Práctica 2: Descubrimiento y Enumeración de Redes — De Python a Nmap

Clase de Hacking Ético

24 de octubre de 2025

1. Introducción: El Arte del Reconocimiento Digital

1.1. Objetivos de la Práctica

Esta sesión práctica está diseñada para proporcionar una comprensión fundamental y aplicada de una de las fases más críticas en la metodología del hacking ético: el escaneo y la enumeración. Al finalizar este laboratorio de dos horas, el estudiante será capaz de:

- Comprender el rol estratégico del escaneo de puertos como puente entre el reconocimiento pasivo y la explotación activa.
- Desarrollar un escáner de puertos funcional desde cero utilizando Python, para asimilar los mecanismos de red subyacentes.
- Adquirir competencia en el uso de Nmap, la herramienta estándar de la industria, para realizar tareas de descubrimiento y enumeración a nivel profesional.
- Aprender a interpretar los resultados de un escaneo para identificar servicios, versiones y, en última instancia, posibles vectores de ataque.

1.2. Contexto: ¿Por qué Escanear?

En la metodología de hacking, el escaneo y la enumeración son los pasos que siguen al reconocimiento inicial. Esta fase es donde la información abstracta, como nombres de dominio o rangos de direcciones IP, se transforma en inteligencia técnica accionable. El objetivo es construir un mapa detallado de la superficie de ataque del objetivo, identificando cada "puertaz "ventana"digital. Para ello, es esencial comprender algunos conceptos clave:

- **Puertos TCP/UDP:** Se pueden visualizar como las puertas numeradas de un servidor. Cada servicio (como un sitio web o un servidor de correo) escucha en un puerto específico, esperando conexiones entrantes. Existen 65,535 puertos posibles para TCP y otros 65,535 para UDP.[1, 2]
- **Servicios:** Son los programas o demonios que se ejecutan en un servidor y .[es]cuchan.[en] un puerto determinado. Por ejemplo, un servidor web Apache normalmente escucha en el puerto 80 para tráfico HTTP.
- **Banners:** Cuando se establece una conexión con un servicio, este a menudo responde con un "banner", que es una cadena de texto que puede revelar su nombre y número de versión (por ejemplo, "vsftpd 2.3.4").

- **Estados de un Puerto:** Nmap clasifica los puertos en seis estados posibles: `open` (abierto), `closed` (cerrado), `filtered` (filtrado, un firewall impide el sondeo), `unfiltered` (no filtrado), `open|filtered` (abierto o filtrado), y `closed|filtered` (cerrado o filtrado). Comprender estos estados es crucial para una interpretación precisa de los resultados.

La estructura de esta práctica, comenzando con la construcción manual de un escáner en Python antes de pasar a la sofisticación de Nmap, es deliberada. Un principiante podría ver Nmap como una çaja mágica"que produce resultados, sin comprender qué ocurre a nivel de red. Esta falta de conocimiento fundamental es peligrosa, ya que impide la correcta interpretación de resultados ambiguos y la adaptación de técnicas a escenarios complejos. Al construir primero el mecanismo de escaneo manualmente con Python, que se basa en establecer una conexión TCP completa (el *three-way handshake*) [3, 4], el estudiante experimenta las limitaciones de un enfoque simple y ruidoso". Posteriormente, al introducir el escaneo SYN de Nmap, que es más sigiloso porque no completa dicho handshake [5, 6], la diferencia no es solo teórica, sino tangible. Esta progresión transforma al estudiante de un mero .ºperador de herramientas.ª un analista que comprende el proceso a nivel de paquetes, apreciando el porqué de la eficiencia y el sigilo en las pruebas de penetración.

1.3. ADVERTENCIA ÉTICA Y LEGAL: ZONA DE PRUEBAS CONTROLADA

¡MUY IMPORTANTE!

El escaneo de puertos, aunque es una herramienta fundamental para los profesionales de la seguridad, puede ser interpretado como un acto hostil y es ilegal en muchas jurisdicciones si se realiza sin el consentimiento explícito y por escrito del propietario del sistema objetivo.[7, 8] Realizar un escaneo no autorizado puede resultar en consecuencias académicas, civiles y penales.
Para esta práctica, **TODAS las actividades de escaneo deben realizarse EXCLUSIVAMENTE contra los siguientes objetivos autorizados:**

1. La máquina virtual **Metasploitable** que configurará en su entorno de laboratorio.
2. El dominio `scanme.nmap.org`, un servicio proporcionado por los creadores de Nmap específicamente para fines de prueba y aprendizaje.

