4

MANEJO DE PUERTOS

En este capítulo, exploraremos los conceptos fundamentales del manejo de puertos en sistemas Linux. Desde la identificación y administración de puertos abiertos, hasta el uso de herramientas de consola para analizar y gestionar los puertos, aprenderás cómo mantener tu sistema seguro y protegido contra amenazas externas.

4.1 CONCEPTOS BÁSICOS

La seguridad en sistemas Linux se destaca por su enfoque proactivo y reactivo para mitigar riesgos y proteger los sistemas contra amenazas tanto internas como externas. En comparación con otros sistemas operativos, como Windows, Linux se caracteriza por su arquitectura de código abierto, lo que permite una mayor transparencia y colaboración en el desarrollo de parches de seguridad y actualizaciones. Además, tiene una comunidad activa de usuarios y desarrolladores que contribuyen constantemente a mejorar la seguridad del sistema.

Una de las principales medidas de seguridad en sistemas Linux es la configuración adecuada del cortafuegos. Este actúa como una barrera entre la red interna y externa, filtrando el tráfico de red entrante y saliente según las reglas predefinidas. Mediante el uso de herramientas como **iptables** o **firewalld**, los administradores pueden especificar qué tipos de tráfico están permitidos y qué puertos están abiertos para la comunicación.

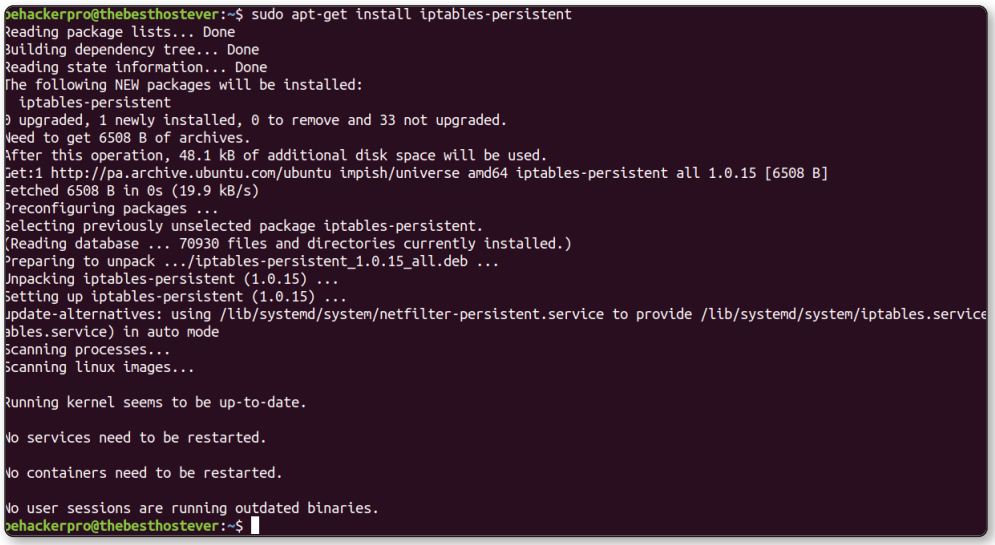

Figura 4.1. Iptables es una herramienta esencial en Linux, diseñada para gestionar y filtrar el tráfico de red a través de reglas definidas.

Otro aspecto crucial de la seguridad en sistemas Linux es la implementación de **políticas de acceso y autenticación** robustas. Esto incluye la gestión de usuarios y grupos, el uso de contraseñas seguras, y la configuración de permisos de archivos y directorios de manera adecuada. Mediante el empleo de herramientas como **sudo** y **SELinux** (Security-Enhanced Linux), los administradores pueden controlar el acceso de los usuarios y restringir los privilegios en función de roles y responsabilidades.

Además, la aplicación regular de actualizaciones de seguridad es fundamental para mitigar vulnerabilidades conocidas y proteger el sistema contra exploits y ataques. Las distribuciones Linux suelen ofrecer mecanismos de actualización automáticos o herramientas de gestión de paquetes que facilitan la instalación de parches y actualizaciones de seguridad de manera oportuna.

Para enfrentar la seguridad de Linux de manera efectiva, es fundamental adoptar un enfoque proactivo y seguir algunas recomendaciones generales que son comunes a cualquier sistema operativo:

- **Mantener el sistema actualizado**: el sistema Linux debe tener las últimas actualizaciones de seguridad y parches. Esto ayudará a mitigar vulnerabilidades conocidas y protegerá el sistema contra exploits y ataques.

- **Utilizar contraseñas seguras**: aplica contraseñas robustas y únicas para todas las cuentas de usuario y servicios en el sistema. Las contraseñas deben ser largas, complejas y difíciles de adivinar, y se recomienda el uso de herramientas de gestión para facilitar su administración.
- **Limitar los privilegios de usuario**: los privilegios de usuario deben estar limitados solo a lo que sea necesario para realizar tareas específicas. Utiliza cuentas de usuario con privilegios mínimos para las operaciones diarias y reserva el acceso de root o administrador solo para tareas administrativas que lo requieran.
- **Configurar un cortafuegos**: habilita un cortafuegos para filtrar el tráfico de red entrante y saliente. Define reglas específicas para permitir o denegar el acceso a determinados puertos y servicios, y asegúrate de revisar regularmente las configuraciones correspondientes para mantenerlas actualizadas.
- **Utilizar herramientas de monitoreo y registro**: implementa esta clase de herramientas para realizar un seguimiento de la actividad del sistema y detectar posibles intrusiones o comportamientos anómalos. Revisa regularmente los registros de eventos para identificar y responder con rapidez a posibles amenazas de seguridad.
- **Proteger los servicios de red**: configura los servicios de red de manera segura, deshabilitando aquellos que no sean necesarios y asegurando los que estén en uso con configuraciones adecuadas de autenticación y cifrado. Utiliza protocolos seguros, como SSH en vez de Telnet, y considera el uso de VPN para conexiones remotas.
- **Realizar copias de seguridad periódicas**: realiza copias de seguridad regulares de tus datos y configuraciones del sistema para poder restaurarlo en caso de que se produzca una brecha de seguridad o un fallo. Almacena dichas copias en ubicaciones seguras y fuera del sitio para protegerlas contra pérdidas catastróficas.

