**FIREWALL**

**2020-2021**

ÍNDICE

[**Introducción**](#_heading=h.c637tgw4xv1q) **1**

[**IPTables**](#_heading=h.gmchm8apend3) **1**

[**Tablas y cadenas**](#_heading=h.mo7i2tny5vv0) **1**

[**Interfaces de máquina router**](#_heading=h.mxcf6wuavgyi) **2**

[**Reglas**](#_heading=h.mekfgwstf31) **2**

[**Configurar script**](#_heading=h.g1fl4nx5mulv) **4**

[**Cuentecitos iniciales**](#_heading=h.gdgv495cgbkb) **4**

[**NAT**](#_heading=h.912xlqwth44q) **4**

[**SNAT**](#_heading=h.lt80iabeehrm) **4**

[**DNAT**](#_heading=h.t1nge9gxvbl2) **4**

[**Filtrar tráfico**](#_heading=h.iuykra1osut2) **5**

[**Significado de cada cadena**](#_heading=h.ktbv8pbrkm6i) **5**

[**Política priorizada**](#_heading=h.r8zcs4g3oxex) **5**

[**Controlar el estado de los paquetes**](#_heading=h.rn1qbmxwfdj7) **6**

[**Tráfico a permitir**](#_heading=h.b03qk5yl4eab) **6**

[**DMZ**](#_heading=h.lmqdq71fauem) **11**

[**Bibliografía**](#_heading=h.td1qg23z3kdw) **12**

# Introducción

El firewall de red es un dispositivo situado en el perímetro de la organización, entre la red privada y el exterior. Es ideal para controlar el tráfico desde el exterior hacia el interior o desde el interior hacia el exterior.

En el núcleo de las máquinas de GNU/Linux está operativo el framework Netfilter, que permite, entre otras cosas, filtrar paquetes, redirigir el tráfico o modificar los valores de algún campo de los paquetes. Para su uso, podemos utilizar nftables o iptables (precursor de nftables). Usaremos IPTables en nuestro laboratorio por estar más extendido, pero sería una buena práctica hacer la conversión a nftables una vez terminada la configuración.

Iptables se configura a través comandos, y nos parece un buen punto de partida para empezar a aprender a configurar firewalls, ya que basta con tener una máquina GNU/Linux sin tener que instalar nada.

# Iptables

## Tablas y cadenas

En IPTables tenemos varias tablas con cadenas en las que incluimos reglas para indicar qué hacer con los paquetes. Estas son las tablas y cadenas que utilizaremos. Introducimos reglas en las cadenas:

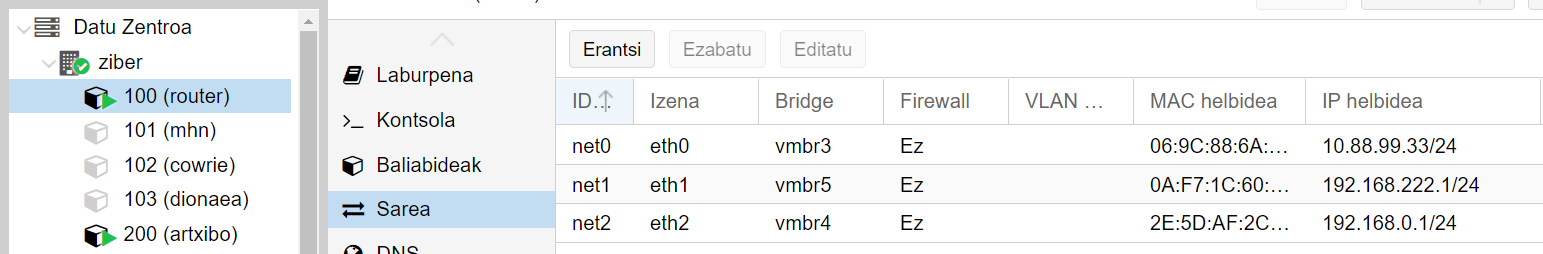
|  |  |
| --- | --- |
| **Nombre Tabla** | **Cadena** |
| filter | INPUT |
| OUTPUT |
| FORWARD |
| nat | PREROUTING |
| POSTROUTING |

En términos generales, las cadenas de la tabla *nat* se utilizarán para redirigir el tráfico, mientras que las de la tabla *filter* se utilizarán para filtrar el tráfico.