Está terminantemente prohibido escanear cualquier otro sistema, incluyendo la red de su universidad, su red doméstica, o cualquier otro sitio en Internet.

1.4. Configuración del Entorno de Laboratorio

Para garantizar un entorno de trabajo seguro y aislado, es necesario configurar un laboratorio virtual.

1. **Descargue e importe Metasploitable 2:** Esta es una máquina virtual Linux intencionadamente vulnerable, diseñada para la práctica de hacking ético.[1, 9] Puede descargarla desde fuentes oficiales como SourceForge.
2. **Configuración de Red:** En VMWare, asegúrese de que tanto su máquina virtual Kali Linux como la máquina virtual Metasploitable estén configuradas en la misma red. Se recomienda usar el modo "Host-only.º una "NAT Network"dedicada. Esto crea una red privada entre sus máquinas virtuales, aislándolas de su red principal y de Internet.

3. **Identificación de IPs:** Inicie ambas máquinas virtuales. En cada una, abra una terminal y ejecute el comando `ip a` para identificar sus respectivas direcciones IP. Anote la dirección IP de la máquina Metasploitable, ya que será su objetivo principal durante toda la práctica.

2. Parte 1: Anatomía de un Escáner — Construcción con Python

2.1. Fundamentos de Sockets y el Handshake de Tres Vías (TCP Three-Way Handshake)

Toda comunicación TCP fiable, desde la navegación web hasta el envío de un correo electrónico, comienza con un proceso llamado "handshake de tres vías". Este es un diálogo de tres pasos entre el cliente y el servidor para establecer una conexión [10]:

1. **SYN (Synchronize):** El cliente envía un paquete TCP con el flag SYN activado al servidor, solicitando iniciar una comunicación.
2. **SYN-ACK (Synchronize-Acknowledge):** Si el puerto en el servidor está abierto y escuchando, responde con un paquete que tiene activados los flags SYN y ACK, reconociendo la solicitud del cliente y proponiendo también su sincronización.
3. **ACK (Acknowledge):** El cliente responde con un paquete ACK, confirmando la recepción del SYN-ACK del servidor. En este punto, la conexión se considera establecida y estable.

Nuestro script de Python utilizará el módulo `socket` [11, 12], que es la interfaz de bajo nivel del sistema operativo para la comunicación en red. Le pediremos al sistema operativo que realice este handshake completo por nosotros para cada puerto que queramos probar.

2.2. Paso 1: Creando el Escáner Básico en VSCode (`scanner_v1.py`)

```python
import socket

# --- Configuracion del Objetivo ---
target_ip = "192.168.X.X"  # REEMPLAZAR con la IP de Metasploitable
port_to_scan = 80

# --- Logica del Escaner ---
try:
    # Crear un objeto socket
    # AF_INET especifica que usaremos IPv4
    # SOCK_STREAM especifica que usaremos el protocolo TCP
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Establecer un tiempo de espera para la conexion (1 segundo)
    sock.settimeout(1)

    # Intentar conectar al objetivo. connect_ex() devuelve 0 si tiene exito.
    # Es preferible a connect() porque no lanza una excepcion en caso de fallo.
    result = sock.connect_ex((target_ip, port_to_scan))

    if result == 0:
        print(f"Puerto {port_to_scan}: Abierto")
    else:
        print(f"Puerto {port_to_scan}: Cerrado o Filtrado")

    # Cerrar el socket
    sock.close()

```

```
except socket.gaierror:
    print("Error: El nombre del host no pudo ser resuelto.")
except socket.error:
    print("Error: No se pudo conectar al servidor.")
```

Listing 1: scanner_v1.py

Este script utiliza `socket.connect_ex((target, port))`. Esta función es ideal para un escáner porque, en lugar de generar un error (excepción) si la conexión falla, simplemente devuelve un código de error. Un valor de retorno de `0` indica que la conexión fue exitosa, lo que significa que el handshake de tres vías se completó y el puerto está abierto.[4, 13]