Siguiendo estas recomendaciones generales iniciales, ya estamos en condiciones de profundizar en medidas de seguridad más avanzadas, por ejemplo, la gestión de puertos.

4.1.1 ¿Qué son los puertos?

En el contexto de la informática y las redes, los **puertos** son puntos de conexión específicos que permiten la comunicación entre dispositivos a través de una red. En sistemas operativos como Linux, se los utiliza para permitir que los servicios y las aplicaciones se comuniquen entre sí y con otros dispositivos en una red.

Cada puerto está asociado a un número único, conocido como **número de puerto**, que identifica de manera exclusiva un servicio o aplicación determinada en un dispositivo. Los números de puerto van desde 0 hasta 65535 y se dividen en tres rangos.

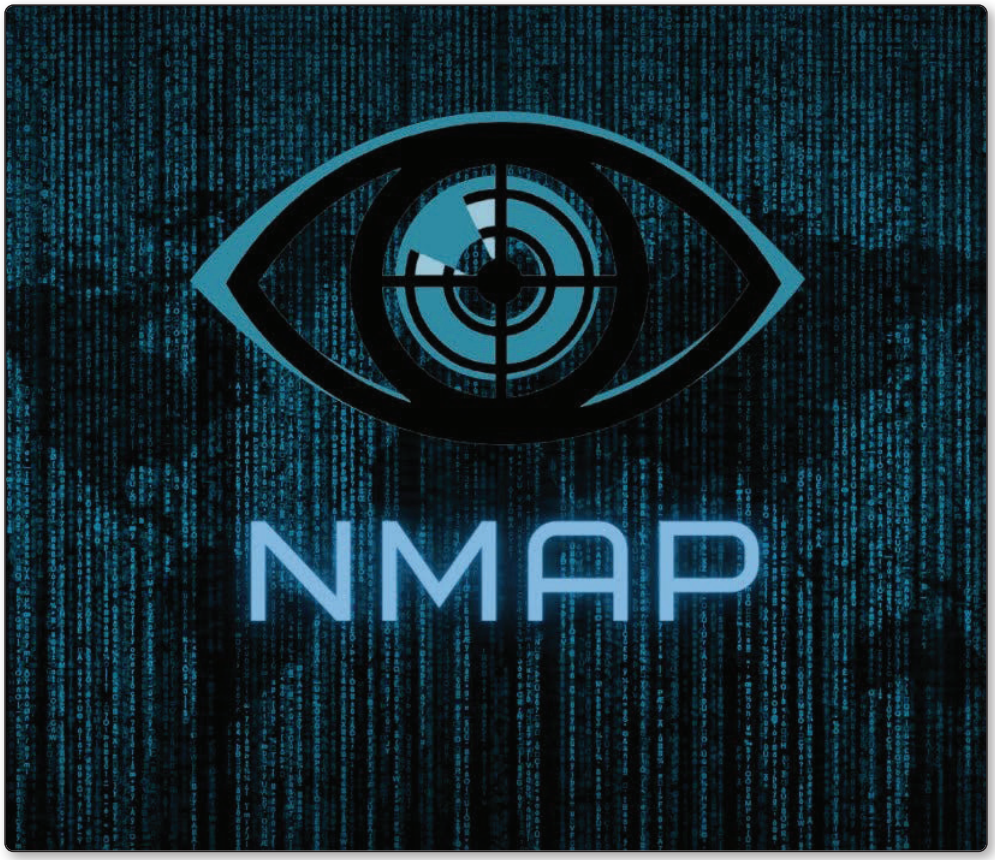

Figura 4.2. La forma más confiable para revisar los puertos que están escuchando en la red es mediante el uso de un escáner tal como nmap.

4.1.1.1 PUERTOS BIEN CONOCIDOS (0-1023)

También se los conoce como puertos reservados, y van del 0 al 1023. Están asignados por la Internet Assigned Numbers Authority (IANA) a servicios específicos, como el 80 para HTTP, el 443 para HTTPS y el 22 para SSH. Están estandarizados en sistemas operativos y aplicaciones de red. A continuación, verás algunos ejemplos de estos puertos:

Puerto	Servicio	Protocolo	Descripción
21	FTP	TCP	Protocolo de transferencia de archivos. Permite la transferencia de archivos entre sistemas conectados a una red TCP/IP.
22	SSH	TCP	Protocolo seguro de acceso remoto. Proporciona un canal seguro a través del cual se puede acceder y controlar un sistema de manera remota.
25	SMTP	TCP	Protocolo de transferencia de correo electrónico. Utilizado para el envío de correo entre servidores.
53	DNS	TCP/UDP	Sistema de nombres de dominio. Se emplea para traducir nombres de dominio legibles por humanos en direcciones IP numéricas.
80	HTTP	TCP	Protocolo de transferencia de hipertexto. Se usa para la transmisión de datos en la World Wide Web.
110	POP3	TCP	Protocolo de oficina de correos 3. Utilizado por clientes de correo electrónico para recuperar mensajes de un servidor de correo.
143	IMAP	TCP	Protocolo de acceso a mensajes de Internet. Permite a los usuarios acceder a sus correos electrónicos almacenados en un servidor de correo.
443	HTTPS	TCP	Protocolo seguro de transferencia de hipertexto. Utilizado para la comunicación segura a través de la World Wide Web.
3389	RDP	TCP/UDP	Protocolo de escritorio remoto. Permite a los usuarios conectarse y controlar de manera remota un equipo a través de una red.
20, 21	FTP	TCP	Protocolo de transferencia de archivos (datos y control). Utilizado para transferir archivos entre sistemas conectados a una red TCP/IP.

