Avance 2 Programación Avanzada

Red Team:

scanner.py

```python
import nmap

def scan(target):
    nm = nmap.PortScanner()
    print(f"Escaneando {target}...")

    nm.scan(target, arguments="-sS -sV -Pn")

    # Guardar salida en archivo
    with open("resultado_scan.txt", "w") as f:
        f.write(nm.csv())

    # Mostrar resultados simples
    for host in nm.all_hosts():
        print(f"\nHost: {host} ({nm[host].hostname()})")
        for proto in nm[host].all_protocols():
            puertos = nm[host][proto].keys()
            for puerto in puertos:
                estado = nm[host][proto][puerto]["state"]
                print(f"Puerto {puerto}/{proto}: {estado}")

if __name__ == "__main__":
    objetivo = input("IP o dominio objetivo: ")
    scan(objetivo)
```

El script usa la librería nmap para escanear una IP o dominio. Primero crea un objeto PortScanner() y ejecuta un escaneo con los parámetros -sS -sV -Pn, que detectan puertos abiertos, servicios y versiones. Luego guarda la salida completa en un archivo resultado_scan.txt y muestra en pantalla los puertos encontrados, indicando su estado (open/closed).

```
IP o dominio objetivo: 20.109.21.115
Escaneando 20.109.21.115...

Host: 20.109.21.115 ()
Puerto 22/tcp: open
Puerto 8080/tcp: closed
```

El escaneo a la IP 20.109.21.115 encontró dos puertos: el 22/tcp aparece **open**, lo que indica que el servicio SSH está accesible. El puerto 8080/tcp aparece **closed**, lo que significa que el servidor no acepta conexiones en ese puerto. Esto confirma que la máquina remota tiene al menos un servicio activo expuesto.

Blue Team:

```python
import subprocess
import datetime

report_file = "os_audit_report.txt"

def run_cmd(cmd):
    try:
        result = subprocess.check_output(cmd, shell=True, text=True)
        return result.strip()
    except Exception as e:
        return f"Error ejecutando {cmd}: {e}"

def main():
    now = datetime.datetime.now()
    with open(report_file, "a") as f:
        f.write("\n============================\n")
        f.write(f" AUDITORÍA DEL SISTEMA - {now}\n")
        f.write("============================\n\n")

        f.write("== Información del sistema ==\n")
        f.write(run_cmd("systeminfo") + "\n\n" if subprocess.run("systeminfo", shell=True).returncode == 0
                else run_cmd("uname -a") + "\n\n")

        f.write("== Procesos corriendo ==\n")
        f.write(run_cmd("tasklist") + "\n\n" if subprocess.run("tasklist", shell=True).returncode == 0
                else run_cmd("ps aux") + "\n\n")

        f.write("== Puertos abiertos ==\n")
        f.write(run_cmd("netstat -ano") + "\n\n")

    print(f"[✓] Auditoría completada. Guardada en {report_file}")

if __name__ == "__main__":
    main()
```

Este código funciona como una herramienta básica de auditoría del sistema. Utiliza el módulo subprocess para ejecutar comandos del sistema operativo y datetime para registrar la fecha y hora de cada auditoría. La función run_cmd() se encarga de ejecutar los comandos y devolver su salida en texto, manejando posibles errores sin detener el programa. En la función principal, el script abre el archivo os_audit_report.txt en modo append y escribe un encabezado con la fecha actual, lo que permite mantener un historial de auditorías consecutivas en el mismo archivo. A continuación, detecta si el sistema es Windows o Linux ejecutando systeminfo o uname -a, respectivamente, para obtener información general del sistema.

Después recopila la lista de procesos en ejecución usando tasklist en Windows o ps aux en Linux. Finalmente, ejecuta netstat -ano, un comando compatible con ambos sistemas, para obtener información sobre puertos abiertos y conexiones de red activas. Toda esta información se guarda en el archivo de reporte para su revisión posterior.