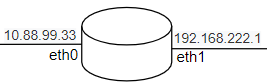
## Interfaces de la máquina *router*

La máquina router dispone de, entre otras, dos interfaces:

* *eth0:* tiene la IP 10.88.99.33 (IP pública de la organización).
* *eth1:* 192.168.222.1. Es el gateway de la red de trabajo.



Configuraremos el firewall en esta máquina. Es muy importante entender qué papel juega la interfaz en función del sentido del tráfico:



Cuando el tráfico va de dentro hacia fuera (en el sentido ← del dibujo, por tanto), *eth1* es *in* y *eth0,* *out*. Cuando el tráfico va de fuera hacia dentro (en el sentido → del dibujo, por tanto), *etho* es *in y eth1*, *out.*

## Reglas

Las reglas del IPTables tienen la siguiente estructura:

iptables [-t *<nombre\_tabla>*] *<comando>* *<nombre\_cadena>* *<parámetro-1>* *<opción-1>* *<parámetro-n>* *<opción-n> -j <acción>*

\*\*\* Si la tabla es filter, [-t filter] puede omitirse.

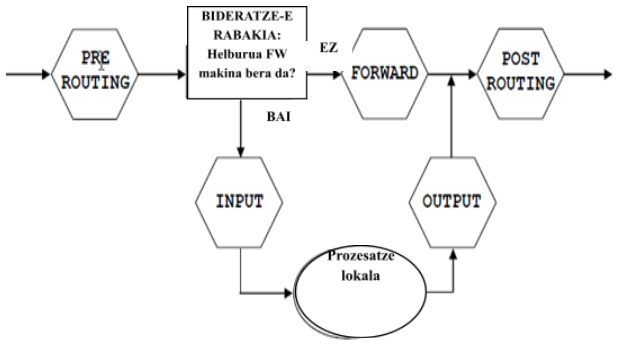
Las reglas tienen una o varias condiciones y una sola acción; si se cumplen todas las condiciones, se ejecutará la acción; una vez ejecutada la acción, el sistema no estudiará ninguna otra regla en esa cadena y seguirá adelante en el esquema IPTables.

Entre las condiciones se encuentran:

* IP de origen o de destino del paquete.
* Puerto de origen o de destino del paquete.
* Podemos poner el puerto objetivo del paquete.
* El comportamiento de una interfaz determinada (*in* u *out*)

En cuanto a las acciones principales, tenemos las siguientes:

* Drop (denegar).
* Accept (aceptar).
* Masquerade (convertir en pública la IP privada de origen).
* DNAT (convertir en privada la IP pública de destino).



Para añadir reglas al sistema, hay dos formas:

* iptables -I 🡪 \*insert\* Introduce la regla al inicio de la cadena indicada; suele utilizarse para hacer una comprobación rápida, y posteriormente se elimina.
* iptables -A 🡪 \*append\*. Añade la regla al final de la cadena indicada; suele utilizarse en los scripts porque nos interesa que las reglas se guarden en el orden escrito en el ellos.

Para terminar con este apartado, veamos un par de reglas como ejemplos:

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

* Comando: **añadir una regla al final de la cadena POSTROUTING de la tabla nat.**
* Condición: **que la interfaz eth0 intervenga como out.**
* Acción: **masquerade, es decir, cambiar la IP privada de origen por la IP pública.**

iptables -A INPUT -s 192.168.222.222 -j DROP

* Comando: **añadir una regla al final de la cadena INPUT de la tabla filter.**
* Condición: **que la IP de origen del paquete sea 192.168.222.222.**
* Acción: **Drop, prohibir el paquete.**

Podemos ir añadiendo las reglas una a una según sea necesario, pero es más cómodo crear un script, introducir en él todo lo que necesitemos y volver a ejecutarlo cuando hagamos algún cambio. Introduciremos las reglas de ese script con el parámetro -A porque nos interesa guardarlas en las cadenas en el mismo orden que en el script.