4.1.1.2 PUERTOS REGISTRADOS (1024-49151)

Están asignados a servicios y aplicaciones específicas por la Internet Assigned Numbers Authority (IANA). Algunos ejemplos incluyen el puerto 3306 para MySQL y el 5432 para PostgreSQL. Veamos algunos otros ejemplos:

Puerto	Protocolo	Descripción
1433	TCP	MSSQL–Servidor de base de datos Microsoft SQL Server
3306	TCP	MySQL–Servidor de base de datos MySQL
5432	TCP	PostgreSQL–Servidor de base de datos PostgreSQL
5900	TCP	VNC–Protocolo de control remoto de computadoras
8080	TCP	HTTP alternativo–Puerto alternativo para servidores web HTTP
8443	TCP	HTTPS alternativo–Puerto alternativo para servidores web HTTPS
9090	TCP	Apache Tomcat–Puerto predeterminado para el servidor web Apache Tomcat
1521	TCP	Oracle SQLNet–Puerto predeterminado para la base de datos Oracle SQLNet
389	TCP	LDAP–Protocolo de directorio ligero utilizado para la gestión centralizada de identidades
5222	TCP	XMPP–Protocolo de mensajería instantánea y presencia extendida

4.1.1.3 PUERTOS DINÁMICOS O PRIVADOS (49152-65535)

Estos puertos están disponibles para ser utilizados por aplicaciones y servicios de manera dinámica. Se emplean, principalmente, para establecer conexiones salientes y temporales. Veamos algunos ejemplos:

Puerto	Descripción
49152-49157	Puertos dinámicos utilizados por el sistema operativo para conexiones salientes
49158	Servicio Oracle Secure Backup
49159	Servicio Couplink GPRS
49160	Servicio de protocolo SNMP
49161	Servicio de control de acceso a red de Microsoft
49162-49166	Puertos de backdoor utilizados por algunos troyanos

Los puertos se clasifican en dos tipos principales: **TCP** (Transmission Control Protocol) y **UDP** (User Datagram Protocol). TCP es un protocolo de comunicación orientado a la conexión que garantiza la entrega de datos de manera ordenada y confiable, mientras que UDP es un protocolo sin conexión que ofrece una comunicación más rápida pero menos confiable.

En Linux, los puertos pueden ser utilizados por servicios y aplicaciones para escuchar conexiones entrantes o para establecer conexiones salientes. Por ejemplo,

un servidor web como Apache escucha las solicitudes de los clientes en el puerto 80, mientras que un servidor SSH escucha las conexiones entrantes en el puerto 22.

Es importante comprender los conceptos de puertos y sus números asociados para administrar la seguridad y el acceso en un sistema Linux. Esto incluye la configuración de reglas de **firewall** para controlar el tráfico de red entrante y saliente, así como la identificación y resolución de problemas relacionados con la conectividad de red y los servicios del sistema.

4.1.1.4 UN EJEMPLO DE FUNCIONAMIENTO

Un puerto en un sistema informático es una entidad virtual que permite que múltiples aplicaciones o servicios se comuniquen entre sí a través de una red. Para comprender su funcionamiento, consideremos un ejemplo específico: el puerto 80, que se utiliza comúnmente para el tráfico HTTP en la Web.

El puerto 80 es un puerto bien conocido que está reservado para el **protocolo HTTP** (Hypertext Transfer Protocol), el utilizado para transferir páginas web y otros recursos en la World Wide Web. Cuando un usuario intenta acceder a un sitio web en su navegador, como por ejemplo, **www.ejemplo.com**, el navegador establece una conexión con el servidor web del sitio utilizando el protocolo HTTP a través del puerto 80.

El proceso de comunicación comienza cuando el navegador envía una solicitud HTTP al servidor web. Esta solicitud está formada por varios componentes, incluyendo el método de solicitud (GET, POST, etc.), la URL del recurso solicitado (**www.ejemplo.com/pagina.html**) y otras cabeceras que pueden proporcionar información adicional al servidor.

La solicitud HTTP se manda desde el navegador al servidor a través de una **conexión TCP/IP** (Transmission Control Protocol/Internet Protocol). TCP es un protocolo orientado a la conexión que garantiza la entrega confiable de los datos, mientras que IP es el protocolo que se utiliza para enrutar los datos a través de la red.

El servidor web recibe la solicitud y la procesa. Busca el recurso pedido en su sistema de archivos o base de datos y genera la respuesta HTTP correspondiente. Esta incluye un código de estado (por ejemplo, 200 OK si la solicitud fue exitosa), cabeceras adicionales (como Content-Type para especificar el tipo de contenido) y el contenido real del recurso solicitado.

Una vez que la respuesta HTTP se ha generado, se envía de regreso al navegador a través de la misma conexión TCP/IP. El navegador recibe la respuesta y la interpreta, y muestra el contenido de la página web al usuario. Si la respuesta

incluye archivos adicionales, como imágenes, hojas de estilo o scripts, el navegador puede enviar solicitudes adicionales al servidor para obtener estos recursos.

La siguiente tabla resume los pasos involucrados en la comunicación a través del puerto 80 en el contexto del protocolo HTTP.

Paso	Descripción
Cliente (navegador web)	• Envía una solicitud HTTP al servidor web para solicitar un recurso. • La solicitud incluye la dirección del servidor (URL) y el método de solicitud HTTP (GET, POST, etc.).
Red	• La solicitud HTTP viaja a través de la red, utilizando el protocolo TCP/IP para la comunicación entre el cliente y el servidor.
Servidor Web	• Recibe la solicitud HTTP en el puerto 80. • Analiza la solicitud para determinar qué recurso se solicita y cómo debe responder. • Busca el recurso solicitado.
Procesamiento y generación de respuesta	• Genera una respuesta HTTP que incluye un código de estado, cabeceras y el contenido del recurso. • Puede ejecutar scripts o consultar una BD.
Red	• La respuesta HTTP viaja de regreso a través de la red al cliente, utilizando el protocolo TCP/IP.
Cliente (navegador web)	• Recibe la respuesta HTTP y la interpreta. • Si es necesario, envía solicitudes adicionales al servidor para obtener otros recursos.

