Anexo. Herramientas de red Linux y Windows

1. Introducción

1.1. Objetivos

- **Explicar la importancia de las herramientas de red en la administración y seguridad de sistemas operativos**:
  - En el mundo de la seguridad informática, las redes juegan un papel crucial. Todo sistema operativo conectado a una red debe ser gestionado y asegurado adecuadamente para prevenir y responder a incidentes de seguridad.
  - Las herramientas de red proporcionadas por los sistemas operativos permiten a los administradores diagnosticar problemas, analizar tráfico, configurar interfaces, y asegurar la infraestructura de red.
  - Sin una comprensión profunda de estas herramientas, los profesionales de TI y seguridad no podrían mantener la integridad, confidencialidad y disponibilidad de los sistemas y datos.
- **Mostrar cómo utilizar herramientas de red en Linux y Windows (CMD y PowerShell) para diagnosticar y solucionar problemas de red**:
  - A lo largo de esta sesión, aprenderemos cómo utilizar varias herramientas nativas de red en Linux y Windows. Estas herramientas permiten desde la simple verificación

de conectividad hasta el análisis avanzado de tráfico y la configuración de políticas de seguridad.

- Ejemplo: **Linux**: ping para verificar conectividad con un servidor; **Windows**: Test-Connection en PowerShell para realizar una prueba similar, pero con más flexibilidad y opciones adicionales.

1.2. Contexto

- **La importancia en la seguridad informática**:
  - En seguridad informática, los problemas de red pueden ser indicativos de amenazas como ataques de denegación de servicio (DDoS), infiltración de redes o fallas de configuración que podrían ser explotadas por atacantes.
  - Las herramientas de red no solo permiten a los administradores identificar y solucionar problemas de conectividad, sino también detectar patrones de tráfico sospechoso, auditar configuraciones de seguridad, y implementar controles que protejan la infraestructura.
  - Ejemplo: En un entorno donde se sospecha de una intrusión, **tcpdump** en Linux puede capturar paquetes de red para un análisis detallado, mientras que en Windows, **Get-NetTCPConnection** en PowerShell puede proporcionar una lista de conexiones activas para inspeccionar posibles conexiones no autorizadas.
- **Particularidades y ventajas de Linux y Windows**:
  - **Linux**: Es conocido por su robustez y flexibilidad en herramientas de red. La gran cantidad de comandos disponibles y su capacidad para ser combinados en scripts permiten un control detallado del entorno de red.
  - **Windows**: Mientras que las herramientas gráficas de Windows son fáciles de usar, CMD y PowerShell ofrecen potentes comandos de red que permiten a los

administradores realizar tareas complejas. PowerShell, en particular, se destaca por su capacidad de automatización y su integración profunda con el sistema operativo.

- Ejemplo: En Linux, el uso de iptables permite un control granular sobre el tráfico de red que atraviesa el sistema. En Windows, el uso de netsh permite configurar reglas avanzadas de firewall y otras configuraciones de red.

1.3. Visión General

- **Enfoque en herramientas nativas y su relevancia en escenarios de seguridad**:
  - En este curso, nos enfocaremos en las herramientas que vienen integradas en Linux y Windows, ya que son las primeras líneas de defensa y diagnóstico que cualquier administrador debe conocer.
  - Estas herramientas son esenciales en escenarios donde la seguridad es crítica, como en la detección de intentos de acceso no autorizados, la monitorización de conexiones activas, y la configuración de políticas de firewall.
  - Ejemplo: Durante un análisis forense de un incidente de seguridad, herramientas como netstat en Windows y Linux pueden revelar conexiones establecidas por posibles atacantes, lo que es crucial para contener y mitigar un incidente.
- **Diferencias en la gestión de red entre Linux y Windows**:
  - **Linux**: Ofrece un enfoque más abierto y personalizable, donde la mayoría de las herramientas de red están basadas en la línea de comandos. Esto permite un alto grado de flexibilidad, pero requiere un conocimiento más profundo de los comandos y su sintaxis.
  - **Windows**: Proporciona herramientas tanto en interfaces gráficas como en línea de comandos, con PowerShell ofreciendo un entorno de scripting poderoso que permite la automatización de tareas de red. La gestión en Windows tiende a ser más accesible para los usuarios novatos, pero no sacrifica potencia para los administradores avanzados.

- Ejemplo: Configurar una dirección IP en Linux puede realizarse con ip addr add en la terminal, mientras que en Windows se podría hacer gráficamente a través del Panel de Control o mediante New-NetIPAddress en PowerShell, dependiendo del nivel de control y automatización deseado.