En la máquina router tienes un script denominado */etc/iptables.sh* que se ejecuta cada vez que se arranca la máquina, pero ten en cuenta que si cambias algo y no reinicias la máquina, lo modificado no será ejecutado; para ejecutar el script:

sh /etc/iptables.sh

## Configurar script

### Asuntos iniciales

Por si acaso, es recomendable limpiar todas las tablas de Iptables al inicio del script. Las dos líneas siguientes vacían todas las cadenas en las tablas nat y fitler:

iptables -F

iptables -t nat -F

Las máquinas GNU/Linux, por defecto, no redirigen tráfico. Como queremos que la máquina router realice las tareas de redireccionamiento, tendremos que activar un bit, por eso tienes la siguiente línea en el script:

echo 1 >/proc/sys/net/ipv4/ip\_forward

### NAT

#### SNAT

Cuando el tráfico va de dentro hacia fuera, la IP privada de origen debe cambiarse por la pública de la organización: Para ello es necesario añadir una regla en la cadena POSTROUTING.

Como necesitábamos que nuestras máquinas privadas tuvieran salida hacia Internet, verás esa regla escrita en el script:

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

#### DNAT

Nuestra organización ofrece una serie de servicios al Mundo. Cuando llega una solicitud a la IP pública, la máquina router debe reenviarla a la máquina privada correspondiente. Por eso tienes escritas estas dos reglas en el script:

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22 -j DNAT --to-destination 192.168.222.150

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to-destination 192.168.222.200

En la primera, cuando entre por la interfaz eth0 un paquete del protocolo TCP hacia el puerto 22 (es decir, desde el exterior hacia dentro), se le cambiará la IP de destino por la IP de la máquina (artxibo) que ofrece el servicio ssh.

Lo mismo ocurre con la segunda, pero esta vez con los paquetes hacia el puerto 80, cambiándose la IP de destino por la IP de la máquina server. Ten en cuenta que en estas prácticas no vamos a tener en cuenta el puerto 443 para facilitar las cosas.

### Filtrar tráfico

#### Significado de cada cadena

Las reglas de filtrado de tráfico deben incluirse en las cadenas de la tabla *filter.* Sin analizar en profundidad el esquema, cada cadena de la tabla*filter* se utiliza en los siguientes casos:

* *input:* cuando el destino de los paquetes es la misma máquina firewall.
* *output:* cuando el paquete ha sido creado por la máquina firewall.
* *forward:* cuando el paquete no ha sido creado por la máquina de firewall y tampoco es para él. Es decir, cuando se trata de un paquete de va de fuera hacia dentro o de dentro hacia fuera.

Por tanto, al crear una regla, debemos identificar el tipo de paquete que queremos filtrar y desde dónde y hacia dónde va.

#### Política por defecto

Todos los firewalls tienen definida una o varias políticas por defecto. Esta indica qué hacer con un paquete (aceptar o prohibir) si no hay acciones para él expresamente. Desde el punto de vista de la seguridad, y sin atender a otras cuestiones, es preferible que la política por defecto sea DROP, para que, si se nos ha olvidado controlar algún tipo de paquete concreto, sea prohibido. De lo contrario, si la política por defecto es ACCEPT, aceptaremos este tipo de paquetes incontrolados y posiblemente dejaremos una puerta abierta a posibles atacantes.

En Iptables tenemos que definir la política por defecto de cada cadena. Nosotros, siguiendo el párrafo anterior, colocaremos DROP en todas las cadenas de la tabla filter, teniendo en cuenta que hasta ahora hemos tenido ACCEPT; es decir, no hemos filtrado el tráfico, facilitando totalmente el trabajo a los atacantes.

