M6 – P3: Segmentación de Red con VLANs en Entorno Virtual

Simular una **segmentación de red por VLANs** en VMware, usando **redes virtuales separadas** (subredes distintas) y un **router Linux** que actúa como cortafuegos, para observar cómo se controla y limita el tráfico entre zonas de red.

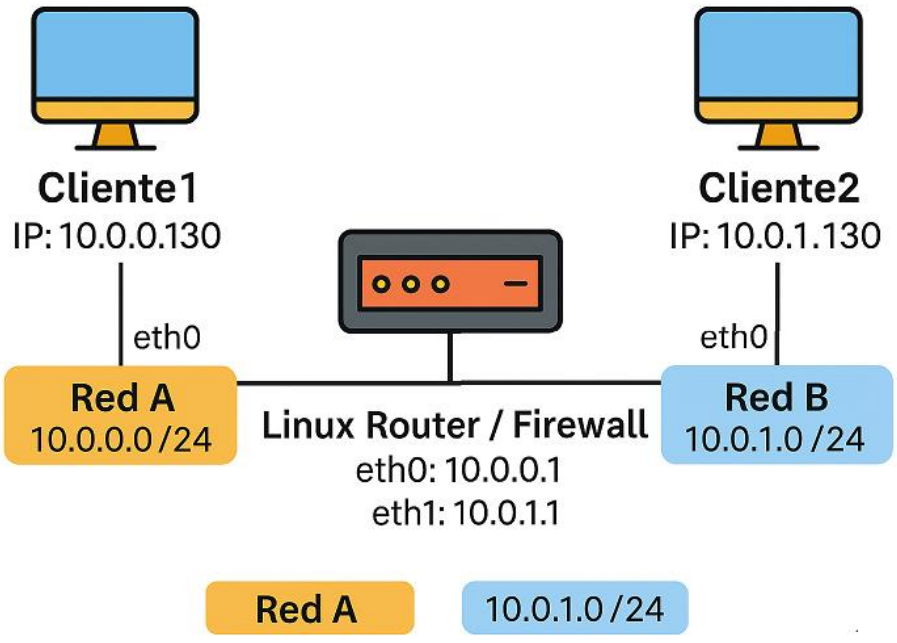
Objetivos específicos

- Comprender qué es la segmentación y por qué se usa en seguridad de red.
- Configurar redes virtuales en VMware que simulan VLANs.
- Implementar reglas de firewall con `iptables` en Linux.
- Probar la conectividad y las restricciones entre subredes.
- Reflexionar sobre las ventajas de segmentar frente a una red plana.

Requisitos

- VMware Workstation o VMware Player.
- 3 máquinas virtuales Linux (Ubuntu/Debian/Kali).
- Acceso a la terminal con permisos de administrador (`sudo`).
- Herramientas: `nmap`, `netcat`, `curl`, `iptables`, `apache2`.

1. Diseño de la topología



En VMware:

- Crea **dos redes virtuales separadas** (por ejemplo, "Red_A" y "Red_B") usando **VMnet** en modo "host-only" o "LAN segment".
- Conecta cada VM a su red correspondiente.

1.1. Crear las redes virtuales

Opción A: Usando *Host-only* con VMnet

1. Abre **VMware Workstation** → menú **Edit > Virtual Network Editor**.
2. Pulsa **"Change Settings"** (requiere permisos de administrador).
3. Crea o edita las redes:
    - **VMnet2** → selecciona **"Host-only"**, sin conexión al adaptador físico.
        - Renómbrala conceptualmente como **Red_A**.
    - **VMnet3** → selecciona también **"Host-only"**.
        - Renómbrala conceptualmente como **Red_B**.
4. (Opcional) Puedes configurar cada red con un rango IP distinto, por ejemplo:
    - VMnet2 (Red_A): `10.0.0.0/24`
    - VMnet3 (Red_B): `10.0.1.0/24`

Esto crea dos redes totalmente aisladas entre sí y del resto del sistema, salvo por el host.

