🔧 Ejercicio Práctico: Escaneo de Subred con Detección de Servicios y Reporte

---

📝 Descripción

Este ejercicio consiste en automatizar un escaneo de red sobre una subred local, detectar los servicios que corren en puertos abiertos, identificar versiones de software y generar un resumen legible para ser utilizado en un reporte técnico.

🎯 Objetivos de aprendizaje

- Automatizar el escaneo de una red completa (ej. `192.168.1.0/24`).
- Detectar **puertos abiertos y versiones de servicios** (`-sV`).
- Identificar **hosts activos**.
- Estructurar e imprimir los resultados de forma clara.
- Reflexionar sobre los servicios expuestos y su posible impacto en la seguridad.

👨‍💻 Instrucciones

1. Asegúrate de tener instalado:
    - `nmap`
    - la librería `python-nmap`
2. Crea un script en Python que:

- Realice un escaneo con `-sV` sobre la subred `192.168.1.0/24` (puedes ajustar a la IP de tu entorno).
- Detecte y liste:
  - IP del host
  - Puerto abierto
  - Nombre del servicio
  - Versión del software si está disponible

3. Imprima un resumen ordenado por host, como si se preparara para un informe de pentesting.
4. Asegúrate de ejecutar este ejercicio **únicamente en redes de laboratorio o con autorización expresa**.

📌 Formato sugerido de salida:

Host: 192.168.1.10
- Puerto 22: ssh (OpenSSH 7.6)
- Puerto 80: http (Apache httpd 2.4.29)

Host: 192.168.1.15
- Puerto 139: netbios-ssn
- Puerto 445: microsoft-ds

Total de hosts activos: 2

⚖️ Consideraciones Éticas

- No se deben ejecutar escaneos en redes públicas ni entornos productivos sin consentimiento explícito.
- Este ejercicio es válido solo en entornos de prueba controlados.
- Documentar y validar con tu instructor o equipo antes de escanear redes compartidas.

✅ Solución: `escaneo_subred.py`

```
import nmap

# Crear un escáner de Nmap
scanner = nmap.PortScanner()

# Definir la subred objetivo (ajustar a tu entorno)
subred = "192.168.1.0/24"

print(f"🔍 Iniciando escaneo sobre la subred {subred}...\n")

# Ejecutar escaneo con detección de versión (-sV) sobre puertos comunes
scanner.scan(hosts=subred, arguments='-p 1-1024 -sV')

hosts_activos = scanner.all_hosts()
print(f"📡 Hosts activos detectados: {len(hosts_activos)}\n")

# Recorrer cada host encontrado
for host in hosts_activos:
    print(f"🖥️ Host: {host}")
    for proto in scanner[host].all_protocols():
        puertos = scanner[host][proto].keys()
        for puerto in sorted(puertos):
            info = scanner[host][proto][puerto]
            servicio = info.get('name', 'desconocido')
            version = info.get('version', '')
            producto = info.get('product', '')
            extra = f"{producto} {version}".strip()
            descripcion = f"{servicio} ({extra})" if extra else servicio
            print(f" - Puerto {puerto}: {descripcion}")
    print("-" * 40)

print("\n✅ Escaneo finalizado.")
```

🧪 Ejemplo de salida esperada (dependiendo de tu red):

```
🔍 Iniciando escaneo sobre la subred 192.168.1.0/24...

📡 Hosts activos detectados: 2

🖥️ Host: 192.168.1.10
 - Puerto 22: ssh (OpenSSH 7.9p1 Debian)
 - Puerto 80: http (Apache httpd 2.4.38)
----------------------------------------
🖥️ Host: 192.168.1.15
 - Puerto 445: microsoft-ds (Samba smbd 4.3.11-Ubuntu)
```

✅ Escaneo finalizado.

🔎 Puntos a destacar:

- El argumento `-sV` permite obtener nombre, producto y versión del servicio.
- El script genera una salida limpia y legible, ideal para incluir en informes técnicos.
- Se limita al rango de puertos 1-1024 para eficiencia y seguridad.
- Este tipo de escaneo es base para detectar software obsoleto o mal configurado.

⚠️ Advertencia legal y ética

Este tipo de exploración debe realizarse **únicamente en entornos autorizados**, laboratorios de práctica o redes propias. Su uso indebido en redes corporativas o externas sin consentimiento constituye una violación ética y legal.