Pon la política por defecto de FORWARD como DROP.

iptables -P FORWARD DROP

Vuelve a ejecutar el script. Si lo haces, verás que no puedes salir a Intenet desde el interior y que tus servicios no estarán accesibles desde el exterior. Es decir, has dejado la organización totalmente aislada de Internet.

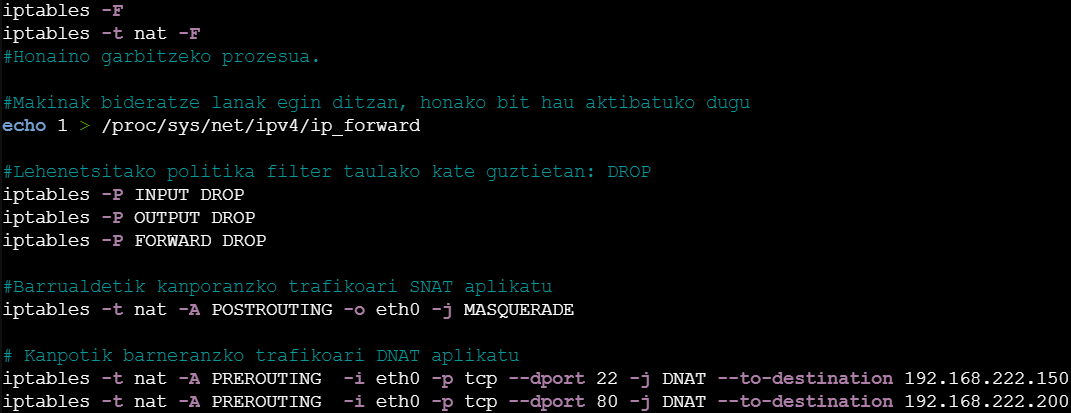
Ahora establece también las políticas por defecto de las otras dos cadenas en DROP.

iptables -P INPUT DROP

iptables -P OUTPUT DROP

Vuelve a ejecutar el script. Has dejado totalmente aislada la máquina *router*, tanto en lo que respecta al tráfico generado por ella misma como al que se dirige a ella. Puedes hacer unos PINGs en todos los sentidos para comprobarlo.

En este momento, el aspecto del script será el siguiente:



Por lo tanto, tras lo realizado, no tendríamos ningún problema de seguridad, pero, obviamente, no tendría ningún sentido que la organización estuviera totalmente aislada. Por ello, a partir de ahora, tendremos que analizar qué tipo de tráfico se admite expresamente.

#### Controlar el estado de los paquetes

Para establecer la conexión cliente/servidor, el primer paquete enviado por el cliente al servidor se encuentra en el estado *NEW.* Todos los demás paquetes de esa conexión están en estado *ESTABLISHED.*

También existe el estado *RELATED,* aunque rara vez aparece*.* Es posible que se envíe el primer paquete utilizando un protocolo concreto y se utilice otro protocolo de respuesta, de manera que esteúltimo paquete esté en estado RELATED. Loprimero que vamos a haceres permitir todos los paquetes con estados *ESTABLISHED* y *RELATED* para todas las cadenas de la tabla filter*.* A partir de ahora, cada vez que introduzcamos una regla ACCEPT, le pondremos como condición que esté en estado NEW. Así evitaremos tener que controlar las respuestas una a una cuando queremos ampliar el script con cualquier tipo de tráfico.

Introduciremos entonces en el script las tres reglas siguientes:

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

#### Tráfico a permitir

**MÁQUINA ROUTER**

Vamos a abrir agujeros en nuestro firewall. En primer lugar, aceptaremos el tráfico de la máquina *router* hacia fuera (hacia Internet). Para ello escribiremos en el script la siguiente regla:

iptables -A OUTPUT -o eth0 -m state --state NEW -j ACCEPT

* Aceptaremos todo el tráfico desde la máquina *router* hacia fuera (cadena OUTPUT y -o eth0).