Es importante destacar que el puerto 80 es solo un ejemplo de un puerto utilizado para un servicio específico (HTTP). Hay muchos otros puertos reservados para otros servicios comunes, como el 443 para **HTTPS** (HTTP seguro), el 25 para **SMTP** (Simple Mail Transfer Protocol) y el 22 para **SSH** (Secure Shell). Cada uno de ellos desempeña un papel importante en la comunicación de red y la transferencia de datos en Internet.

4.1.2 Tipos de puertos: TCP vs. UDP

En el contexto de la comunicación de red, los puertos TCP (Protocolo de Control de Transmisión) y UDP (Protocolo de Datagrama de Usuario) son dos protocolos fundamentales que operan en la capa de transporte del modelo OSI. Cada uno tiene características específicas que los hacen adecuados para diferentes aplicaciones y escenarios de red. A continuación, se detallan las diferencias entre estos dos tipos de puertos.

4.1.2.1 PUERTOS TCP

El **Protocolo de Control de Transmisión** (TCP) es un protocolo orientado a la conexión, confiable, que se utiliza para la transmisión de datos en redes. Algunas de sus características clave son:

- **Conexión orientada**: TCP establece una conexión bidireccional entre el cliente y el servidor antes de la transferencia de datos. Esto garantiza la entrega ordenada y confiable de la información.
- **Control de flujo**: TCP utiliza un mecanismo de control de flujo para regular la velocidad de transmisión de datos entre el emisor y el receptor. Esto ayuda a evitar la congestión de la red y garantiza una transferencia de datos eficiente.
- **Reconocimiento de recepción**: TCP emplea un mecanismo de reconocimiento de recepción para confirmar la entrega de datos. Si un paquete no se recibe correctamente, se solicitará su retransmisión.
- **Seguridad**: TCP ofrece mecanismos de seguridad integrados, como el cifrado SSL/TLS, que puede usarse para proteger la comunicación entre el cliente y el servidor.

Los puertos TCP son adecuados para aplicaciones que requieren una transmisión confiable de datos, como la transferencia de archivos, el correo electrónico y la navegación web.

4.1.2.2 PUERTOS UDP

El **Protocolo de Datagrama de Usuario** (UDP) es un protocolo sin conexión y no confiable que se utiliza para la transmisión de datos en redes. Algunas de sus características clave son:

- **Sin conexión**: UDP no establece una conexión antes de la transmisión de datos. Cada datagrama se envía de forma independiente, lo que puede resultar en una entrega desordenada o pérdida de datos.
- **No confiable**: UDP no proporciona mecanismos integrados para garantizar la entrega de datos o la detección de errores. Esto significa que no hay garantía de que los datos lleguen correctamente al destino.
- **Bajo overhead**: UDP tiene un overhead de protocolo más bajo en comparación con TCP, lo que lo hace adecuado para aplicaciones que

requieren una transmisión de datos rápida y eficiente, como la transmisión de video en tiempo real y la transmisión de voz sobre IP (VoIP).

- **Transmisión unidireccional**: UDP es adecuado para aplicaciones de transmisión unidireccional, donde la pérdida ocasional de datos no afecta significativamente la calidad de la transmisión.

Los puertos UDP son apropiados para aplicaciones donde la velocidad y la eficiencia son más importantes que la confiabilidad, como juegos en línea, streaming de medios y comunicaciones en tiempo real.

En resumen, los puertos son ideales para aplicaciones que requieren una transmisión confiable de datos, mientras que los UDP son más indicados para aplicaciones que priorizan la velocidad y la eficiencia sobre la confiabilidad. La elección entre TCP y UDP dependerá de los requisitos de cada aplicación y del contexto de la red en el que se los utilice.

4.2 IDENTIFICACIÓN DE PUERTOS ABIERTOS

En esta sección, nos adentraremos en la identificación de puertos abiertos en sistemas Linux, un aspecto fundamental para comprender la seguridad y la conectividad de una red. La capacidad para identificar y gestionar los puertos abiertos es esencial para proteger los sistemas contra amenazas externas y garantizar un entorno de red seguro y estable.

Antes de sumergirnos en los detalles técnicos, es importante repasar algunos conceptos clave. En el contexto de la comunicación de red, un puerto es un punto final único que se utiliza para la comunicación entre aplicaciones y dispositivos en una red.

Cada servicio o aplicación en un sistema operativo Linux utiliza un puerto específico para recibir y enviar datos.

El uso de la **consola de comandos** en sistemas Linux es una habilidad crítica para administradores de sistemas y profesionales de seguridad, ya que proporciona un nivel de control y flexibilidad sin igual en comparación con las interfaces gráficas de usuario. A través de la consola, los usuarios pueden ejecutar una amplia gama de comandos y herramientas especializadas para realizar tareas de administración, diagnóstico y configuración del sistema de manera eficiente.

La consola de comandos, también conocida como **terminal** o **shell**, permite a los usuarios interactuar directamente con el núcleo del sistema operativo a través de líneas de texto. Esto proporciona un mayor nivel de control sobre el sistema y permite realizar tareas avanzadas que pueden no ser accesibles a través de interfaces gráficas.

Una de las ventajas clave de la consola de comandos es su capacidad para **automatizar tareas** a través de **scripts y secuencias de comandos**. Esto permite a los administradores de sistemas realizar tareas repetitivas de manera eficiente y consistente, lo que ahorra tiempo y reduce la posibilidad de errores humanos.

