🛠️ Ejercicio Práctico

📌 **Título:** Simulación de Pruebas de Penetración Web con Burp Suite y OWASP DVWA

🎯 Objetivo del ejercicio:

Familiarizarse con la ejecución de pruebas básicas de penetración en aplicaciones web utilizando **Burp Suite** como proxy de interceptación y **DVWA** (Damn Vulnerable Web Application) como entorno vulnerable de práctica.

📝 Escenario:

Te han asignado la tarea de realizar un análisis básico de seguridad en una aplicación web vulnerable en un entorno de laboratorio. Para ello, utilizarás **Kali Linux**, donde tienes acceso a **Burp Suite** y a una instancia local de DVWA. El objetivo es interceptar una solicitud web y detectar una posible vulnerabilidad de inyección.

🔍 Tu tarea:

✅ Paso 1 – Configuración del entorno

1. Asegúrate de que DVWA esté instalado y corriendo localmente en `http://localhost/dvwa`.
2. Inicia **Burp Suite** desde Kali Linux.
3. Configura el navegador para que utilice el proxy local:
    - Dirección: `127.0.0.1`
    - Puerto: `8080`

✅ Paso 2 – Interceptar una solicitud web

1. Accede a DVWA desde el navegador y selecciona el módulo **SQL Injection**.
2. Introduce un valor como 1 en el campo de usuario y envía la solicitud.
3. Verifica que Burp Suite esté interceptando la petición.
4. Observa los parámetros enviados en la solicitud y anótalos.

✅ Paso 3 – Modificar el parámetro

1. Con la solicitud aún en Burp Suite, cambia el valor enviado por el siguiente payload:

```
1' OR '1'='1
```

2. Reenvía la solicitud modificada.
3. Observa el comportamiento de la aplicación (¿se devuelve más información de la esperada?).

✅ Paso 4 – Documentación de resultados

1. Anota:
    - El parámetro modificado
    - El resultado observado en la respuesta
    - Una interpretación del por qué ocurrió esa respuesta

✅ Paso 5 – Recomendación básica

1. Escribe una recomendación simple para mitigar este tipo de vulnerabilidad en una aplicación real.

🧪 Resultado esperado:

- Interceptación correcta de una solicitud HTTP con Burp Suite
- Modificación de parámetros en tiempo real
- Respuesta alterada por un payload de SQLi básico
- Documentación de hallazgos con interpretación lógica
- Recomendación clara y comprensible

🧠 Reflexión Final:

¿Qué aprendiste sobre la interceptación de solicitudes y la manipulación de parámetros?
¿Por qué es importante validar y sanitizar las entradas del usuario en aplicaciones web?
¿Cómo podrías aplicar esta metodología en un entorno profesional con autorización?

🛠️ Solución – Ejercicio Práctico

📌 Simulación de Pruebas de Penetración Web con Burp Suite y OWASP DVWA

🎯 Objetivo cumplido:

Se realizó exitosamente la interceptación de solicitudes web mediante **Burp Suite**, se manipuló un parámetro vulnerable y se comprobó la existencia de una posible inyección SQL en la aplicación DVWA. Se documentaron los hallazgos y se emitió una recomendación de mejora.

✅ Paso 1 – Configuración del entorno

- **Sistema operativo:** Kali Linux
- **Aplicación vulnerable:** DVWA (http://localhost/dvwa)
- **Proxy interceptante:** Burp Suite configurado en `127.0.0.1:8080`
- **Navegador configurado:** Firefox con proxy manual activo

✅ El entorno de pruebas se configuró correctamente y se logró acceder al módulo “SQL Injection” de DVWA.

✅ Paso 2 – Interceptación de solicitud

- Se accedió a la sección “SQL Injection” en DVWA.
- En el campo "User ID", se ingresó el valor `1` y se hizo clic en "Submit".
- Burp Suite interceptó la solicitud enviada por el navegador.

Parámetros observados en la solicitud:

GET /dvwa/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1

✅ La solicitud fue interceptada exitosamente y quedó en espera para su modificación.

✅ Paso 3 – Modificación del parámetro

Payload utilizado:

1' OR '1'='1

Solicitud modificada en Burp Suite:

GET /dvwa/vulnerabilities/sqli/?id=1'%20OR%20'1'%3D'1&Submit=Submit HTTP/1.1

✅ Se reenviaron los datos manipulados al servidor.

Resultado en la aplicación:

- Se listaron múltiples usuarios, lo que indica que la aplicación es vulnerable a inyecciones SQL básicas.

✅ Paso 4 – Documentación de resultados

Parámetro manipulado:
`id=1' OR '1'='1`

Resultado observado:
DVWA devolvió todos los registros de la tabla `users`, incluso sin autenticación específica.

Interpretación:
La aplicación no filtra ni valida adecuadamente las entradas del usuario, lo que permite modificar la lógica de la consulta SQL original.

✅ Paso 5 – Recomendación básica

🛡️ **Recomendación técnica:**
Para prevenir este tipo de vulnerabilidad, se debe:

- Implementar **consultas SQL preparadas** (prepared statements) en el backend
- Validar y sanitizar todas las entradas del usuario
- Evitar la concatenación directa de variables dentro de las consultas SQL

🧠 Reflexión Final

Este ejercicio me permitió comprender cómo un atacante puede manipular los parámetros HTTP con herramientas como Burp Suite.
La observación directa de los efectos de un payload SQLi básico demuestra la importancia de una validación robusta del lado del servidor.
Practicar estas técnicas en un entorno controlado fortalece las habilidades éticas de auditoría web y fomenta el pensamiento crítico en seguridad.
