




## Ejercicio Práctico

 **Título:** Detección y prueba manual de una vulnerabilidad XSS reflejado en DVWA

---

### **Objetivo del ejercicio:**

Identificar si un campo de entrada en una aplicación vulnerable refleja contenido ingresado por el usuario sin validación, y comprobar si es posible ejecutar un script JavaScript básico en el navegador de forma ética.

---

### **Escenario:**

Estás trabajando en la plataforma de prácticas **DVWA (Damn Vulnerable Web Application)** en modo **Low Security**. El módulo “XSS (Reflected)” simula un formulario vulnerable a ataques de scripting en el navegador. Tu tarea será detectar y demostrar si este formulario permite la ejecución de un **payload XSS reflejado**.

---

### **Actividades:**

---

#### **Paso 1 – Configurar el entorno**

1. Accede a DVWA desde <http://localhost/dvwa>.
  2. Inicia sesión con:
    - Usuario: `admin`
    - Contraseña: `password`
  3. Cambia el nivel de seguridad a **Low** desde la sección **DVWA Security**.
-

## ✓ Paso 2 – Ingresar al módulo XSS Reflejado

1. En el menú lateral izquierdo, selecciona **"XSS (Reflected)"**.
  2. Identifica el campo de entrada de texto y el botón de envío.
- 

## ✓ Paso 3 – Probar con un payload básico

1. En el campo de entrada escribe el siguiente código:

```
<script>alert('XSS')</script>
```

2. Presiona el botón "Submit" o "Enviar".
- 

## ✓ Paso 4 – Observar el resultado

- ¿Apareció una alerta emergente en el navegador?
  - ¿El código fue ejecutado como JavaScript?
  - ¿El campo reflejó exactamente lo ingresado, sin codificación?
- 

## Entregables:

1. Captura de pantalla de la alerta (si fue ejecutada).
  2. Breve explicación de por qué el navegador ejecutó el código.
  3. Conclusión sobre si la vulnerabilidad existe.
  4. Recomendación básica para prevenir este tipo de fallo en producción.
- 

## Preguntas de reflexión:

- ¿Qué hubiera pasado si el script malicioso robaba cookies en lugar de solo mostrar una alerta?
  - ¿Cómo puede un atacante usar esta vulnerabilidad para afectar a otros usuarios?
  - ¿Qué medidas debe tomar un desarrollador para evitar esta situación?
- 

## Solución Modelo – Ejercicio Práctico

### Prueba de vulnerabilidad XSS reflejado en DVWA

---

#### Paso 1 – Configuración del entorno

- Accedí correctamente a **DVWA** en <http://localhost/dvwa>.
  - Inicié sesión con:
    - **Usuario:** admin
    - **Contraseña:** password
  - Desde el menú **DVWA Security**, configuré el nivel de seguridad en **Low** para permitir la ejecución de pruebas de explotación básica.
- 

#### Paso 2 – Análisis del módulo vulnerable

- Navegué al módulo "**XSS (Reflected)**".
  - Identifiqué un campo de entrada de texto y un botón que al hacer clic envía el valor al servidor.
  - El valor ingresado se refleja directamente en la misma página, indicando posible falta de sanitización.
- 

#### Paso 3 – Prueba de vulnerabilidad

- Ingresé el siguiente payload en el campo de entrada:

```
<script>alert('XSS')</script>
```

- Al presionar el botón de envío, el navegador **ejecutó el script JavaScript**, mostrando una **ventana emergente** con el mensaje **XSS**.

---

## Resultado observado

- El código no fue codificado ni filtrado, sino reflejado tal cual en la respuesta HTML.
- El navegador interpretó el **<script>** como código legítimo.
- Esto confirma la presencia de una vulnerabilidad de tipo **XSS reflejado**.

---

## Captura de pantalla (simulada):

Se visualiza una alerta en el navegador con el texto “XSS”, generada por el script inyectado.

---

## Explicación técnica

- El sistema refleja el valor ingresado por el usuario en la página sin procesarlo ni codificarlo.
- Esto permite la **ejecución de scripts arbitrarios** por parte de cualquier atacante que manipule la URL o el formulario.
- En un entorno real, este tipo de ataque podría ser usado para **robar cookies**, **redirigir usuarios a sitios falsos**, o **registrar pulsaciones de teclado**.

---

## Recomendaciones básicas de mitigación

1. **Codificar caracteres especiales en las salidas** (**<**, **>**, **"**, **'**, **&**)

2. **Validar y limpiar entradas del usuario** tanto del lado cliente como del servidor
  3. Aplicar **Content Security Policy (CSP)** para restringir ejecución de scripts externos
  4. Utilizar frameworks con **renderizado seguro por defecto**
- 

## **Reflexión final**

Este ejercicio demuestra que una falla sencilla como reflejar texto sin sanitizar puede convertirse en una **puerta abierta para ataques serios**. Aunque esta prueba mostró solo una alerta, un atacante real podría usar este vector para comprometer sesiones, robar credenciales o propagar malware.

**Prevenir XSS no es una opción, es una responsabilidad del desarrollador.**

---