Entre las herramientas y comandos especializados disponibles en la consola de comandos de Linux hay herramientas de redes como **netstat**, **nmap** y **ss**, que permiten realizar tareas de monitoreo y diagnóstico de la red. También existen comandos para administrar usuarios y permisos, gestionar servicios y procesos, realizar copias de seguridad y restauraciones, entre otros.

A continuación, exploraremos en detalle cómo utilizar estas herramientas para identificar puertos abiertos, comprender su estado y determinar qué servicios están escuchando en esos puertos. Con este conocimiento, podrás fortalecer la seguridad de tu sistema y garantizar una conectividad de red óptima y segura.

4.2.1 Herramientas de consola para el escaneo de puertos

En la administración y seguridad de sistemas Linux, el escaneo de puertos es una tarea fundamental para identificar los servicios en ejecución y detectar posibles vulnerabilidades. Para llevar a cabo esta tarea, existen diversas herramientas de consola altamente especializadas que proporcionan a los administradores de sistemas una amplia gama de opciones para analizar y evaluar la seguridad de sus sistemas. En esta sección, exploraremos algunas de las herramientas de consola más comúnmente utilizadas para el escaneo de puertos en sistemas Linux, destacando sus características principales, opciones de uso y ejemplos prácticos.

4.2.1.1 NMAP

Nmap es una de las herramientas más populares y potentes para el escaneo de puertos y el descubrimiento de dispositivos en una red. Ofrece una amplia gama de características y opciones avanzadas que permiten a los usuarios realizar escaneos detallados y exhaustivos de puertos en sistemas remotos. Para escanear todos los puertos en una máquina remota, podemos usar el siguiente comando:

```
nmap <host>
```

Por ejemplo, para escanear todos los puertos en la dirección IP 192.168.1.100, ejecutaríamos el siguiente comando:

```
nmap 192.168.1.100
```

Esto mostrará una lista de todos los puertos abiertos y los servicios en ejecución en la máquina remota.

4.2.1.2 NETCAT (NC)

Netcat es una herramienta versátil que se utiliza para leer y escribir datos a través de conexiones de red utilizando el protocolo TCP o UDP. Además de sus capacidades básicas de transferencia de datos, **Netcat** también se puede utilizar para escanear puertos en un sistema remoto. Para escanear un rango de puertos en una máquina remota, podemos usar el siguiente comando:

```
nc -zv <host> <puerto_inicio>-<puerto_fin>
```

Por ejemplo, para escanear los puertos del 1 al 100 en la dirección IP 192.168.1.100, ejecutaríamos el siguiente comando:

```
nc -zv 192.168.1.100 1-100
```

Esto generará una lista de puertos abiertos en la máquina remota.

4.2.1.3 HPING3

Hping3 es una herramienta de línea de comandos que se usa para generar paquetes TCP/IP y enviarlos a través de la red. Además de sus capacidades básicas de envío de paquetes, **Hping3** también se puede utilizar para escanear puertos en una máquina remota. Para realizar un escaneo de puertos SYN en una máquina remota, podemos usar el siguiente comando:

```
hping3 -S <host> -p <puerto>
```

Por ejemplo, para realizar un escaneo de puertos SYN en el puerto 80 en la dirección IP 192.168.1.100, ejecutaríamos el siguiente comando:

```
hping3 -S 192.168.1.100 -p 80
```

Esto generará un informe que indicará si el puerto está abierto, cerrado o filtrado.

4.2.1.4 MASSCAN

Masscan es una herramienta de escaneo de puertos de alta velocidad que utiliza técnicas de barrido paralelo para escanear grandes rangos de direcciones IP y puertos en poco tiempo. Es especialmente útil para realizar escaneos rápidos en redes de gran tamaño. Para escanear todos los puertos en una máquina remota, se puede usar el siguiente comando:

```
masscan <host> -p1-65535
```

Por ejemplo, para escanear todos los puertos en la dirección IP 192.168.1.100, ejecutaríamos este comando:

```
masscan 192.168.1.100 -p1-65535
```

Esto generará un informe detallado de los puertos abiertos en la máquina remota.

4.2.1.5 SS

Ss es una herramienta de línea de comandos que se utiliza para mostrar estadísticas de socket. Además de su funcionalidad básica, Ss también puede usarse para mostrar información sobre los puertos en escucha en un sistema Linux. Para listar los puertos en escucha en una máquina local, se puede utilizar el siguiente comando:

```
ss -ltn
```

Esto mostrará una lista de los puertos TCP en escucha en el sistema local, junto con el estado y la dirección IP asociada.

4.2.1.6 UNICORNSCAN

Unicornscan es una herramienta de escaneo de puertos de red que utiliza técnicas avanzadas para realizar escaneos rápidos y eficientes. Ofrece características como escaneo de puertos TCP y UDP, detección de servicios y detección de sistemas operativos. Para escanear un rango de puertos en una máquina remota, se puede aplicar el siguiente comando:

```
unicornscan <host>:<puerto_inicio>-<puerto_fin>
```

Por ejemplo, para escanear los puertos del 1 al 100 en la dirección IP 192.168.1.100, hay que ejecutar el siguiente comando:

```
unicornscan 192.168.1.100:1-100
```

Esto proporcionará información detallada sobre los puertos abiertos en la máquina remota.

4.2.2 Ejemplo práctico de escaneo de puertos con nmap y análisis de resultados

Veamos un ejemplo práctico básico de identificación de puertos abiertos y análisis de resultados. Este ejemplo proporciona una visión más detallada del proceso de identificación de puertos abiertos y el análisis de los resultados utilizando la herramienta **nmap** en una máquina remota. Es importante tener en cuenta que este proceso puede variar según las necesidades y el entorno específico, y es fundamental contar con un entendimiento sólido de los conceptos de seguridad de red y las herramientas disponibles para llevar a cabo esta tarea de manera efectiva.