También se permitirá el tráfico hacia la red local, con una excepción: en ningún caso se podrán enviar paquetes a la máquina *artxibo* desde la máquina de router.

iptables -A OUTPUT -o eth1 ! -d 192.168.222.150 -m state --state NEW -j ACCEPT

* aceptaremos todo el tráfico desde la máquina *router* hacia dentro (cadena *OUTPUT* y *-o eth1*), excepto el destinado a la IP 192.168.222.150 (*! -d*).

También controlaremos el tráfico de entrada. Se permitirá todo el tráfico desde la red local hacia el router, pero no desde el exterior hacia el router. Ten en cuenta que en el router no estamos ofreciendo servicios públicos al Mundo, por lo que, ¿para qué vamos a dejar esa puerta abierta?. Si un posible atacante realiza un ping a nuestra IP pública, por ejemplo, no nos interesa que reciba una respuesta, ya que esa simple respuesta puede ofrecerle informarle.

iptables -A INPUT -i eth1 -m state --state NEW -j ACCEPT

* Se permitirá todo el tráfico desde la red local hacia el, router (*-i eth1* y cadena *INPUT*).

**TRÁFICO DESDE LA RED LOCAL**

A continuación, se permite el tráfico hacia fuera desde la red local, teniendo en cuenta:

* No pondremos límites al tráfico ICMP.
* En cuanto a los protocolos TCP y UDP, solo dejaremos abiertos los iniciales, los "Well known ports". Además, hay otros puertos conocidos, pero para esta práctica bastará con esos. Podríamos no poner límites a los puertos, pero, ¿para qué dejar abierto lo que no necesitamos? Nunca sabemos cómo nos puede llegar un ataque, y piensa que los atacantes buscan constantemente nuevas debilidades.

Deberemos añadir las siguientes reglas:

iptables -A FORWARD -o eth0 -p icmp -m state --state NEW -j ACCEPT

iptables -A FORWARD -o eth0 -p tcp --dport 1:1023 -m state --state NEW -j ACCEPT

iptables -A FORWARD -o eth0 -p udp --dport 1:1023 -m state --state NEW -j ACCEPT

* Aceptaremos todo el tráfico icmp hacia fuera de la red local (cadena *FORWARD* y *-o eth0*).
* En los protocolos TCP y UDP, sólo dejaremos abiertos los puertos 0-1023 (los iniciales, algunos de los "Well known ports").

**TRÁFICO HACIA LA RED LOCAL DESDE INTERNET**

Nuestra red ofrece al Mundo dos servicios: el servicio SSH y el servicio web.

En cuanto al servicio SSH, ya vimos en una práctica anterior el daño que nos puede hacer tenerlo abierto sin seguridad. El servicio SSH no está pensado para que sea público, es decir, no tiene sentido que sea accesible para cualquiera, ya que su objetivo es que sólo nosotros, los autorizados, accedamos solos a la máquina remotamente. Por lo tanto, desde el punto de vista del firewall, el primer paso puede ser limitar los intentos de ssh a una o varias IPs de origen:

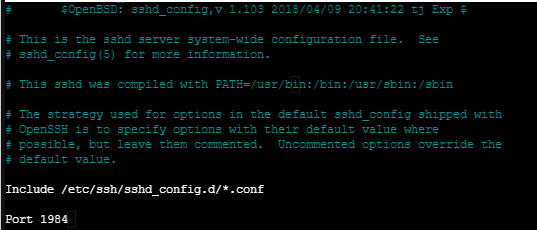
iptables -A FORWARD -i eth0 -p tcp --dport 22 -s 10.88.99.66 -m state --state NEW -j ACCEPT

* Permitiremos los paquetes desde Internet (FORWARD y -i eth0) hacia el puerto 22 (el servicio está en una máquina de la red local, recuerda que hemos redirigido las peticiones) siempre que la IP de origen (la del solicitante) sea 10.88.99.66.