2.3. Paso 2: Escaneo de Múltiples Puertos (`scanner_v2.py`)

```
import socket

target_ip = "192.168.X.X"  # REEMPLAZAR con la IP de Metasploitable
common_ports =

print(f"Iniciando escaneo en {target_ip}...")

for port in common_ports:
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(0.5)  # Reducimos el timeout para mayor velocidad

        result = sock.connect_ex((target_ip, port))

        if result == 0:
            print(f"Puerto {port}: Abierto")

        sock.close()

    except socket.error:
        print(f"Error al escanear el puerto {port}")

print("Escaneo finalizado.")
```

Listing 2: scanner_v2.py

Este script introduce un bucle `for` para iterar sobre la lista `common_ports`, aplicando la misma lógica de conexión a cada uno.[2, 4] Es importante notar que este escaneo es secuencial: prueba un puerto, espera el resultado, y luego pasa al siguiente. Escanear los 65,535 puertos de esta manera sería extremadamente lento, una limitación que Nmap supera con técnicas de paralelización.[3]

2.4. Paso 3: Hacia una Herramienta Reutilizable (`scanner_v3.py`)

```
import socket
import sys

def scan_ports(target_ip, start_port, end_port):
    print(f"Iniciando escaneo en {target_ip} desde el puerto {start_port} hasta
    {end_port}...")

    for port in range(start_port, end_port + 1):
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.settimeout(0.5)

            result = sock.connect_ex((target_ip, port))
```

```
13
14              if result == 0:
15                  print(f"Puerto {port}: Abierto")
16
17              sock.close()
18
19          except socket.error:
20              print(f"Error al escanear el puerto {port}")
21
22      print("Escaneo finalizado.")
23
24  if __name__ == "__main__":
25      if len(sys.argv)!= 4:
26          print("Uso: python3 scanner_v3.py <IP_OBJETIVO> <PUERTO_INICIO> <
        PUERTO_FIN>")
27          sys.exit()
28
29      target = sys.argv[14]
30      start = int(sys.argv[1])
31      end = int(sys.argv[2])
32
33      scan_ports(target, start, end)
```

Listing 3: scanner_v3.py

Ahora puede ejecutar el script desde la terminal de la siguiente manera: `python3 scanner_v3.py 192.168.X.X 1 1000`.[4]

2.5. Análisis de Resultados y Limitaciones

Ejecute el script final contra su máquina Metasploitable, escaneando los primeros 1000 puertos. Observe la salida y el tiempo que tarda. Aunque funcional, la herramienta que ha construido es un artefacto de detección perfecto para cualquier sistema de seguridad moderno. Cada vez que encuentra un puerto abierto, completa un handshake TCP, una acción que es registrada por el sistema operativo del objetivo.[8] Un Sistema de Detección de Intrusos (IDS) o un firewall vería una avalancha de conexiones completas desde una única IP a múltiples puertos en un corto período de tiempo, un patrón clásico de un escaneo de puertos básico que dispararía una alerta inmediatamente.[6]

Esta limitación inherente —ser lento y extremadamente ruidoso"— hace que este tipo de escáner sea ineficaz en un entorno real y protegido. Sin embargo, comprender esta debilidad es fundamental. Enseña una lección crucial: la efectividad de una herramienta de hacking no se mide solo por si "funciona", sino por su capacidad para evadir la detección. Esta toma de conciencia es el trampolín perfecto para entender la filosofía detrás de las técnicas avanzadas de Nmap, como el escaneo sigiloso y las tácticas de evasión.[14, 1, 15]

3. Parte 2: Dominando el Terreno con Nmap

3.1. Introducción a Nmap: La Navaja Suiza del Hacker

Nmap (Network Mapper) es la herramienta de facto para el mapeo de redes y la auditoría de seguridad. Es una utilidad de código abierto inmensamente poderosa, flexible y bien documentada, utilizada por profesionales de la ciberseguridad en todo el mundo.[1, 9] Mientras que nuestro script de Python solo podía determinar si un puerto estaba abierto o cerrado mediante una conexión completa, Nmap puede hacer eso y mucho más, de forma más rápida, eficiente y, sobre todo, sigilosa.

3.2. Paso 4: Descubrimiento de Hosts y Escaneos Básicos

- **Descubrimiento de Hosts (Ping Scan):** `nmap -sn <RANGO_IP>` Este comando le dice a Nmap que realice un "Ping Scan". La opción `-sn` (anteriormente `-sP`) deshabilita el escaneo de puertos y se enfoca únicamente en descubrir qué hosts están en línea en el rango especificado.[15, 16, 17] *Ejemplo:* `nmap -sn 192.168.X.0/24` (reemplace con la subred de su laboratorio).