4.2.2.1 IDENTIFICACIÓN DE PUERTOS ABIERTOS

Utilizaremos la herramienta **nmap** para realizar un escaneo de puertos en una máquina remota. Esta herramienta es una de las más populares y potentes para el escaneo de puertos y la detección de servicios en una red. Supongamos que queremos escanear la dirección IP 192.168.1.100 para identificar los puertos abiertos. Ejecutamos el siguiente comando en la terminal:

```
nmap 192.168.1.100
```

Este comando realizará un escaneo de los 1000 puertos más comunes en la máquina remota y mostrará información detallada sobre los puertos abiertos, los servicios que están escuchando en esos puertos y otra información relevante.

4.2.2.2 ANÁLISIS DE RESULTADOS

Después de ejecutar el comando **nmap**, obtendremos un resultado que se verá de esta manera:

```
Starting Nmap 7.80 ( https://nmap.org ) at 2023-10-29 12:00 UTC
Nmap scan report for 192.168.1.100
Host is up (0.0020s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open ssh
80/tcp    open http
443/tcp   open https
3306/tcp  open mysql
```

Este resultado indica que la máquina remota tiene los siguientes puertos abiertos:

- El puerto 22 (SSH), lo que significa que el servidor SSH está en funcionamiento y aceptando conexiones.
- El puerto 80 (HTTP), lo que indica que hay un servidor web HTTP activo y accesible.
- El puerto 443 (HTTPS), lo que sugiere que hay un servidor web HTTPS en funcionamiento, probablemente para conexiones seguras.
- El puerto 3306 (MySQL), lo que indica que hay un servidor MySQL en funcionamiento y aceptando conexiones de red.

4.2.2.3 ANÁLISIS ADICIONAL

Basándonos en estos resultados, podemos realizar un análisis adicional para determinar el nivel de riesgo y las posibles vulnerabilidades de seguridad en la máquina remota. Podríamos querer investigar más a fondo cada servicio identificado para asegurarnos de que estén correctamente configurados y actualizados. También podríamos considerar tomar medidas adicionales, como el bloqueo de puertos no utilizados o la implementación de medidas de seguridad adicionales en los servicios expuestos.

4.2.3 Ejemplo práctico de escaneo de puertos con netcat y análisis de resultados

Veamos un ejemplo utilizando la herramienta **netcat** para realizar un escaneo de puertos y luego analizar los resultados:

4.2.3.1 IDENTIFICACIÓN DE PUERTOS ABIERTOS

Utilizaremos la herramienta **netcat** para realizar un escaneo de puertos en una máquina remota. Se trata de una herramienta versátil que puede utilizarse para leer y escribir datos a través de conexiones de red. Supongamos que queremos escanear la dirección IP 192.168.1.100 para identificar los puertos abiertos. Ejecutamos el siguiente comando en la terminal:

```
nc -zv 192.168.1.100 1-1000
```

Este comando realizará un escaneo de los primeros 1000 puertos en la máquina remota y mostrará información detallada sobre los que están abiertos.

4.2.3.2 ANÁLISIS DE RESULTADOS

Después de ejecutar el comando **netcat**, obtendremos un resultado que se verá de este modo:

```
Connection to 192.168.1.100 port 22 [tcp/ssh] succeeded!
Connection to 192.168.1.100 port 80 [tcp/http] succeeded!
Connection to 192.168.1.100 port 443 [tcp/https] succeeded!
Connection to 192.168.1.100 port 3306 [tcp/mysql] succeeded!
```

Este resultado indica que la máquina remota tiene los siguientes puertos abiertos:

- El puerto 22 (SSH), lo que significa que el servidor SSH está en funcionamiento y aceptando conexiones.
- El puerto 80 (HTTP), lo que indica que hay un servidor web HTTP activo y accesible.
- El puerto 443 (HTTPS), lo que sugiere que hay un servidor web HTTPS en funcionamiento, probablemente para conexiones seguras.
- El puerto 3306 (MySQL), lo que indica que hay un servidor MySQL en funcionamiento y aceptando conexiones de red.

4.2.3.3 ANÁLISIS ADICIONAL

Basándonos en estos resultados, podemos realizar un análisis adicional para determinar el nivel de riesgo y las posibles vulnerabilidades de seguridad en la máquina remota. Podríamos querer investigar más a fondo cada servicio identificado para asegurarnos de que estén correctamente configurados y actualizados. También podríamos considerar tomar medidas adicionales, como el bloqueo de puertos no utilizados o la implementación de medidas de seguridad adicionales en los servicios expuestos.

4.2.4 Ejemplo práctico de escaneo de puertos con hping3 y análisis de resultados

Veamos un ejemplo completo utilizando el comando **hping3** para realizar un escaneo de puertos y analizar los resultados.

4.2.4.1 IDENTIFICACIÓN DE PUERTOS ABIERTOS

Utilizaremos la herramienta **hping3** para realizar un escaneo de puertos en una máquina remota. Esta herramienta puede generar paquetes TCP/IP personalizados y enviarlos a través de la red. Supongamos que queremos escanear la dirección IP 192.168.1.100 para identificar los puertos abiertos. Ejecutamos el siguiente comando en la terminal:

```
hping3 -S -p 1-1000 -c 1 192.168.1.100
```

Este comando enviará un solo paquete SYN a cada puerto en el rango del 1 al 1000 y mostrará la respuesta recibida.

4.2.4.2 ANÁLISIS DE RESULTADOS

Después de ejecutar el comando **hping3**, obtendremos un resultado que se verá de la siguiente manera:

```
len=46 ip=192.168.1.100 ttl=64 DF id=0 sport=22 flags=SA seq=0 win=29200 rtt=4.4
ms
len=46 ip=192.168.1.100 ttl=64 DF id=0 sport=80 flags=SA seq=0 win=29200 rtt=4.7
ms
len=46 ip=192.168.1.100 ttl=64 DF id=0 sport=443 flags=SA seq=0 win=29200 rtt=4.2
ms
len=46 ip=192.168.1.100 ttl=64 DF id=0 sport=3306 flags=SA seq=0 win=29200
rtt=4.9 ms
```

Este resultado indica que la máquina remota ha respondido a los paquetes SYN enviados a los siguientes puertos:

- El puerto 22 (SSH) está abierto y listo para establecer una conexión.
- El puerto 80 (HTTP) está abierto y aceptando conexiones.
- El puerto 443 (HTTPS) está abierto y disponible para conexiones seguras.
- El puerto 3306 (MySQL) está abierto y listo para conexiones a la base de datos MySQL.