2. Herramientas de Red en Linux

2.1. Introducción a las Herramientas de Red en Linux

- **Descripción**:
  - Linux es un sistema operativo ampliamente utilizado en servidores, dispositivos de red y sistemas embebidos debido a su flexibilidad, robustez y capacidad para ser configurado a fondo desde la línea de comandos.
  - En el ámbito de la red, Linux ofrece una variedad de herramientas nativas de línea de comandos que permiten a los administradores de sistemas y profesionales de seguridad gestionar, diagnosticar y monitorear redes con un control detallado.
  - Estas herramientas son fundamentales para realizar tareas como verificar la conectividad, analizar el tráfico de red, configurar interfaces, gestionar reglas de firewall, y resolver problemas de DNS, entre otros.

2.2. Herramientas Principales

1. **ping**:
   - **Uso**: Verificar la conectividad entre el host y otro dispositivo en la red.
   - **Descripción**: El comando ping envía paquetes ICMP "echo request" a un host remoto y espera respuestas "echo reply", permitiendo comprobar si el host está accesible y medir la latencia de la conexión.
   - **Ejemplo**:
     ```
     ping google.com
     ```
     Este comando enviará paquetes ICMP a google.com y mostrará el tiempo de respuesta y la tasa de pérdida de paquetes.

2. **ifconfig/ip**:
   - **Uso**: Mostrar y configurar interfaces de red.
   - **Descripción**: ifconfig es la herramienta tradicional en Linux para mostrar y configurar interfaces de red. Sin embargo, ip, parte del paquete iproute2, es la herramienta moderna y más poderosa para estas tareas.
   - **Ejemplo**:
     ```
     ip addr show
     ```
     Este comando muestra la configuración de todas las interfaces de red, incluyendo direcciones IP, estado de las interfaces y más.

3. **netstat/ss**:
   - **Uso**: Mostrar conexiones de red, puertos en uso, y estadísticas de red.
   - **Descripción**: netstat es una herramienta clásica para ver las conexiones de red activas y las estadísticas del sistema de red. ss es su reemplazo moderno, ofreciendo más velocidad y funcionalidad mejorada.
   - **Ejemplo**:
     ```
     netstat -an
     ```
     Muestra todas las conexiones de red activas y los puertos en uso.
     ```
     ss -tuln
     ```
     Muestra puertos TCP y UDP en escucha junto con las estadísticas relevantes.
4. **traceroute**:
   - **Uso**: Rastrear la ruta que sigue un paquete a través de una red.
   - **Descripción**: traceroute envía paquetes con incrementos en el TTL (Time to Live) para cada salto, permitiendo identificar cada nodo intermedio entre el origen y el destino.
   - **Ejemplo**:
     ```
     traceroute google.com
     ```
     Muestra todos los nodos por los que pasan los paquetes hasta llegar a google.com, útil para diagnosticar problemas de red.

5. **nslookup/dig**:
   - **Uso**: Consultar DNS para resolver nombres de dominio o direcciones IP.
   - **Descripción**: nslookup y dig son herramientas para consultar servidores DNS. dig es más poderosa y flexible, proporcionando información detallada sobre el proceso de resolución.
   - **Ejemplo**:
     ```
     dig google.com
     ```
     Muestra detalles completos sobre la resolución de google.com, incluyendo los servidores consultados y la respuesta del DNS.
6. **tcpdump**:
   - **Uso**: Capturar y analizar paquetes de red en tiempo real.
   - **Descripción**: tcpdump es una herramienta de captura de paquetes que permite monitorear y analizar el tráfico de red en tiempo real. Es esencial para el análisis forense de redes y resolución de problemas complejos.
   - **Ejemplo**:
     ```
     tcpdump -i eth0
     ```
     Captura todos los paquetes que pasan por la interfaz eth0. Los datos capturados pueden ser analizados directamente o guardados para su posterior análisis con herramientas como Wireshark.

7. **iptables**:
   - **Uso**: Configurar reglas de firewall.
   - **Descripción**: iptables es una herramienta de administración de firewall que permite definir reglas para filtrar el tráfico de red que pasa a través del sistema Linux.
   - **Ejemplo**:
     ```
     iptables -L
     ```
     Lista las reglas de firewall actuales configuradas en el sistema.
8. **nmap**:
   - **Uso**: Escanear puertos y realizar auditorías de red.
   - **Descripción**: nmap es una poderosa herramienta de escaneo de redes utilizada para descubrir hosts, servicios y detectar vulnerabilidades en una red. Es ampliamente utilizada en pruebas de penetración y auditorías de seguridad.
   - **Ejemplo**:
     ```
     nmap -sP 192.168.1.0/24
     ```
     Escanea la red local 192.168.1.0/24 para identificar dispositivos activos.