El servicio ssh estará accesible desde la máquina *kanpokoa*, pero no desde la *kali*. Por hacer más pruebas, si quieres, asigna otras IPs públicas a la máquina *kali*; no debería permitirse el acceso desde ninguna de ellas.

Sin embargo, como concluimos con el ataque ssh, es recomendable cambiar el puerto por defecto del servicio por uno conocido sólo por nosotros mismos. Para ello, descomentamos el parámetro "Port" y asignamos el nuevo valor:

nano /etc/ssh/sshd\_config



Una vez guardados los cambios y cerrado el fichero, deberemos reiniciar el servicio ssh:

service sshd restart

Ahora acude de nuevo a la máquina *exterior* y trata de conectarte al servicio ssh:

ssh admin@10.88.99.33



Inténtalo de nuevo, pero esta vez especificando el puerto 1984:

ssh -p 1984 admin@10.88.99.33

No conseguirás conectarte. ¿Por qué? Tendremos que volver a tocar el firewall. tenemos redirigido el puerto 22 (y permitido el tráfico hacia él), y no el 1984. Por lo tanto:

~~iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22 -j DNAT --to-destination 192.168.222.150~~

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 1984 -j DNAT --to-destination 192.168.222.150

~~iptables -A FORWARD -i eth0 -p tcp --dport 22 -s 10.88.99.66 -m state --state NEW -j ACCEPT~~

iptables -A FORWARD -i eth0 -p tcp --dport 1984 -s 10.88.99.66 -m state --state NEW -j ACCEPT

Vuelve a intentar conectarte desde la máquina *kanpokoa* utilizando el puerto 1984:

ssh -p 1984 admin@10.88.99.33

Como verás, puedes conectarte.

A diferencia del servicio SSH, generalmente nos interesa que el servicio web sea público (salvo en el caso de una intranet), es decir, que sea accesible desde cualquier lugar de Internet. Eso es lo que vamos a hacer, aceptando todas las solicitudes hacia el servicio web, vengan de donde vengan.

Para ello introduciremos en el acript la siguiente regla:

iptables -A FORWARD -i eth0 -p tcp --dport 80 -m state --state NEW -j ACCEPT

* Permitiremos los paquetes desde Internet (FORWARD y -i eth0) hacia el puerto 80 (el servicio está en una máquina de la red local, recuerda que hemos redirigido las peticiones).

Navega desde la máquina kanpokoa a la IP 10.88.99.33 y podrás ver la página web de nuestra organización. Lo mismo desde la máquina *kali*.

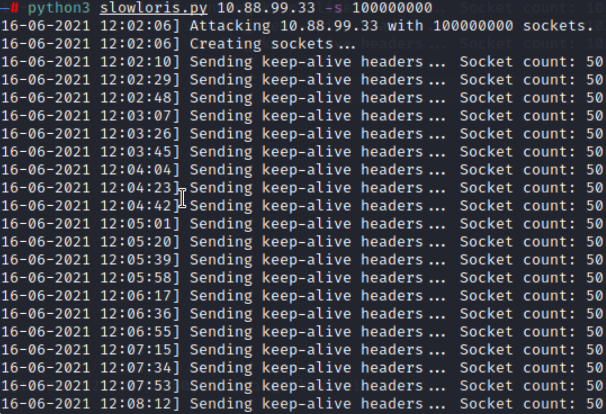
No obstante, recuerda que nuestro servicio web presentaba una vulnerabilidad al ataque DoS. Si ejecutas el ataque DoS con Slowloris desde la máquina *kali,* verás que el servicio no responde correctamente. En lo concerniente al firewall, plantearemos una solución sencilla y simple: limitar el número de conexiones paralelas enviadas desde una IP determinada. Nosotros las limitaremos a 50.