4.2.4.3 ANÁLISIS ADICIONAL

Como en el ejemplo anterior, podemos realizar un análisis adicional para evaluar el riesgo y las posibles vulnerabilidades de seguridad en la máquina remota. Esto puede incluir investigaciones adicionales sobre los servicios identificados, así como la implementación de medidas de seguridad extra según sea necesario.

4.3 ADMINISTRACIÓN DE PUERTOS

La administración de puertos en sistemas Linux es un aspecto fundamental de la seguridad informática, ya que los puertos son puntos de entrada y salida para la comunicación de red. La correcta gestión de los puertos es esencial para garantizar la integridad, confidencialidad y disponibilidad de los servicios y datos alojados en el sistema.

4.3.1 Enfoque basado en archivos

Linux utiliza un enfoque basado en archivos para la administración de puertos, donde los puertos están representados como archivos en el sistema de archivos virtual **/proc** y **/sys**. Estos archivos proporcionan información sobre el estado y la configuración de los puertos, así como sobre la capacidad de configurarlos y administrarlos.

En sistemas Linux, los directorios **/proc** y **/sys** son fundamentales para acceder y manipular información sobre el hardware, los procesos en ejecución y otros aspectos del sistema operativo en tiempo de ejecución. Estos directorios proporcionan una interfaz de acceso a datos virtual, donde los usuarios y programas pueden acceder a información y configuraciones del kernel y del sistema en tiempo real.

El directorio **/proc** contiene una serie de archivos y subdirectorios que representan procesos, controladores de dispositivos, información de hardware y otros recursos del sistema. Cada proceso en ejecución tiene un directorio correspondiente en **/proc**, con número de identificación de proceso (PID). Dentro de estos directorios, se pueden encontrar archivos como **cmdline**, que contiene la línea de comandos utilizada para iniciar el proceso, y **status**, que proporciona información detallada sobre el estado del proceso.

Además de la información de los procesos, **/proc** también contiene archivos especiales que permiten acceder a información sobre la configuración del kernel, como **cpuinfo**, que muestra información sobre la CPU del sistema, y **meminfo**, que proporciona información sobre el uso de la memoria del sistema.

Por otro lado, el directorio **/sys** es una interfaz virtual que permite a los usuarios y programas acceder a información sobre dispositivos y controladores de dispositivos en el sistema. Similar a **/proc**, **/sys** contiene una serie de archivos y directorios que representan dispositivos y características del hardware del sistema. Por ejemplo, **/sys/class/net** contiene subdirectorios para cada interfaz de red en el

sistema, donde se pueden encontrar archivos como **speed**, que muestra la velocidad de la interfaz de red, y **carrier**, que indica si la interfaz está conectada o no.

En los directorios **/proc** y **/sys** en sistemas Linux, no hay archivos específicos que proporcionen información directa sobre los puertos abiertos en el sistema. Sin embargo, es posible obtener información relacionada con los puertos a través de otros archivos y directorios en estos directorios virtuales.

Por ejemplo, el directorio **/proc/net** tiene información sobre el estado de las conexiones de red del sistema. Los archivos en este directorio, como **tcp**, **udp** y **tcp6**, contienen información detallada sobre las conexiones TCP y UDP activas, incluidos los puertos locales y remotos asociados con cada conexión.

En el caso de **/proc/net/tcp** y **/proc/net/udp**, cada línea en estos archivos representa una conexión TCP o UDP activa, respectivamente. La información en estas líneas incluye el estado de la conexión, las direcciones IP y los números de puerto locales y remotos, entre otros detalles. Al analizar estos archivos, es posible identificar los puertos que están siendo utilizados por conexiones de red en el sistema.

Por otro lado, en el directorio **/sys/class/net** existe información sobre las interfaces de red del sistema, pero no sobre los puertos específicos que están en uso. Sin embargo, al monitorear el estado de las interfaces de red, es posible inferir qué puertos están siendo utilizados por las conexiones de red en el sistema.

4.3.2 Comandos especiales

Uno de los aspectos clave en la administración de puertos en Linux es el uso del comando **netstat** para listar los puertos abiertos y las conexiones de red activas. El comando **netstat** proporciona una visión detallada de los puertos en uso, junto con información sobre el protocolo, la dirección IP local y remota, el estado de la conexión y el proceso asociado.

Además del comando **netstat**, Linux ofrece otras herramientas, como **ss** (socket statistics) y **lsof** (list open files), que pueden usarse para obtener información adicional sobre los puertos y los procesos asociados a ellos.

La administración de puertos en Linux también implica la configuración del cortafuegos para controlar el tráfico de red entrante y saliente. El cortafuegos puede configurarse utilizando herramientas como **iptables** o **firewalld**, que permiten especificar reglas de filtrado basadas en direcciones IP, puertos y protocolos.

Es importante tener en cuenta que la apertura indiscriminada de puertos puede exponer el sistema a vulnerabilidades de seguridad y ataques de red. Por

lo tanto, es recomendable mantener abiertos solo los puertos necesarios y aplicar medidas adicionales de seguridad, como el cifrado de datos y la autenticación de usuarios, para proteger el sistema contra amenazas externas.

