Practica 4

ILLIA NECHESA | PABLO OSPINO

Seguridad en Sistemas Informáticos | 4º Curso | 16-12-2019

Seguridad perimetral

Introducción

Para comenzar creando la arquitectura definida en la práctica vamos a reutilizar una de las máquinas virtuales que creamos en la práctica anterior. De la práctica 3 clonamos la máquina virtual base 4 veces generando así las nuevas 4 máquinas que necesitamos para esta práctica.

Además, se nos pide que configuremos 2 adaptadores de red host-only que estén habilitados como servidores DHCP. Para ello vamos a preferencias de virtualbox, apartado network y pestaña host-only. En esta ventana añadimos un nuevo adaptador, habilitamos DHCP y le asignamos otro rango de ips que pertenecen a otra subred.

VM1

Configuramos la máquina 1 definiendo la interfaz de tipo NAT y descargando todos los paquetes necesarios. Una vez descargados la apagamos e iniciamos con la interfaz configurada con el adaptador 1.

Ejecutamos los siguientes comandos para descargar los paquetes necesarios.

- **sudo apt-get install ssh**
- **sudo apt-get install apache2**
- **sudo apt-get install w3m**

VM2

Realizamos el mismo proceso que en VM1. Activamos el forwarding y descargamos iptables.

Para activar el forwarding accedemos al fichero /etc/sysctl.conf. Mediante el comando `sudo nano /etc/sysctl.conf` accedemos al editor de texto nano y descomentamos la linea net.ipv4.ip_forward = 1. Sin embargo, realizando esto solamente hemos cambiado la configuración y para que realmente se aplica es necesario que se reinicie el sistema. Para ello ejecutamos el siguiente comando `sysctl -p /etc/sysctl.conf` .

Para instalar iptables realizamos lo siguiente:

- **sudo apt-get update**
- **sudo apt-get install iptables**

Comprobamos que se ha instalado correctamente:

- sudo iptables -L -v

  Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target prot opt in out source destination

  Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target prot opt in out source destination

  Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target prot opt in out source destination

VM3

Con el mismo proceso de VM1 descargamos **ssh**.

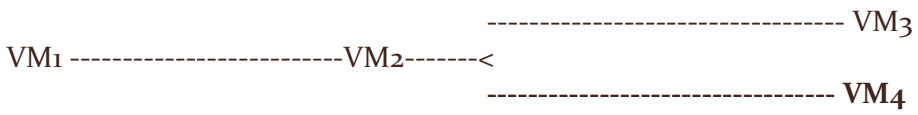
VM4

Con el mismo proceso que VM1 y VM3 descargamos el **apache2**.

Tras haber descargado todo lo necesario iniciamos todas las máquinas

VM1 con una interfaz el adaptador 1.
VM2 con dos interfaces, una con cada adaptador (HOST-ONLY).
VM3 con una interfaz el adaptador 2.
VM3 con una interfaz el adaptador 2.

Al iniciar todas hay ips que se repiten por lo que cambiamos las ips de algunas de ellas para que no coincidan.



vm1 se queda como está en la red 192.168.56.101
vm2 el etho 192.168.56.102 y creamos otra interfaz eth1 con la ip 192.168.57.103
vm3 en el 192.168.57.101
vm4 en el 192.168.57.102

Una vez hemos configurado todas las ips comprobamos que se pueden conectar entre ellas. Para ello realizamos pings entre las diferentes máquinas. Intentamos acceder desde VM3 a VM1 mediante el comando:

- ping **192.168.56.101.**

Esto falla, nos dice que no se puede llegar al host. Esto es debido a que nos falta configurar los gateways por defecto de todas las máquinas para que vayan a VM2. Para ello ejecutamos el siguiente comando.:

En VM3 Y MV4 hacemos lo siguiente:

- **sudo ip route add default via 192.168.57.103**

VM1 hacemos

- **sudo ip route add default via 192.168.56.102**

De esta forma hemos configurado la ip de VM2 para que sea el gateway de las máquinas de cada red con las ips correspondientes a cada red.

Una vez tenemos todas las ips configuradas ya podemos empezar con los ejercicios.

IPTABLES

Iptables es una utilidad de CLI que permite configurar las reglas de los cortafuegos (firewalls). En este caso vamos a configurar a VM2 como un cortafuegos pues es la única máquina en la que hemos descargado la utilidad de iptables.

Mostramos todas las reglas que tenemos a la hora de iniciar la máquina.