2.3. Ejemplos Prácticos

1. **Diagnóstico de Conectividad**:
   - **Uso de ping y traceroute para diagnosticar problemas de conexión**:
     - **Ejemplo**: Si un servidor web no responde, puedes usar ping para verificar si el servidor está accesible y traceroute para identificar si hay algún nodo intermedio causando problemas.

       ```
       ping example.com
       traceroute example.com
       ```

     Esto te ayudará a determinar si el problema es de conectividad general o si se encuentra en un punto específico de la red.

2. **Captura de Tráfico de Red**:
   - **Uso de tcpdump para capturar tráfico en una interfaz y analizarlo con herramientas como Wireshark**:
     - **Ejemplo**: Para capturar todo el tráfico HTTP en la interfaz eth0 y guardarlo en un archivo para su análisis posterior.

       ```
       tcpdump -i eth0 port 80 -w http_traffic.pcap
       ```

     El archivo http_traffic.pcap puede ser abierto con Wireshark para un análisis detallado de las comunicaciones HTTP.

3. **Configuración de Interfaces de Red**:
   - **Mostrar cómo configurar una dirección IP estática usando ip**:
     - **Ejemplo**: Configurar una dirección IP estática 192.168.1.100 en la interfaz eth0.

```
ip addr add 192.168.1.100/24 dev eth0
ip link set dev eth0 up
```

Esto configurará la interfaz eth0 con la dirección IP 192.168.1.100 y la activará.

3. Herramientas de Red en Windows (CMD y PowerShell)

Este apartado está diseñado para ofrecer una visión detallada de las herramientas de red disponibles en Windows a través de CMD y PowerShell. Los alumnos aprenderán cómo utilizar estas herramientas para diagnosticar problemas de red, configurar interfaces, gestionar reglas de firewall, y más.

3.1. Introducción a las Herramientas de Red en Windows

- **Descripción**:
  - Windows proporciona un conjunto robusto de herramientas de red accesibles desde CMD y PowerShell. CMD es la interfaz de línea de comandos tradicional en Windows, mientras que PowerShell es una herramienta más avanzada que ofrece un entorno de scripting potente con capacidades adicionales.
  - Estas herramientas son esenciales para administradores de sistemas y profesionales de seguridad que necesitan gestionar configuraciones de red, diagnosticar problemas de conectividad, y asegurar redes en entornos Windows.

3.2. Herramientas Principales en CMD

1. **ping**:
   - **Uso**: Verificar la conectividad entre el host y otro dispositivo en la red, al igual que en Linux.
   - **Descripción**: Envía paquetes ICMP "echo request" para determinar si un host es accesible y medir la latencia de la conexión.
   - **Ejemplo**:
     ```
     ping google.com
     ```
     Este comando verifica la conectividad con google.com, mostrando el tiempo de respuesta y las estadísticas de los paquetes.

2. **ipconfig**:
   - **Uso**: Mostrar y gestionar la configuración IP del sistema.
   - **Descripción**: ipconfig muestra la configuración actual de las interfaces de red, incluyendo direcciones IP, máscaras de subred, y puertas de enlace predeterminadas.
   - **Ejemplo**:
     ```
     ipconfig /all
     ```
     Muestra una lista detallada de todas las interfaces de red y sus configuraciones IP.

3. **netstat**:
   - **Uso**: Mostrar conexiones de red activas, puertos en uso, y estadísticas de red.
   - **Descripción**: netstat es útil para monitorear el tráfico de red y ver qué puertos están abiertos o en uso por aplicaciones.
   - **Ejemplo**:
     ```
     netstat -an
     ```
     Muestra todas las conexiones de red activas y los puertos en escucha en el sistema.
4. **tracert**:
   - **Uso**: Rastrear la ruta que siguen los paquetes a través de la red, similar a traceroute en Linux.
   - **Descripción**: tracert muestra cada salto que un paquete realiza para llegar a su destino, lo que es útil para diagnosticar problemas de ruta en la red.
   - **Ejemplo**:
     ```
     tracert google.com
     ```
     Muestra la ruta que sigue un paquete desde el origen hasta google.com.