4.3.3 Uso de comandos de consola

4.3.4 para administrar puertos abiertos

Para administrar puertos abiertos en sistemas Linux, se utilizar una variedad de comandos de consola que permiten visualizar, abrir y cerrar puertos, así como gestionar las reglas del cortafuegos. A continuación, se presentan algunos de estos comandos junto con ejemplos concretos:

netstat: permite mostrar información sobre las conexiones de red activas, incluidos los puertos que está utilizando el sistema:

```
netstat -tuln
```

Este comando muestra una lista de puertos TCP (**-t**) y UDP (**-u**) que están escuchando (**-l**) y muestra los números de puerto en formato numérico (**-n**).

ss: similar al anterior, es otra herramienta para mostrar información sobre las conexiones de red:

```
ss -tuln
```

Al igual que **netstat**, este comando muestra una lista de puertos TCP y UDP que están escuchando.

lsof: muestra los archivos que están siendo utilizados por los procesos en el sistema, incluyendo los sockets de red, lo que permite identificar los procesos que están usando determinados puertos:

```
lsof -i :80
```

Este comando muestra los procesos que están utilizando el puerto 80.

iptables: permite configurar las reglas del cortafuegos en sistemas Linux:

```
iptables -A INPUT -p tcp —dport 22 -j DROP
```

Este comando añade una regla a la cadena INPUT para bloquear (DROP) el tráfico TCP entrante al puerto 22.

firewall-cmd: si estás utilizando **firewalld**, puedes utilizar este comando para administrar las reglas del cortafuegos:

```
firewall-cmd –zone=public –add-port=80/tcp –permanent
```

Este comando añade permanentemente el puerto TCP 80 a la zona public del cortafuegos.

A continuación se presentan algunos ejemplos de comandos de consola que permiten abrir y cerrar puertos en sistemas Linux:

- Abrir un puerto TCP utilizando **iptables**:

  ```
  sudo iptables -A INPUT -p tcp –dport 8080 -j ACCEPT
  ```

  Este comando añade una regla a la cadena INPUT del cortafuegos para aceptar (ACCEPT) el tráfico TCP entrante al puerto 8080.

- Cerrar un puerto TCP utilizando **iptables**:

  ```
  sudo iptables -A INPUT -p tcp –dport 8080 -j DROP
  ```

  Este comando añade una regla a la cadena INPUT del cortafuegos para rechazar (DROP) el tráfico TCP entrante al puerto 8080.

- Abrir un puerto UDP utilizando **iptables**:

  ```
  sudo iptables -A INPUT -p udp –dport 123 -j ACCEPT
  ```

  Este comando añade una regla a la cadena INPUT del cortafuegos para aceptar el tráfico UDP entrante al puerto 123 (por ejemplo, para el servicio NTP).

- Cerrar un puerto UDP utilizando **iptables**:

  ```
  sudo iptables -A INPUT -p udp –dport 123 -j DROP
  ```

  Este comando añade una regla a la cadena INPUT del cortafuegos para rechazar el tráfico UDP entrante al puerto 123.

4.4 ACTIVIDADES

A continuación se presentan las preguntas y los ejercicios que deberías saber responder y resolver para considerar aprendido el capítulo.

4.4.1 Test de autoevaluación

1. *¿Qué son los puertos en el contexto de la seguridad de Linux?*
2. *¿Cuál es la diferencia entre los puertos bien conocidos, los puertos registrados y los puertos dinámicos?*
3. *¿Por qué es importante escanear puertos en un sistema Linux?*
4. *¿Cuál es la diferencia entre TCP y UDP en términos de comunicación de red?*
5. *¿Cómo se puede abrir un puerto TCP utilizando el comando **iptables**?*
6. *¿Qué comando se utiliza para cerrar un puerto UDP utilizando **iptables**?*
7. *¿Qué tipo de archivo en Linux se utiliza para configurar reglas de firewall de manera permanente?*
8. *¿Cuál es la diferencia entre los directorios **/proc** y **/sys** en Linux?*
9. *¿Qué comando se puede utilizar para listar todos los puertos abiertos en un sistema Linux?*
10. *¿Por qué es importante administrar correctamente los puertos abiertos en un sistema Linux?*

4.4.2 Ejercicios prácticos

1. *Utiliza el comando **netstat** para identificar todos los puertos TCP abiertos en un sistema Linux.*
2. *Utiliza el comando **iptables** para abrir el puerto 22 (SSH) en un sistema Linux.*
3. *Escanea una red local en busca de puertos abiertos utilizando la herramienta **nmap**.*
4. *Utiliza el comando **ss** para mostrar estadísticas de conexión de red en un sistema Linux.*
5. *Configura una regla de firewall persistente utilizando el archivo de configuración adecuado en Linux para abrir el puerto 80 (HTTP).*

5

PENTESTING EN REDES LOCALES

En el mundo actual, donde la seguridad de la información es una preocupación constante, el pentesting en redes locales se ha vuelto una práctica fundamental. Este capítulo te guiará a través del proceso de evaluación de la seguridad de una red interna y te permitirá descubrir posibles brechas que podrían ser aprovechadas por ciberatacantes. Desde la configuración del entorno de pruebas hasta la ejecución de ataques de explotación, explorarás las herramientas y técnicas necesarias para fortalecer la seguridad de tu red.

5.1 PREPARACIÓN DEL ENTORNO

La preparación del entorno es una fase crítica en el proceso de **pentesting** en redes locales. Antes de comenzar a evaluar la seguridad de una red interna, es fundamental configurar un entorno de pruebas adecuado y seleccionar las herramientas necesarias para realizar las evaluaciones de manera efectiva.

Una de las primeras consideraciones al preparar el entorno es la configuración de una máquina virtual dedicada para pruebas. Utilizar una máquina virtual te permite simular diferentes escenarios de red sin comprometer la seguridad de tu entorno de producción. Puedes instalar sistemas operativos específicos, configurar servicios y realizar pruebas de penetración sin temor a dañar sistemas reales. Herramientas como VirtualBox o VMware son ampliamente utilizadas para este fin y ofrecen una variada gama de funcionalidades para la creación y gestión de máquinas virtuales.