**¡Cuidado! Recuerda que el sistema lee secuencialmente las reglas de las cadenas en el orden guardado, por lo que la regla que tienes aquí debajo deberá estar por encima de la anterior en el script; de lo contrario, nunca se ejecutará:**

iptables -A FORWARD -p tcp --dport 80 -m connlimit --connlimit-above 50 -j DROP

* Desde Internet (FORWARD y -i eth0) 80 atakaranzos (el servicio está en una máquina de red local, recuerda que hemos redirigido los pedidos) prohibiremos los intentos si hay más de 50 solicitudes paralelas del IP original en ese momento.
* Prohibiremos los paquetes desde Internet (FORWARD y -i eth0) hacia el puerto 80 (el servicio está en una máquina de la red local, recuerda que hemos redirigido las peticiones) si hay más de 50 peticiones simultáneas desde la IP de origen indicada.

Vuelve a poner en marcha el ataque en la máquina *kali*. Por un lado, verás que el número de sockets no pasa de 50.



Por otro lado, podrás navegar tanto desde la máquina *kali* como desde la máquina *kanpokoa* sin problemas.

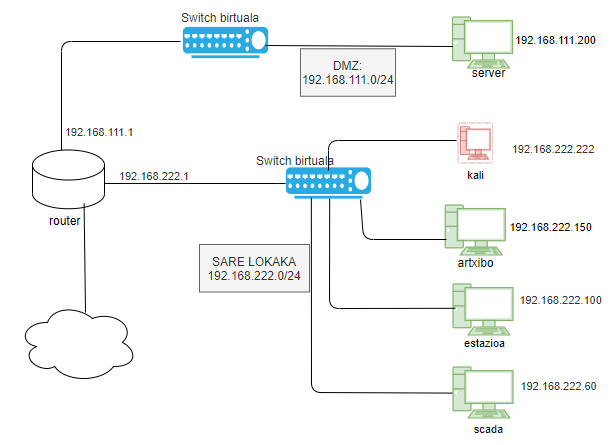
Puede ser un buen momento para que pruebes todo lo que has hecho. Algunas pruebas que puedes hacer son:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Creador del paquete** | **Objetivo** | **Acción** | **¿Debe autorizarlo?** | **¿Lo permite?** |
| *kanpokoa* | IP pública → artxibo | ssh -p 1984 10.88.99.33 | Sí | Sí/No |
| *kanpokoa* | IP pública → server | w3m 10.88.99.33 | Sí | Sí/No |
| *kanpokoa* | IP pública | ping 10.88.99.33 | No | Sí/No |
| *router* | kanpokoa | ping 10.88.99.66 | Sí | Sí/No |
| *router* | Server | w3m 192.168.222.200 | Sí | Sí/No |
| *router* | artxibo | ssh 192.168.222.150 | No | Sí/No |
| *router* | kanpokoa | ssh -p 1555 10.88.99.66 | Sí | Sí/No |
| *artxibo* | tknika.eus | w3m www.tknika.eus | Sí | Sí/No |
| *artxibo* | Gateway | ping 192.168.222.1 | Sí | Sí/No |
| *server* | kanpokoa | ping 10.88.99.66 | Sí | Sí/No |
| *server* | kanpokoa | ssh -p 1555 10.88.99.66 | No | Sí/No |
| *kali* | IP pública → artxibo | ssh -p 1984 10.88.99.33 | No | Sí/No |
| *kali* | IP pública → server | Navegar a 10.88.99.33. | Sí | Sí/No |
| *kali* | IP pública | ping 10.88.99.33 | No | Sí/No |
| *kali(cambiar IP) (10.88.99.111)* | IP pública → artxibo | ssh -p 1984 10.88.99.33 | No | Sí/No |
| *Kali (cambiar IP) (10.88.99.111)* | IP pública → server | Navega a 10.88.99.33 | Sí | Sí/No |
| *Kali (cambiar IP) (10.88.99.111)* | IP pública | Ping 10.88.99.33 | No | Sí/No |