Opción B: Usando *LAN Segments*

1. Abre la configuración de cualquier máquina virtual.
2. En la sección **Network Adapter**, selecciona:
    - **"Custom: Specific virtual network"**
    - Luego elige **"LAN Segment…"**.
3. Crea:
    - Un segmento llamado **Red_A**
    - Otro segmento llamado **Red_B**
4. VMware gestionará internamente estas LAN sin usar VMnet ni adaptadores del host.

Esta opción es ideal si solo quieres aislamiento completo entre VMs sin que el host intervenga.

1.2. Conectar las VMs a sus redes

Para cada VM Cliente:

- **VM1** (cliente en Red_A):
    1. Abre **Settings > Network Adapter**
    2. Selecciona:
        - *Custom (VMnet2)* o *LAN Segment: Red_A*
- **VM2** (cliente en Red_B):
    1. Abre **Settings > Network Adapter**

2. Selecciona:
    - *Custom (VMnet3)* o *LAN Segment: Red_B*

En la VM de **Kali Linux (router)**:

1. Apaga la VM si está encendida
2. Abre **Settings > Network Adapter**.
3. Agrega **dos adaptadores de red**:
    - **Adaptador 1:** conectado a **Red_A**
      (por ejemplo, `VMnet2` o `LAN Segment: Red_A`)
    - **Adaptador 2:** conectado a **Red_B**
      (por ejemplo, `VMnet3` o `LAN Segment: Red_B`)
4. Guarda los cambios e inicia Kali.

2. Asignar direcciones IP estáticas

Cliente1

Abrir terminal y ejecutar la configuración de la IP

```
sudo ip addr add 10.0.0.130/24 dev eth0
sudo ip link set eth0 up
sudo ip route add default via 10.0.0.1
sudo ip a
```

Cliente2

Abrir terminal y ejecutar la configuración de la IP

```
sudo ip addr add 10.0.1.130/24 dev eth0
sudo ip link set eth0 up
sudo ip route add default via 10.0.1.1
```

Router (Kali Linux)

Abrir terminal y ejecutar la configuración de la IP

```
sudo ip addr add 10.0.0.1/24 dev eth0
sudo ip addr add 10.0.1.1/24 dev eth1
sudo ip link set eth0 up
sudo ip link set eth1 up
```

Activar el reenvío de paquetes

```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
sudo sysctl -w net.ipv4.ip_forward=1
```

O de forma permanente

```
sudo nano /etc/sysctl.conf
```

Descomenta o añade `net.ipv4.ip_forward=1`

Rrecargar la configuración del sistema definida en el archivo */etc/sysctl.conf* y en otros archivos bajo */etc/sysctl.d/* sin necesidad de reiniciar el equipo.

```
sudo sysctl -p
```

3. Verificar la conectividad básica

Antes de aplicar reglas, asegúrarse de que el router enruta correctamente:

Desde **Cliente1**:

```
ping -c 2 10.0.0.1   # Debe responder (router)
ping -c 2 10.0.1.130 # Debe responder (Cliente2, si no hay firewall)
```

En este punto, el tráfico entre subredes todavía no está filtrado.

4. Aislar las subredes con iptables

En la máquina **Router (Kali Linux)**:

A veces Kali arranca con reglas por defecto que bloquean tráfico entre interfaces.

Verificar:

```
sudo iptables -L -v -n
```

Si se obsera alguna política por defecto `DROP` o reglas sospechosas, limpiar la configuración para hacer nuestras pruebas básicas:

```
sudo iptables -F
sudo iptables -P INPUT ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
```

1. Establecer política por defecto restrictiva:

```
sudo iptables -P FORWARD DROP
```

2. Permitir el tráfico dentro de cada subred:

```
sudo iptables -A FORWARD -s 10.0.0.0/24 -d 10.0.0.0/24 -j ACCEPT
```

```
sudo iptables -A FORWARD -s 10.0.1.0/24 -d 10.0.1.0/24 -j ACCEPT
```

3. Comprobar aislamiento:
   - Desde **Cliente1**:

     ```
     ping -c 2 10.0.1.130   # Debe fallar
     ```

   - Desde **Cliente2**:

     ```
     ping -c 2 10.0.0.130   # Debe fallar
     ```

5. Permitir tráfico HTTP entre las VLANs

1. Habilitar conexiones con estado necesario para respuestas:

   ```
   sudo iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
   ```

2. Permitir HTTP (puerto 80) de Cliente1 a Cliente2:

   ```
   sudo iptables -A FORWARD -s 10.0.0.130 -d 10.0.1.130 -p tcp --dport 80 -m state --
   state NEW -j ACCEPT
   ```

```
┌──(kali㉿kali)-[~]
└─$ sudo iptables -L -v -n

Chain INPUT (policy ACCEPT 102 packets, 8280 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 7 packets, 588 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       10.0.0.0/24          10.0.0.0/24
    0     0 ACCEPT     all  --  *      *       10.0.1.0/24          10.0.1.0/24
   15 11788 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            state RELATED,ESTABLISHED
    1    60 ACCEPT     tcp  --  *      *       10.0.0.130           10.0.1.130           tcp dpt:80 state NEW

Chain OUTPUT (policy ACCEPT 102 packets, 8280 bytes)
 pkts bytes target     prot opt in     out     source               destination

┌──(kali㉿kali)-[~]
└─$
```

3. En Cliente2, instalar, si no se tiene, y arrancar un servidor web:

   ```
   sudo apt update
   sudo apt install -y apache2
   sudo systemctl start apache2
   ```

4. Probar desde Cliente1:

   ```
   curl http://10.0.1.130
   ```

   Debe mostrar la página de inicio de Apache.

```
ubuntu@ubuntu-virtual-machine:~$ ping -c 3 10.0.1.130
PING 10.0.1.130 (10.0.1.130) 56(84) bytes of data.
^C
--- 10.0.1.130 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1017ms

ubuntu@ubuntu-virtual-machine:~$ curl http://10.0.1.130
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional
.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2022-03-22
    See: https://launchpad.net/bugs/1966004
  -->
  <head>
```

6. Verificar con herramientas

Desde **Cliente1**:

```
ping 10.0.1.130      # Debe fallar (ICMP bloqueado)
curl http://10.0.1.130   # Debe funcionar (HTTP permitido)
nmap 10.0.1.130      # Solo el puerto 80 debería aparecer como "open"
```

```
ubuntu@ubuntu-virtual-machine:~$ nmap 10.0.1.130
Starting Nmap 7.80 ( https://nmap.org ) at 2025-10-20 11:55 CEST
Nmap scan report for 10.0.1.130
Host is up (0.0045s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 4.73 seconds
ubuntu@ubuntu-virtual-machine:~$
```

7. Actividades complementarias

1. Permitir solo ICMP bloqueando todo lo demás:

```
sudo iptables -F
sudo iptables -P FORWARD DROP
sudo iptables -A FORWARD -p icmp -s 10.0.0.0/24 -d 10.0.1.0/24 -j ACCEPT
sudo iptables -A FORWARD -p icmp -s 10.0.1.0/24 -d 10.0.0.0/24 -j ACCEPT
```

2. Agregar una tercera red (10.0.2.0/24) y diseñar las reglas de interconexión (por ejemplo, permitir solo HTTP entre Red_A y Red_B, y solo ICMP entre Red_B y Red_C).
3. Implementar persistencia de reglas:

```
sudo apt install -y iptables-persistent
sudo netfilter-persistent save
```

8. Reflexión guiada

Responde brevemente:

1. ¿Qué beneficios ofrece la segmentación frente a una red plana?
2. ¿Qué tipo de equipos físicos (switches gestionables, routers) usarías en producción?
3. Si un atacante compromete una máquina en Red_A, ¿cómo le afecta la segmentación para moverse lateralmente?

9. Entregables sugeridos

Los alumnos deben incluir en su entrega:

- Capturas de:
  - Configuración IP de cada máquina (`ip addr`).
  - Reglas de iptables (`sudo iptables -L -v -n`).
  - Resultados de `ping`, `curl` y `nmap`.
- Breve texto reflexivo (5–10 líneas).
- Diagrama simple de la topología (puede ser hecho a mano o digital).