5. **nslookup**:
   - **Uso**: Resolver nombres de dominio y obtener detalles DNS, al igual que en Linux.
   - **Descripción**: nslookup permite consultar servidores DNS para resolver nombres de dominio o direcciones IP.
   - **Ejemplo**:
     ```
     nslookup google.com
     ```
     Muestra la dirección IP asociada con google.com, así como información sobre el servidor DNS utilizado para la consulta.
6. **route**:
   - **Uso**: Mostrar o modificar la tabla de enrutamiento de IP.
   - **Descripción**: route permite ver y modificar la tabla de rutas del sistema, determinando cómo los paquetes IP son dirigidos a través de las interfaces de red.
   - **Ejemplo**:
     ```
     route print
     ```
     Muestra la tabla de enrutamiento actual del sistema.

7. **netsh**:
    - **Uso**: Configurar y monitorizar interfaces de red, firewall, y otras configuraciones de red.
    - **Descripción**: netsh es una herramienta poderosa que permite configurar la red local, incluyendo la administración de interfaces, políticas de seguridad, y configuraciones del firewall.
    - **Ejemplo**:
      ```
      netsh interface ip show config
      ```
      Muestra la configuración IP de todas las interfaces de red.

3.3. Herramientas Principales en PowerShell

1. **Test-Connection**:
   - **Uso**: Similar a ping, pero con más opciones de configuración y flexibilidad.
   - **Descripción**: Test-Connection envía solicitudes ICMP a un host y puede configurarse para realizar múltiples pruebas, medir el tiempo de ida y vuelta, y más.
   - **Ejemplo**:
     ```
     Test-Connection google.com
     ```
     Realiza un ping a google.com y muestra los resultados de la prueba de conectividad.
2. **Get-NetIPAddress**:
   - **Uso**: Mostrar la configuración IP detallada del sistema.
   - **Descripción**: Get-NetIPAddress proporciona una vista detallada de todas las configuraciones IP en las interfaces de red, incluyendo direcciones IPv4 e IPv6.
   - **Ejemplo**:
     ```
     Get-NetIPAddress
     ```
     Muestra la configuración de todas las direcciones IP en el sistema.

3. **Get-NetRoute**:
   - **Uso**: Mostrar la tabla de enrutamiento de IP.
   - **Descripción**: Get-NetRoute permite ver las rutas configuradas en el sistema, similar a route print en CMD.
   - **Ejemplo**:
     ```
     Get-NetRoute
     ```
     Muestra la tabla de enrutamiento actual del sistema.
4. **Resolve-DnsName**:
   - **Uso**: Resolver nombres de dominio, similar a nslookup.
   - **Descripción**: Resolve-DnsName es una herramienta avanzada para realizar consultas DNS, permitiendo opciones adicionales y mayor control sobre la resolución.
   - **Ejemplo**:
     ```
     Resolve-DnsName google.com
     ```
     Muestra la resolución DNS de google.com, incluyendo múltiples registros si están disponibles.

5. **New-NetFirewallRule**:
   - **Uso**: Crear reglas de firewall para permitir o bloquear tráfico.
   - **Descripción**: New-NetFirewallRule permite crear reglas de firewall desde PowerShell, proporcionando un control granular sobre el tráfico de red.
   - **Ejemplo**:

```
New-NetFirewallRule -DisplayName "Allow HTTP" -Direction Inbound -Protocol
TCP -LocalPort 80 -Action Allow
```

   Crea una regla de firewall que permite el tráfico HTTP entrante en el puerto 80.

3.4. Ejemplos Prácticos

1. **Verificación de Configuración de Red**:
   - **CMD**:
     - Uso de ipconfig para verificar la configuración de la red.
       ```
       ipconfig /all
       ```
   - **PowerShell**:
     - Uso de Get-NetIPAddress para mostrar la configuración IP.
       ```
       Get-NetIPAddress
       ```
2. **Diagnóstico de Red Avanzado**:
   - **CMD**:
     - Uso de tracert para rastrear la ruta de los paquetes.
       ```
       tracert google.com
       ```
   - **PowerShell**:
     - Uso de Test-Connection para verificar la conectividad con parámetros personalizados.
       ```
       Test-Connection google.com -Count 10 -BufferSize 128
       ```

     Envía 10 paquetes con un tamaño de buffer de 128 bytes a google.com.

3. **Configuración del Firewall**:
    - **CMD**:
        - Uso de netsh para configurar una regla de firewall.

```
netsh advfirewall firewall add rule name="Allow HTTP" protocol=TCP dir=in
localport=80 action=allow
```

    - **PowerShell**:
        - Uso de New-NetFirewallRule para crear una regla de firewall.

```
New-NetFirewallRule -DisplayName "Allow HTTP" -Direction Inbound -Protocol
TCP -LocalPort 80 -Action Allow
```

4. Comparación entre Linux y Windows

Este apartado se centra en comparar las herramientas de red disponibles en Linux y Windows, destacando las diferencias en los enfoques, las ventajas y desventajas de cada sistema operativo en el contexto de la administración de redes y la seguridad informática.