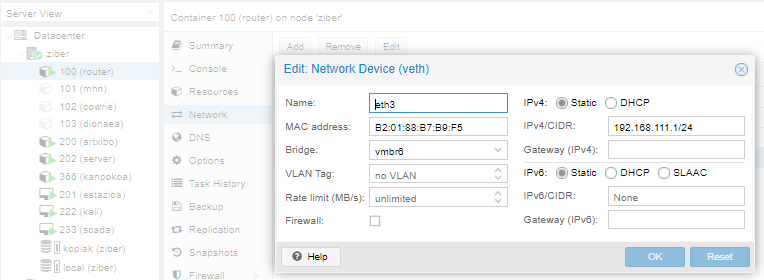
Tenemos configurado un firewall sencillo y adaptado a nuestra estructura, que podríamos calificar de bastante efectivo. En cualquier caso, hay que tener en cuenta que la red local no ha sido segmentada en absoluto. Piensa que será fácil que el atacante vaya contra el servidor web para dañar de alguna manera otros dispositivos de la red (entre ellos, la importante máquina *artxibo*, llena de información confidencial). Por ello, conviene aislar a los servidores públicos en otra red.

#### DMZ

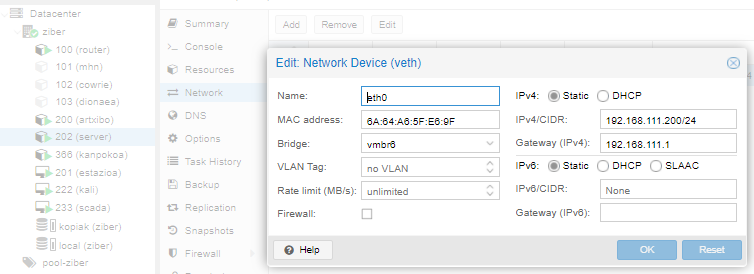
La DMZ es una zona especial de nuestra estructura en la que contaremos con una red independiente en la que estarán las máquinas que queremos dejar abiertas a las demandas de Internet, es decir, las máquinas que ofrecen servicios públicos. ¿Por qué? Si un atacante consiguuiera atacar a esos servidores públicos, podría extenderlo a otros dispositivos de la red en la que se encuentra. Se consideran servicios públicos *ftp*, *mail* y *web*, entre otros. Entre esos, nosotros sólo tenemos uno: un servicio web. Crearemos la red 192.168.111.0/24 como DMZ, y ahí situaremos en servidor web.



Lo primero que haremos será crear una nueva interfaz de red en la máquina *router*. Le asignamos la IP 192.168.111.1/24 y la colocamos en el "bridge" *vmbr6*.

****

Ahora vamos a mover la máquina *server* a esa nueva red. Le asignamos a su interfaz la IP 192.168.111.200, escribimos como gateway 192.168.111.1 y la ponemos en el "bridge" *vmbr6*:



A continuación, comprobamos que la máquina *router* tiene salida hacia Internet.

No basta con haber creado una nueva red si no controlamos el tráfico entre todas las redes. ¿En qué consiste conceptualmente una DMZ?

* El tráfico de Internet hacia la DMZ estará permitido. En nuestro caso, sólo tenemos que abrir el puerto 80. ¿Para qué dejar abiertos los puertos innecesarios?
* Todo el tráfico desde Internet hacia la red local estará prohibido. En nuestro caso, excepcionalmente, autorizaremos el tráfico hacia el servidor ssh, respetando las condiciones de establecidas anteriormente.
* Queda prohibido todo el tráfico desde la DMZ hacia la red local. De lo contrario, una vez atacada la máquina de la DMZ, el cracker podría dirigir el ataque hacia la red local. Por lo tanto, la creación de una nueva red no habría servido de mucho.
* El tráfico desde la red local hacia la DMZ estará permitido, pero sólo en el puerto 80 ¿Para qué dejar abiertos los puertos innecesarios?

# Bibliografía

**Losada, Iñigo** (2015). Ibex Servicios Informáticos. "Configuración de un cortafuegos bajo Linux".

[UBUNTU - IPTables HowTo](https://help.ubuntu.com/community/IptablesHowTo)

# 

# 