- **TCP Connect Scan:** `nmap -sT <IP_OBJETIVO>` Este es el equivalente en Nmap de nuestro script de Python. Realiza un handshake TCP completo con cada puerto. Es el escaneo por defecto cuando un usuario no tiene privilegios de administrador (root).[6, 17, 18] *Ejemplo:* `nmap -sT 192.168.X.X`

- **TCP SYN Scan (Stealth Scan):** `sudo nmap -sS <IP_OBJETIVO>` Este es el verdadero poder de Nmap. El escaneo SYN es conocido como .ºcaneo medio abierto"(*half-open scanning*). Nmap envía un paquete SYN, y si recibe un SYN-ACK, en lugar de completar el handshake con un ACK, envía un paquete RST (Reset), terminando la conexión. Como la conexión nunca se establece por completo, es mucho menos probable que sea registrada, haciéndolo más sigiloso y rápido.[5, 6, 8] Requiere privilegios de root (`sudo`). *Ejemplo:* `sudo nmap -sS 192.168.X.X`

3.3. Paso 5: Enumeración Profunda — ¿Qué se está ejecutando?

- **Detección de Servicios y Versiones:** `nmap -sV <IP_OBJETIVO>` La opción `-sV` instruye a Nmap para que intente determinar la versión del servicio que se ejecuta en cada puerto abierto.[5, 9] *Ejemplo:* `sudo nmap -sS -sV 192.168.X.X`

- **Detección del Sistema Operativo:** `sudo nmap -O <IP_OBJETIVO>` Nmap puede hacer una suposición educada sobre el sistema operativo del objetivo analizando su pila de red TCP/IP.[5, 9, 15] *Ejemplo:* `sudo nmap -sS -O 192.168.X.X`

- **El Escaneo Agresivo:** `nmap -A <IP_OBJETIVO>` La opción `-A` es un atajo que activa la detección de SO (`-O`), detección de versiones (`-sV`), escaneo con scripts por defecto (`-sC`) y un `traceroute`.[6, 9, 17] *Ejemplo:* `nmap -A 192.168.X.X`

3.4. Paso 6: Automatización con el Nmap Scripting Engine (NSE)

- **Scripts por Defecto:** `nmap -sC <IP_OBJETIVO>` Esta opción ejecuta un conjunto de scripts seguros y no intrusivos.[5, 9] *Ejemplo:* `nmap -sC 192.168.X.X`

- **Scripts de Vulnerabilidades:** `nmap --script=vuln <IP_OBJETIVO>` Este comando ejecuta todos los scripts en la categoría `vuln`, diseñados para buscar vulnerabilidades conocidas.[1, 9] *Ejemplo:* `sudo nmap -sV --script=vuln 192.168.X.X`

3.5. Paso 7: Control del Escaneo y Gestión de la Salida

- **Plantillas de Tiempo (Timing):** `nmap -T4 <IP_OBJETIVO>` Nmap ofrece plantillas de tiempo de `-T0` (muy lento) a `-T5` (muy rápido). `-T4` (agresivo) es bueno para redes rápidas.[6, 9, 17] *Ejemplo:* `nmap -A -T4 192.168.X.X`

- **Guardar Resultados:**

  - `-oN <archivo.txt>`: Formato Normal.[6, 19]
  - `-oX <archivo.xml>`: Formato XML.[19]
  - `-oA <nombre_base>`: Guarda en todos los formatos principales.[6, 19]