4.1. Enfoques Diferentes

Linux:

- **Alta Flexibilidad y Control**:
  - **Descripción**: Linux es conocido por su flexibilidad y el control detallado que proporciona a los usuarios a través de sus herramientas de línea de comandos. Los administradores de sistemas pueden realizar tareas de red con gran precisión, ya que casi todas las configuraciones y comandos pueden ser ajustados según las necesidades específicas.
  - **Ejemplo**:
    - **Personalización de Scripts**: Un administrador puede crear un script bash personalizado para automatizar tareas de monitoreo de red usando herramientas como ping, netstat, y tcpdump. Por ejemplo:

```
# Script para monitorear la conectividad a un servidor web
# Guardar como monitor.sh
while true; do
    if ping -c 1 google.com &> /dev/null
    then
```

```
        echo "Conectado a Google"
    else
        echo "Fallo en la conexión"
    fi
    sleep 60
done
```

- Este script realiza un ping a Google cada 60 segundos, notificando si la conexión falla.

Windows:

- **Herramientas Gráficas y de Línea de Comandos**:
  - **Descripción**: Windows ofrece una combinación de herramientas gráficas y de línea de comandos. Mientras que CMD proporciona comandos básicos similares a los de Linux, PowerShell extiende las capacidades con un entorno de scripting avanzado que permite tareas complejas de gestión de redes.
  - **Ejemplo**:
    - **Uso de PowerShell para Diagnóstico de Red**:

```
# Comando en PowerShell para verificar múltiples destinos
$hosts = @("google.com", "microsoft.com", "github.com")
foreach ($equipo in $hosts) {
    Test-Connection -ComputerName $equipo -Count 10
}
```

- Este script en PowerShell realiza pruebas de conectividad a varios hosts y muestra los resultados de cada uno.

4.2. Ventajas y Desventajas

Linux:

- **Ventajas**:
  - **Capacidad de Automatización y Personalización**:
    - **Descripción**: Linux permite una automatización avanzada a través de scripts y cron jobs, lo que es ideal para tareas de mantenimiento de redes, auditorías, y configuraciones repetitivas.
    - **Ejemplo**: Usar un script bash para programar la limpieza de logs de red automáticamente, reduciendo así la posibilidad de sobrecarga en los sistemas.

```
# Script para limpiar logs de red cada semana
find /var/log/ -name "*.log" -type f -mtime +7 -exec rm -f {} \;
```

- **Desventajas**:
  - **Mayor Curva de Aprendizaje**:
    - **Descripción**: Linux puede ser intimidante para los usuarios que no están familiarizados con la terminal. La falta de una interfaz gráfica para ciertas tareas hace que el aprendizaje sea más lento y requiera una mayor inversión de tiempo.
    - **Ejemplo**: Configurar iptables para gestionar reglas de firewall requiere un conocimiento profundo de cómo funcionan las cadenas y políticas de red en Linux.

Windows:

- **Ventajas**:
  - **Facilidad de Uso con Interfaces Gráficas**:
    - **Descripción**: Windows ofrece muchas herramientas de red que pueden ser gestionadas a través de interfaces gráficas, lo que es más accesible para usuarios menos técnicos.
    - **Ejemplo**: Configurar un firewall utilizando la "Consola de Firewall de Windows" es intuitivo y no requiere conocimientos de línea de comandos.
  - **Entorno de Scripting en PowerShell**:
    - **Descripción**: PowerShell no solo ofrece una línea de comandos avanzada, sino que también integra un potente entorno de scripting que permite la automatización y gestión de tareas de red complejas con mayor facilidad.
    - **Ejemplo**: Crear un script en PowerShell para auditar las reglas de firewall y generar un reporte en formato CSV:

```
Get-NetFirewallRule | Select-Object DisplayName, Enabled,
Direction, Action, Profile | Export-Csv -Path
"firewall_rules_report.csv"
```

- **Desventajas**:
  - **Menor Flexibilidad en Algunas Herramientas**:
    - **Descripción**: Aunque PowerShell es muy avanzado, algunas herramientas nativas de Windows pueden ser menos flexibles en comparación con las

de Linux. Por ejemplo, herramientas como netsh o ipconfig tienen menos opciones de personalización que sus contrapartes en Linux.

- **Ejemplo**: La configuración de redes avanzadas como el enrutamiento o la manipulación de paquetes puede ser más limitada en CMD en comparación con Linux.

