📘 Trabajo Práctico Automatización de Reconocimiento, Escaneo y Explotación Ética de Aplicaciones Web con Python

---

🎯 Objetivo General

Diseñar, implementar y validar un conjunto de scripts en Python que permitan realizar de manera automatizada:

- el reconocimiento de una superficie de ataque web,
- el escaneo de servicios expuestos,
- la explotación ética de vulnerabilidades comunes (SQLi y XSS), y
- la generación de evidencias y reporte técnico,
  siguiendo principios de ética profesional, precisión técnica y documentación formal.

📌 Aprendizajes Esperados Evaluados

1. Aplicar técnicas de **reconocimiento activo y pasivo** para detectar puntos de entrada.
2. Automatizar escaneos de puertos y servicios con **herramientas integradas en Python**.
3. Desarrollar scripts funcionales para **detectar y validar SQLi y XSS** de manera ética.
4. Analizar y correlacionar **respuestas HTTP** como evidencia técnica.
5. Documentar hallazgos en un **reporte profesional automatizado y técnico**.
6. Integrar principios de **ética, legalidad y responsabilidad profesional** en todo el flujo de trabajo.

🧭 Instrucciones Generales

Trabaja en un entorno seguro de laboratorio (como OWASP Juice Shop, DVWA, bWAPP o testphp.vulnweb.com) y completa las siguientes etapas:

🔎 Etapa 1: Reconocimiento Automatizado de Superficie de Ataque

- Desarrolla un script que explore una URL base para:
    - Detectar rutas ocultas conocidas (`/admin`, `/config`, etc.)
    - Identificar formularios HTML y parámetros dinámicos.
- Utiliza `requests`, `BeautifulSoup` o similar.
- Extrae cabeceras HTTP relevantes para análisis.

Producto entregable: `reconocimiento.py` + log estructurado (JSON o texto).

🌐 Etapa 2: Escaneo Automatizado de Servicios

- Emplea `python-nmap` para escanear una red simulada o localhost.
- Detecta puertos abiertos, servicios, versiones.
- Estructura los resultados para futura correlación.

Producto entregable: `scanner.py` + resumen por host.

☠️ Etapa 3: Explotación Ética de Vulnerabilidades (SQLi y XSS)

- Construye dos scripts separados:
    1. `sqli_tester.py`: prueba múltiples payloads sobre parámetros vulnerables.
    2. `xss_checker.py`: simula ejecución de alertas en parámetros reflejados.

- Detecta e interpreta las respuestas: errores SQL, ejecuciones DOM, etc.

Producto entregable: scripts + tabla de resultados por vector probado.

🧠 Etapa 4: Validación y Control de Falsos Positivos

- Implementa lógica de comparación de respuestas (diferencias entre respuestas legítimas vs. manipuladas).
- Documenta cómo identificaste falsos positivos o negativos y qué ajustes realizaste.

Producto entregable: sección explicativa en el informe final.

📝 Etapa 5: Generación de Informe Profesional Automatizado

- Desarrolla un script (`report_generator.py`) que genere un reporte técnico en `.docx` o `.pdf`, con:
    - Sumario ejecutivo
    - Tabla de hallazgos
    - Evidencias (capturas, fragmentos de respuesta)
    - Recomendaciones técnicas
    - Nivel de criticidad

Producto entregable: archivo `.pdf` o `.docx` con marca de agua "Informe de Seguridad Ética Automatizada".

⚖️ Etapa 6: Reflexión Ética y Justificación Técnica

Incluye en el informe una **reflexión escrita** que responda a las siguientes preguntas:

- ¿Cómo garantizaste que tus pruebas fueron éticas y controladas?
- ¿Qué aprendiste sobre el poder y los límites de la automatización ofensiva?

- ¿Qué decisiones tomaste que reflejan responsabilidad profesional en ciberseguridad?

Producto entregable: sección final del informe titulada *"Reflexión Ética y Profesional"*.

🧪 Criterios de Evaluación (máximo 10 puntos)

1. **Funcionalidad del código (2.5 pts)**
   Los scripts funcionan correctamente y cumplen su objetivo.

2. **Originalidad y autonomía (2 pts)**
   El trabajo fue hecho de forma independiente, sin copiar ejemplos.

3. **Precisión técnica (1.5 pts)**
   Las pruebas detectan vulnerabilidades reales y evitan falsos positivos.

4. **Calidad del informe final (2 pts)**
   El reporte es claro, bien estructurado y con buenas recomendaciones.

5. **Reflexión ética (1 pt)**
   Se responde con seriedad a las preguntas sobre ética y responsabilidad.

6. **Documentación y buenas prácticas (1 pt)**
   El código está bien escrito, comentado y organizado.

📚 Recursos de Apoyo y Documentación Sugerida

A continuación, se entregan recursos clave para apoyar el desarrollo del proyecto:

📦 Librerías Python

- `requests`: Manejo de peticiones HTTP.
- `BeautifulSoup`: Extracción y análisis de contenido HTML.
- `python-nmap`: Wrapper de Nmap en Python para escaneo de puertos.

- `difflib`: Comparación de diferencias entre cadenas, útil para validación de respuestas.
- `python-docx`: Creación automatizada de reportes `.docx`.
- `pdfkit`: Generación de archivos PDF desde HTML (requiere instalación de `wkhtmltopdf`).

🧪 Laboratorios Seguros

- OWASP Juice Shop
- DVWA – Damn Vulnerable Web Application
- bWAPP – Buggy Web Application
- testphp.vulnweb.com

📘 Lecturas Complementarias

- OWASP Testing Guide
- The Hacker Playbook
- Python Offensive Security – Repositorio de payloads para pruebas.

🗒️ Reflexión Final

> **“La automatización ofensiva debe ser guiada por principios éticos, no solo por capacidad técnica. Comprender los límites legales, el impacto de cada prueba y la responsabilidad del profesional en ciberseguridad es clave para una práctica segura y útil.”**