Ejemplo: `nmap -A -T4 192.168.X.X -oA metasploitable_scan`

3.6. Tabla de Referencia Rápida de Comandos Nmap

Cuadro 1: Referencia Rápida de Comandos Nmap

Categoría	**Comando (Flag)**	**Ejemplo de Uso**	**Descripción (¿Para qué sirve?)**
Selección de Objetivo	(ninguno)	`nmap 192.168.1.10`	Escanea una única IP.
	(CIDR)	`nmap 192.168.1.0/24`	Escanea toda la subred.
Descubrimiento	`-sn`	`nmap -sn 192.168.1.0/24`	Descubrimiento de hosts (Ping Scan). No escanea puertos.
	`-Pn`	`nmap -Pn 192.168.1.10`	Asume que el host está activo y escanea puertos (salta el descubrimiento). Útil si el objetivo bloquea pings.
Técnicas de Escaneo	`-sS`	`sudo nmap -sS 192.168.1.10`	TCP SYN Scan (Stealth). Rápido y sigiloso. Requiere root.
	`-sT`	`nmap -sT 192.168.1.10`	TCP Connect Scan. Más lento y ruidoso. No requiere root.
	`-sU`	`sudo nmap -sU 192.168.1.10`	UDP Scan. Lento, para servicios como DNS, SNMP.
Enumeración	`-p-`	`nmap -p- 192.168.1.10`	Escanea los 65,535 puertos TCP.
	`-sV`	`nmap -sV 192.168.1.10`	Detecta la versión de los servicios en puertos abiertos.
	`-O`	`sudo nmap -O 192.168.1.10`	Intenta detectar el Sistema Operativo del objetivo.
	`-A`	`nmap -A 192.168.1.10`	Escaneo Agresivo (-sV, -O, -sC, –traceroute).
NSE (Scripts)	`-sC`	`nmap -sC 192.168.1.10`	Ejecuta los scripts por defecto (seguros).
	`--script=<cat>`	`nmap --script=vuln 192.168.1.10`	Ejecuta todos los scripts de una categoría (e.g., `vuln`, `discovery`).
Rendimiento y Salida	`-T<0-5>`	`nmap -T4 192.168.1.10`	Establece la plantilla de tiempo (velocidad). T4 es agresivo.
	`-oN <file>`	`nmap... -oN scan.txt`	Guarda la salida en formato normal.
	`-oA <basename>`	`nmap... -oA scan_results`	Guarda la salida en todos los formatos principales.

4. Conclusión y Desafío

4.1. Síntesis: De Programador a Analista de Seguridad

A lo largo de esta práctica, ha transitado un camino crucial en el desarrollo de un hacker ético: desde la construcción de una herramienta básica hasta el dominio de un instrumento profesional. La comparación entre ambos enfoques revela una lección fundamental:

- **Python Scanner:** Es una herramienta pedagógica invaluable. Le obligó a comprender los fundamentos de la comunicación TCP y la lógica detrás de un escaneo de puertos. Sin embargo, en la práctica, es lento, ruidoso (fácilmente detectable) y funcionalmente limitado. Demuestra el *cómo* funciona un escaneo a nivel básico.
- **Nmap:** Es la herramienta del profesional. Es rápido, configurable para ser sigiloso (con `-sS`), y extraordinariamente versátil gracias a la detección de servicios/SO y al potente Nmap Scripting Engine. Demuestra el *qué* se puede lograr con herramientas especializadas y optimizadas durante décadas.

4.2. Resumen de Habilidades Adquiridas

- Programación de herramientas de red básicas con sockets de Python.
- Descubrimiento de hosts activos en una red.
- Escaneo de puertos TCP utilizando diferentes técnicas (Connect vs. SYN).
- Enumeración de servicios, versiones de software y sistemas operativos.
- Uso del Nmap Scripting Engine para la detección automatizada de información y vulnerabilidades.
- Gestión de la salida de Nmap para la elaboración de informes y análisis posteriores.

4.3. Desafío Opcional (Para Exploración Adicional)

Su escaneo con `nmap -A` en la máquina Metasploitable ha revelado una multitud de servicios abiertos. Elija uno de ellos que le parezca interesante (por ejemplo, vsftpd en el puerto 21, proftpd, o samba en los puertos 139/445).

1. Utilice el sistema de archivos de su máquina Kali para buscar scripts NSE específicos para ese servicio. Por ejemplo, puede usar el comando `ls /usr/share/nmap/scripts/*ftp*` para encontrar scripts relacionados con FTP.
2. Ejecute un nuevo escaneo de Nmap dirigido solo a ese puerto, pero esta vez utilizando uno o varios de los scripts específicos que encontró. Por ejemplo: `sudo nmap -p 21 --script=ftp-anon,ftp-vuln-cve2010-4221 192.168.X.X`.
3. Analice la salida. ¿Encuentra alguna información nueva o alguna vulnerabilidad específica que los scripts `vuln` hayan reportado? Investigue el nombre de la vulnerabilidad (por ejemplo, un código CVE) en Google o en bases de datos de exploits como Exploit-DB.

Este desafío le empuja a ir más allá de los comandos predefinidos, a explorar el NSE de forma independiente y a comenzar a pensar en el siguiente paso lógico del ciclo de hacking: la explotación.