Resumen:

- **Linux** es ideal para usuarios que buscan máxima personalización y control sobre las redes, pero requiere un mayor esfuerzo de aprendizaje.
- **Windows**, por otro lado, equilibra la facilidad de uso con poderosas capacidades de scripting a través de PowerShell, aunque a veces puede ofrecer menos flexibilidad en comparación con Linux.

5. Casos Prácticos y Ejemplos

Este apartado está diseñado para mostrar cómo aplicar las herramientas de red en Linux y Windows a situaciones reales. Los casos prácticos proporcionan un enfoque práctico para que los estudiantes comprendan cómo diagnosticar problemas de red y configurar interfaces utilizando las herramientas disponibles en ambos sistemas operativos.

5.1. Diagnóstico de Red en Linux

Escenario: Un servidor no responde a las solicitudes de red. El objetivo es identificar el problema utilizando las herramientas de red en Linux.

1. **Verificación de Conectividad con ping:**
   - **Descripción**: ping se utiliza para comprobar si el servidor está alcanzable y responderá a las solicitudes ICMP.
   - **Comando:**

     ```
     ping -c 4 192.168.1.100
     ```

   - **Interpretación**:
     - Si las respuestas de ping son exitosas, el servidor es alcanzable, y el problema podría estar en un nivel superior (como un servicio específico).
     - Si no hay respuesta, podría haber un problema de conectividad, como una interrupción en la red o un firewall bloqueando las solicitudes.

2. **Rastreo de la Ruta con traceroute:**
   - **Descripción**: traceroute muestra la ruta que siguen los paquetes desde el origen hasta el destino, lo que puede ayudar a identificar en qué punto de la red se está produciendo un fallo.
   - **Comando:**
     ```
     traceroute 192.168.1.100
     ```
   - **Interpretación**:
     - Se revisa cada salto (router) que los paquetes atraviesan hasta llegar al destino. Si el rastreo se interrumpe en algún punto, ese podría ser el lugar donde ocurre el problema.
3. **Captura y Análisis de Tráfico con tcpdump:**
   - **Descripción**: tcpdump captura y analiza paquetes de red en tiempo real, lo que puede ser útil para observar si el servidor está recibiendo tráfico y cómo responde.
   - **Comando:**
     ```
     sudo tcpdump -i eth0 host 192.168.1.100
     ```
   - **Interpretación**:
     - Se examina el tráfico entrante y saliente para el servidor problemático. Si no hay tráfico, podría indicar un problema en la red o en la configuración de la interfaz del servidor.

4. **Resolución del Problema:**
    - Según los resultados de ping, traceroute, y tcpdump, se puede determinar si el problema es de conectividad, enrutamiento o configuración de red. A partir de aquí, se podrían tomar medidas adicionales, como revisar las reglas de firewall (iptables) o la configuración de red (ip).

5.2. Configuración de Red en Windows

Escenario: Configurar y asegurar una interfaz de red en un servidor Windows utilizando ipconfig y netsh.

1. **Verificación de Configuración de Red con ipconfig:**
   - **Descripción**: ipconfig proporciona detalles sobre la configuración IP actual de las interfaces de red del servidor.
   - **Comando:**
     ```
     ipconfig /all
     ```
   - **Interpretación**:
     - Se revisa la dirección IP, la máscara de subred, la puerta de enlace predeterminada, y las direcciones DNS para asegurarse de que estén configuradas correctamente.
2. **Configuración de una Dirección IP Estática con netsh:**
   - **Descripción**: netsh se utiliza para configurar diversos aspectos de la red, incluyendo la asignación de una dirección IP estática a una interfaz específica.
   - **Comando:**
     ```
     netsh interface ip set address "Ethernet0" static 192.168.1.50
     255.255.255.0 192.168.1.1
     ```
   - **Explicación**:
     - **"Ethernet0"**: Es el nombre de la interfaz de red.
     - **192.168.1.50**: Es la dirección IP asignada.

- **255.255.255.0**: Es la máscara de subred.
- **192.168.1.1**: Es la puerta de enlace predeterminada.