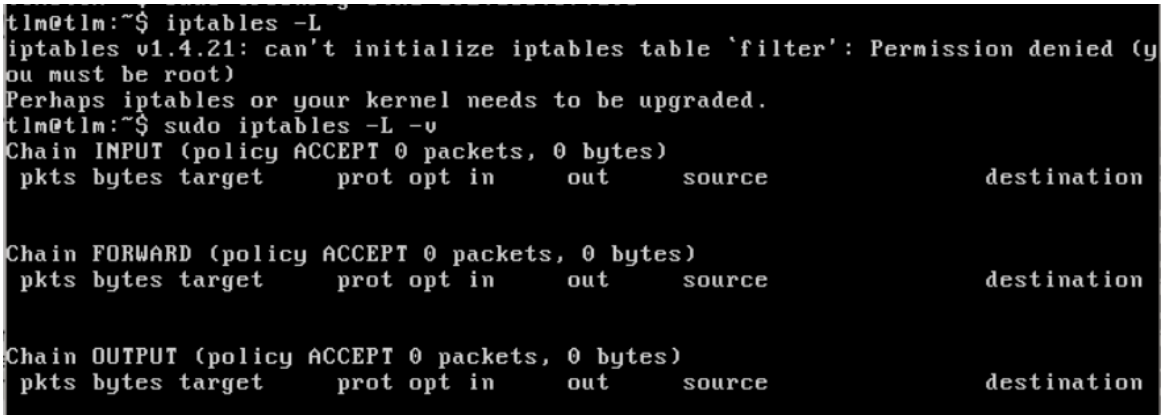
- **sudo iptables -L -v**

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination

Como vemos no tenemos ninguna definida por ahora.



Investigación sobre IPTABLES

Aprendemos a usar la herramienta:

Configurar reglas por defecto para cada cadena:

Aceptar por defecto

- **iptables -P INPUT ACCEPT # política por defecto: aceptar**
- **iptables -P FORWARD ACCEPT # política por defecto: aceptar**
- **iptables -P OUTPUT ACCEPT # política por defecto: aceptar**

Rechazar todos los paquetes por defecto

- **iptables -P INPUT DROP # política por defecto: descartar**
- **iptables -P FORWARD DROP # política por defecto: descartar**
- **iptables -P OUTPUT DROP # política por defecto: descartar**

Borrar todas las reglas:

- **iptables -F # borrar reglas**

Sin embargo, esto no nos borra las reglas por defecto que hayamos definido. Para realizar esto debemos ejecutar:

- **iptables -X # borrar cadenas**

Resetear todas las estadísticas a cero

- **iptables -Z # estadísticas a cero**

Añadir reglas.

- **Iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT**

En el ejemplo de arriba, vamos a añadir una nueva regla. Vamos paso a paso.
El -A significa añadir al final una regla, es decir, si ya tenemos 2 reglas en el input, por ejemplo, y queremos añadir otra al final, ponemos **-A**. Sin embargo, si quisiésemos añadirla al principio, pondríamos **-I**, que significaría insertarla en primera posición. Esto es muy importante, ya que luego si queremos borrar alguna regla, el orden importa. Luego ponemos **INPUT**, **FORWARD**, O **OUTPUT,** dependiendo de si nos llega el paquete a nosotros (Input), o si el paquete pasa por nosotros, pero no va dirigido a nosotros, es decir, solo reenviamos (Forward), o si el paquete es generado por nosotros y enviado a otro (Output). Luego ponemos **-s/-d** dependiendo de si nos queremos referir al source o al destino. El **-j** indica el jump a la acción que vamos a ejecutar en el caso de que esa regla se cumpla. En este caso **ACCEPT** que tal y como su nombre indica, aceptamos. Sino podríamos poner **DROP,** para rechazar.

- **Iptables -A INPUT -p tcp –dport 5160 -j ACCEPT**

  En este caso, por ejemplo, con el **-p** indicamos el protocolo, en este caso tcp, y con **-dport,** que es el puerto destino.

- **Iptables -D INPUT 3**

  En este caso lo que indicamos es que queremos borrar (**-D**) la regla de la posición 3. Es por ello muy importante listar antes todas las reglas, y asegurarnos de cuál es la posición de la regla que queremos borrar.

