🔒 Teoría de Firewalls en Linux

¿Qué es un Firewall?

Un **firewall** es un sistema de seguridad que controla el tráfico de red (entrante y saliente) según reglas predefinidas. Su propósito principal es proteger los sistemas frente a accesos no autorizados y ataques, filtrando los paquetes de datos en función de criterios como direcciones IP, puertos, protocolos, etc.

Tipos de Firewalls

Tipo	Descripción
Firewall de red	Controla el tráfico entre distintas redes (por ejemplo, entre una red interna y la Internet).
Firewall de host	Controla el tráfico en un sistema específico (nivel de sistema operativo).
Firewalls de capa de aplicación	Inspeccionan datos a nivel de aplicación, como HTTP o DNS.
Stateful Firewall	Tienen conocimiento del estado de las conexiones (pueden permitir respuestas a solicitudes válidas).
Stateless Firewall	Analizan los paquetes sin conocimiento de contexto (más rápidos pero menos seguros).

🔧 Firewalls en Linux

Linux permite administrar firewalls a través de diversas herramientas. Las más comunes son:

- `firewalld` (con `firewall-cmd`) en **Red Hat**, **CentOS**, **Fedora**
- `ufw` (Uncomplicated Firewall) en **Ubuntu**, **Debian**
- `iptables` y `nftables`, que funcionan como backends (núcleo del filtrado)

🔥 `firewall-cmd` (para sistemas Red Hat-based)

`firewalld` es un daemon que gestiona las reglas de firewall de forma dinámica. Usa zonas y servicios para aplicar reglas.

🗒️ Tabla de Comandos de `firewall-cmd`

Comando	Descripción	Ejemplo
`--state`	Verifica si `firewalld` está activo	`firewall-cmd --state`
`--get-zones`	Lista las zonas disponibles	`firewall-cmd --get-zones`
`--get-active-zones`	Muestra zonas actualmente activas	`firewall-cmd --get-active-zones`

Comando	Descripción	Ejemplo
`--zone=public --add-port=80/tcp`	Añade puerto TCP 80 a la zona `public`	`firewall-cmd --zone=public --add-port=80/tcp --permanent`
`--reload`	Recarga la configuración	`firewall-cmd --reload`
`--list-all`	Lista todas las reglas de una zona	`firewall-cmd --list-all`

🔧 `ufw` (para sistemas Ubuntu-based)

`ufw` es una interfaz simplificada para `iptables`. Diseñada para facilitar el manejo del firewall para usuarios no expertos.

🗒️ Tabla de Comandos de `ufw`

Comando	Descripción	Ejemplo
`enable`	Activa el firewall	`sudo ufw enable`
`disable`	Desactiva el firewall	`sudo ufw disable`
`status`	Muestra el estado del firewall	`sudo ufw status`
`allow 22`	Permite tráfico en el puerto 22	`sudo ufw allow 22`
`deny 80`	Deniega tráfico en el puerto 80	`sudo ufw deny 80`
`delete allow 22`	Elimina una regla	`sudo ufw delete allow 22`
`reset`	Restaura configuración por defecto	`sudo ufw reset`

🛡️ ¿Qué es `iptables`?

`iptables` es una utilidad de línea de comandos para configurar las reglas del firewall que interactúan con el **Netfilter** en el kernel de Linux.

🗒️ Tabla de Opciones Comunes de `iptables`

Opción	Descripción
`-A`	Añade una regla
`-D`	Elimina una regla
`-I`	Inserta una regla en una posición específica
`-L`	Lista todas las reglas
`-F`	Limpia todas las reglas
`-p`	Especifica protocolo (`tcp`, `udp`, `icmp`)
`-s`	IP origen
`-d`	IP destino
`--dport`	Puerto destino
`-j`	Acción (`ACCEPT`, `DROP`, `REJECT`)

🌐 ¿Qué es `nftables`?

`nftables` es el reemplazo moderno de `iptables`, que unifica filtrado de paquetes, NAT y otras funciones. Se gestiona con el comando `nft`.

Ventajas frente a iptables:

- Sintaxis más limpia y coherente.
- Rendimiento mejorado.
- Soporte para conjuntos (sets) de IPs o puertos.
- Registros y estadísticas integradas.

🧪 Ejemplo con `iptables`: Bloqueo de ICMP (ping)

El protocolo ICMP es usado por herramientas como `ping`. A veces, se bloquea para evitar escaneos o ataques de red.

Paso 1: Ver las reglas actuales

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

Paso 2: Bloquear ICMP (ping)

```
sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

Esta regla **descarta** paquetes ICMP tipo "echo-request" (ping entrante).

Paso 3: Permitir tráfico ICMP

```
sudo iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
```

> 🔁 **Orden de las reglas importa**. Se procesan de arriba a abajo. La **primera coincidencia gana**. Si hay una regla `DROP` antes que una `ACCEPT`, se descarta el paquete.

🔁 Ejemplo práctico (corte de ICMP)

```
# Permitir todo el tráfico por defecto
sudo iptables -P INPUT ACCEPT

# Bloquear pings
sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

```
# Probar con ping desde otra máquina
ping <IP_del_host>

# No se recibe respuesta
```

🗒️ Tabla de ICMP (por tipo)

Tipo ICMP	Descripción
`0`	Echo reply
`3`	Destination unreachable
`5`	Redirect
`8`	Echo request (ping)
`11`	Time exceeded

📋 Tabla resumen de `iptables`

Elemento	Ejemplo	Descripción
Cadena (chain)	`INPUT`, `OUTPUT`, `FORWARD`	Dirección del tráfico
Tabla (table)	`filter`, `nat`, `mangle`	Tipo de procesamiento
Acción (target)	`ACCEPT`, `DROP`, `REJECT`	Qué hacer con el paquete
Protocolo	`-p tcp`, `-p udp`, `-p icmp`	Especifica protocolo
Puerto	`--dport 80`, `--sport 443`	Puerto destino u origen
Dirección IP	`-s 192.168.1.10`, `-d 8.8.8.8`	IP origen o destino