3. **Asegurar la Interfaz de Red con netsh:**
   - **Descripción**: Para mejorar la seguridad, se puede utilizar netsh para configurar el firewall y limitar el tráfico entrante y saliente a la interfaz.
   - **Comando para Permitir Solo el Tráfico HTTP Entrante:**
     ```
     netsh advfirewall firewall add rule name="Allow HTTP" protocol=TCP
     dir=in localport=80 action=allow
     ```
   - **Comando para Bloquear Todo el Tráfico Excepto HTTP:**
     ```
     netsh advfirewall set allprofiles firewallpolicy
     blockinbound,allowoutbound
     netsh advfirewall firewall add rule name="Allow HTTP" protocol=TCP
     dir=in localport=80 action=allow
     ```
4. **Verificación Final:**
   - **Descripción**: Después de la configuración, es esencial verificar que los cambios se hayan aplicado correctamente.
   - **Comando:**
     ```
     ipconfig /all
     ```
   - **Interpretación**:
     - Revisar la configuración para asegurarse de que la IP, la máscara de subred, y la puerta de enlace están correctamente configuradas.

- Utilizar ping o Test-Connection (en PowerShell) para comprobar la conectividad con otros dispositivos en la red.

6. Ejercicios

Ejercicios Básicos para Linux

Ejercicio 1: Verificar Conectividad con ping

- **Objetivo**: Verificar si el servidor DNS de Google es alcanzable desde tu máquina.
- **Instrucciones**: Utiliza el comando ping para verificar la conectividad con la dirección IP 8.8.8.8.
- **Comando**:

```
ping -c 4 8.8.8.8
```

- **Solución**: Deberías recibir cuatro respuestas desde 8.8.8.8, lo que indica que el servidor es alcanzable.

Ejercicio 2: Consultar Configuración de Red con ip

- **Objetivo**: Mostrar la configuración IP de todas las interfaces de red en tu sistema.
- **Instrucciones**: Usa el comando ip para mostrar la configuración de red.
- **Comando**:

```
ip addr show
```

- **Solución**: Deberías ver una lista de todas las interfaces de red con sus respectivas configuraciones IP.

Ejercicio 3: Resolver un Nombre de Dominio con nslookup

- **Objetivo**: Resolver la dirección IP del dominio example.com.
- **Instrucciones**: Usa nslookup para obtener la dirección IP.
- **Comando**:

```
nslookup example.com
```

- **Solución**: El comando devolverá la dirección IP asociada a example.com.

Ejercicio 4: Listar Conexiones de Red Activas con netstat

- **Objetivo**: Listar todas las conexiones de red activas en tu máquina.
- **Instrucciones**: Utiliza netstat para listar las conexiones.
- **Comando**:

```
netstat -an
```

- **Solución**: Deberías ver una lista de todas las conexiones de red activas, junto con su estado.

Ejercicio 5: Rastrear la Ruta con traceroute

- **Objetivo**: Identificar la ruta que sigue un paquete para llegar a google.com.
- **Instrucciones**: Utiliza traceroute para rastrear la ruta.
- **Comando**:

```
traceroute google.com
```

- **Solución**: El comando debería mostrar cada salto (router) que los paquetes atraviesan para llegar a google.com.

Ejercicios Básicos para Windows

Ejercicio 1: Verificar Conectividad con ping

- **Objetivo**: Verificar si el servidor DNS de Google es alcanzable desde tu máquina.
- **Instrucciones**: Utiliza el comando ping para verificar la conectividad con la dirección IP 8.8.8.8.
- **Comando**:

```
ping 8.8.8.8
```

- **Solución**: Deberías recibir cuatro respuestas desde 8.8.8.8, lo que indica que el servidor es alcanzable.

Ejercicio 2: Mostrar Configuración de Red con ipconfig

- **Objetivo**: Mostrar la configuración IP detallada de tu máquina.
- **Instrucciones**: Utiliza ipconfig para ver la configuración.
- **Comando**:

```
ipconfig /all
```

- **Solución**: Deberías ver información detallada sobre todas las interfaces de red.

Ejercicio 3: Resolver un Nombre de Dominio con nslookup

- **Objetivo**: Resolver la dirección IP del dominio example.com.
- **Instrucciones**: Usa nslookup para obtener la dirección IP.
- **Comando**:

```
nslookup example.com
```

- **Solución**: El comando devolverá la dirección IP asociada a example.com.

Ejercicio 4: Rastrear la Ruta con tracert

- **Objetivo**: Identificar la ruta que sigue un paquete para llegar a google.com.
- **Instrucciones**: Utiliza tracert para rastrear la ruta.
- **Comando**:

```
tracert google.com
```

- **Solución**: El comando debería mostrar cada salto (router) que los paquetes atraviesan para llegar a google.com.

Ejercicio 5: Listar Conexiones de Red Activas con netstat

- **Objetivo**: Listar todas las conexiones de red activas en tu máquina.
- **Instrucciones**: Utiliza netstat para listar las conexiones.
- **Comando**:

```
netstat -an
```

- **Solución**: Deberías ver una lista de todas las conexiones de red activas, junto con su estado.

