.03 ms
64 bytes from 192.168.2.2: icmp_seq=8 ttl=63 time=1.05 ms
64 bytes from 192.168.2.2: icmp_seq=9 ttl=63 time=1.07 ms
64 bytes from 192.168.2.2: icmp_seq=10 ttl=63 time=1.17 ms
```

```
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  144 12446 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  enp0s3 enp0s8  0.0.0.0/0            0.0.0.0/0           tcp dpt:80 state NEW
    0     0 ACCEPT     tcp  --  enp0s3 enp0s8  0.0.0.0/0            0.0.0.0/0           tcp dpt:22 state NEW
   27  2032 ACCEPT     all  --  enp0s9 enp0s3  0.0.0.0/0            0.0.0.0/0           state NEW,RELATED,ESTABLISHED
    0     0 ACCEPT     all  --  enp0s3 enp0s9  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
    2   168 ACCEPT     all  --  enp0s9 enp0s8  0.0.0.0/0            0.0.0.0/0           state NEW,RELATED,ESTABLISHED
    0     0 ACCEPT     all  --  enp0s8 enp0s9  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
```

Realizamos una prueba de que no se pueda conectar desde internet a la LAN



3.5.5. Script final

```
#!/bin/bash

# Limpieza de reglas previas
iptables -F
iptables -X
iptables -t nat -F

# Políticas por defecto: todo DROP
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# ---------------------------------------
# Reglas generales de FORWARD

# Permitir tráfico ESTABLISHED y RELATED
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# Permitir tráfico HTTP (puerto 80) y SSH (puerto 22) desde LAN a DMZ

iptables -A FORWARD -i enp0s3 -o enp0s8 -p tcp --dport 80 -m state
--state NEW -j ACCEPT
iptables -A FORWARD -i enp0s3 -o enp0s8 -p tcp --dport 22 -m state
--state NEW -j ACCEPT

# ---------------------------------------
# Conexión a Internet desde LAN

# Mascarado de tráfico para que LAN pueda acceder a Internet
iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

# Permitir tráfico desde LAN a Internet (NAT)
iptables -A FORWARD -i enp0s9 -o enp0s3 -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i enp0s3 -o enp0s9 -m state --state
ESTABLISHED,RELATED -j ACCEPT

# ---------------------------------------
# Reglas para el tráfico entre LAN y DMZ
iptables -A FORWARD -i enp0s9 -o enp0s8 -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT

iptables -A FORWARD -i enp0s8 -o enp0s9 -m state --state
```

```
ESTABLISHED,RELATED -j ACCEPT


# ----------------------------------------
# Reglas de NAT para redirigir puertos

# Redirigir tráfico HTTP (puerto 80) y SSH (puerto 22) hacia el host
bastión
iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j DNAT
--to-destination 192.168.2.2:80
iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 22 -j DNAT
--to-destination 192.168.2.2:22

# ----------------------------------------
# Enmascarar las respuestas desde la DMZ

# Enmascarar respuestas TCP desde el host en la DMZ hacia el puerto 80
iptables -A POSTROUTING -t nat -p tcp -o enp0s8 --dport 80 -j MASQUERADE

# ----------------------------------------
# Guardado y reinicio de servicio

# Guardar las reglas de iptables
# iptables-save > /etc/sysconfig/iptables
service iptables save
systemctl restart iptables
```

Para este trabajo se utilizaron los comandos basicos y necesarios para el funcionamiento de cada aspecto requerido, sin embargo por extension del trabajo no pude explicar explicitamente y con mucho mas detalles la funcionalidad de cada regla, para mas informacion recomiendo el siguiente link: https://www.digitalocean.com/community/tutorials/iptables-essentials-common-firewall-rules-and-commands

4. Conclusión

En este trabajo implementé de forma práctica los conceptos clave de **seguridad de redes y firewalling**, utilizando Linux como plataforma principal y **iptables** como herramienta de control de tráfico. Aprendí a diseñar y aplicar políticas de filtrado, NAT, ruteo y segmentación mediante topologías reales como **DMZ**, **LAN**, **bastión** y **firewalls perimetrales**, simulando entornos similares a los que se encuentran en redes corporativas.

5. Apreciaciones personales:

Me pareció un excelente trabajo y super útil en el sentido de tener consciencia acerca de lo peligros de internet, principios básicos de la seguridad informática como por ejemplo los permisos minimos, y tambien el hecho de ir viendo y realizando la evolución de una infraestructura fue super util