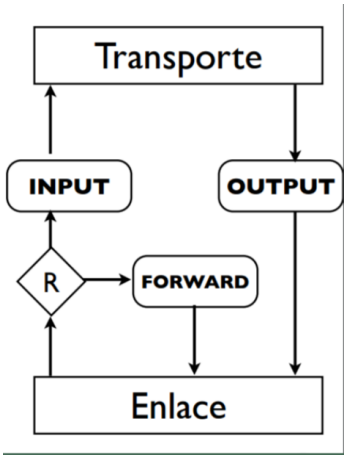
  https://elbauldelprogramador.com/20-ejemplos-de-iptables-para-sysadmins/

- **sudo sysctl -p /etc/sysctl.conf**

etho no estaba definido, porque se nos había olvidado. Forward activado. Ahora ya funciona todo correctamente

Prueba a bloquear aplicaciones en una política de ACCEPT por defecto. Por ejemplo, que no se puede hacer ssh desde una de sus máquinas, pero si desde las demás.

Como básicamente VM2 hace la función de router solo nos interesa definir reglas en la cadena de FORWARD. Basándonos en la siguiente imagen:



Definimos por defecto aceptar los paquetes: **iptables -P FORWARD ACCEPT**
Bloqueamos ssh para la máquina VM3.

iptables -A FORWARD -s 192.168.57.102 -p tcp --dport 22 -j DROP

Además iniciamos el servicio de ssh de la máquina VM1.

```
tlm@tlm:~$ sudo iptables -P FORWARD ACCEPT
tlm@tlm:~$ sudo iptables -A FORWARD -s 192.168.57.102 -p tcp --dport 22 -j DROP
tlm@tlm:~$ _
```

De esta forma añadimos en la cadena FORWARD la regla de bloquear todo paquete que venga de la ip 192.168.57.101 y tenga como destino el puerto 22 (Puerto SSH).

Una vez realizado esto no podemos acceder desde VM3 pero sí que podemos acceder desde VM4 al servicio de ssh que está corriendo en VM1.

Prueba a dejar pasar aplicaciones en una política de DROP por defecto. Por ejemplo, a solo dejar pasar el web hacia una máquina.

De la misma forma solo actuamos sobre la cadena FORWARD.
Definimos por defecto bloquear los paquetes: **iptables -P FORWARD DROP.**
Arrancamos el servidor de apache en VM1 en el puerto 80.
Definimos que VM3 si que pueda acceder a la web.

iptables -A FORWARD -s 192.168.57.101 -dport 80 -j ACCEPT

```
tlm@tlm:~$ sudo iptables -P FORWARD DROP
tlm@tlm:~$ sudo iptables -A FORWARD -s 192.168.57.101 -p tcp --dport 80 -j ACCEP
T
tlm@tlm:~$ _
```

De esta forma no somos capaces de acceder a la web desde la máquina VM4 pero sí desde la máquina VM3 ya que lo hemos definido explícitamente.

SSH TUNELING.

Como sabemos por la teoría somos capaces de generar 2 tipos de túneles con ssh. Estos dos tipos son local port forwarding y remote port forwarding. Una de las utilidades de los túneles es que nos permite esquivar los firewalls. Así que como hemos definido unas reglas en el firewall de VM2 vamos a intentar esquivarlas y poder realizar las acciones que en un principio hemos bloqueado.

Comenzamos por local port forwarding.

Local port forwarding nos permite enrutar todo el tráfico que recibe un puerto local al puerto del destino que definamos. El comando tiene la siguiente estructura:

> ssh usuario@host -L puertolocal:destino:puerto

Vamos a intentar que la máquina VM4 sea capaz de conectarse al servidor WEB. Para ello es necesario que podamos conectarnos al servidor ssh desde VM3 por lo que eliminamos la regla que habíamos definido.

> iptables -D FORWARD 1

Además es necesario que añadamos una nueva regla que permita al servicio de ssh conectarse ya que ahora mismo la política por defecto es DROP.

> iptables -A FORWARD -s 192.168.57.102 -dport 22 -j ACCEPT

```
tlm@tlm:~$ sudo iptables -A FORWARD -s 192.168.57.102 -p tcp --dport 22 -j ACCEP
T
tlm@tlm:~$ _
```

Ya podemos realizar el túnel.

> ssh ssi29@192.168.56.101 -L 8000:192.168.57.102:80

De esta forma redirigimos todo el tráfico que recibe la máquina VM3 en el puerto 8000 al servidor web que corre en el puerto 80 del servidor. De esta forma, si nosotros nos conectamos desde la máquina VM4 al puerto 8000 de la máquina VM3 seremos capaces de acceder al servidor web.

Por otro lado tenemos el remote port forwarding.

Remote port forwarding nos permite enrutar todo el tráfico que recibe un puerto del servidor de ssh al puerto del destino que definamos. El comando tiene la siguiente estructura:

> ssh usuario@host -R puertoremoto:destino:puerto.