Ejercicios Intermedios para Linux

Ejercicio 1: Capturar Tráfico con tcpdump

- **Objetivo**: Capturar paquetes ICMP en la interfaz eth0.
- **Instrucciones**: Usa tcpdump para capturar solo paquetes ICMP.
- **Comando**:

```
sudo tcpdump -i eth0 icmp
```

- **Solución**: Deberías ver un flujo de paquetes ICMP capturados en la interfaz eth0.

Ejercicio 2: Configurar una Dirección IP Estática con ip

- **Objetivo**: Configurar una dirección IP estática en la interfaz eth0.
- **Instrucciones**: Configura la IP 192.168.1.50/24 con puerta de enlace 192.168.1.1.
- **Comando**:

```
sudo ip addr add 192.168.1.50/24 dev eth0
sudo ip route add default via 192.168.1.1
```

- **Solución**: La interfaz eth0 debería estar configurada con la IP y la puerta de enlace especificadas.

Ejercicio 3: Escanear la Red Local con nmap

- **Objetivo**: Escanear todos los dispositivos activos en la red 192.168.1.0/24.
- **Instrucciones**: Usa nmap para realizar un escaneo de la red local.
- **Comando**:

```
sudo nmap -sP 192.168.1.0/24
```

- **Solución**: Deberías ver una lista de dispositivos activos en la red 192.168.1.0/24.

Ejercicio 4: Configurar Reglas de Firewall con iptables

- **Objetivo**: Bloquear todo el tráfico entrante en la interfaz eth0 excepto el puerto 80.
- **Instrucciones**: Configura iptables para bloquear todo el tráfico entrante excepto HTTP.
- **Comando**:

```
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
sudo iptables -A INPUT -i eth0 -j DROP
```

- **Solución**: Solo el tráfico HTTP en el puerto 80 debería ser permitido.

Ejercicio 5: Análisis de Puertos Abiertos con ss

- **Objetivo**: Listar todos los puertos TCP abiertos y escuchando en la máquina.
- **Instrucciones**: Usa ss para mostrar los puertos abiertos.
- **Comando**:

```
ss -tuln
```

- **Solución**: Deberías ver una lista de puertos TCP abiertos y sus direcciones de escucha.

Ejercicios Intermedios para Windows

Ejercicio 1: Diagnosticar Conectividad con Test-Connection en PowerShell

- **Objetivo**: Probar la conectividad con google.com enviando 5 paquetes ICMP.
- **Instrucciones**: Usa Test-Connection para realizar esta tarea.
- **Comando**:

```
Test-Connection google.com -Count 5
```

- **Solución**: Deberías recibir 5 respuestas indicando que la conectividad es correcta.

Ejercicio 2: Configurar una Dirección IP Estática con netsh

- **Objetivo**: Configurar una dirección IP estática en la interfaz Ethernet0.
- **Instrucciones**: Configura la IP 192.168.1.50/24 con puerta de enlace 192.168.1.1.
- **Comando**:

```
netsh interface ip set address "Ethernet0" static 192.168.1.50 255.255.255.0
192.168.1.1
```

- **Solución**: La interfaz Ethernet0 debería estar configurada con la IP y la puerta de enlace especificadas.

Ejercicio 3: Configurar una Regla de Firewall con New-NetFirewallRule en PowerShell

- **Objetivo**: Crear una regla de firewall que permita solo el tráfico HTTP entrante.
- **Instrucciones**: Usa PowerShell para crear esta regla.
- **Comando**:

```
New-NetFirewallRule -DisplayName "Allow HTTP" -Direction Inbound -Protocol TCP
-LocalPort 80 -Action Allow
```

- **Solución**: La regla de firewall debería permitir solo tráfico HTTP en el puerto 80.

Ejercicio 4: Mostrar la Tabla de Enrutamiento con route

- **Objetivo**: Ver la tabla de enrutamiento actual en tu máquina.
- **Instrucciones**: Usa route para mostrar la tabla de enrutamiento.
- **Comando**:

```
route print
```

- **Solución**: Deberías ver una tabla detallada de rutas IP configuradas en tu máquina.

Ejercicio 5: Escanear Puertos Abiertos en la Red Local con Test-Port (PowerShell)

- **Objetivo**: Verificar si el puerto 22 está abierto en un servidor específico de la red local.
- **Instrucciones**: Usa PowerShell para verificar el estado del puerto.
- **Comando**:

```
Test-NetConnection -ComputerName 192.168.1.100 -Port 22
```

- **Solución**: Deberías recibir una respuesta indicando si el puerto 22 está abierto o cerrado.