

Ejercicio Práctico

 **Tema:** Prevención de vulnerabilidades comunes en formularios web

Objetivo del ejercicio:

Aprender a identificar código vulnerable en una aplicación web simple y aplicar medidas básicas de mitigación frente a **SQL Injection** y **Cross-Site Scripting (XSS)**.

Instrucciones:



1. Observa el siguiente código PHP que procesa datos desde un formulario web:

```
<?php
$usuario = $_POST['usuario'];
$comentario = $_POST['comentario'];

$consulta = "INSERT INTO comentarios (usuario, comentario) VALUES ('$usuario',
'$comentario')";
mysqli_query($conexion, $consulta);
?>
```

Parte 1: Análisis

a) ¿Qué vulnerabilidades puedes identificar en este código?

- No tiene diseño responsivo
 - Es vulnerable a SQL Injection 
 - Es vulnerable a XSS 
-

b) ¿Cuál de las siguientes entradas podría ser maliciosa?

- juanperez
 - `<script>alert('XSS')</script>` ✓
 - `' OR '1'='1` ✓
-

Parte 2: Solución propuesta

Reescribe el código anterior aplicando dos medidas de protección:

- Evitar la **inyección SQL** con consultas preparadas (`mysqli`)
 - Evitar **XSS** escapando los datos del usuario antes de mostrarlos
-

Ejemplo de solución esperada:

```
<?php
```

```
$usuario = htmlspecialchars($_POST['usuario'], ENT_QUOTES, 'UTF-8');
```

```
$comentario = htmlspecialchars($_POST['comentario'], ENT_QUOTES, 'UTF-8');
```

```
$stmt = $conexion->prepare("INSERT INTO comentarios (usuario, comentario) VALUES (?, ?)");
```

```
$stmt->bind_param("ss", $usuario, $comentario);
```

```
$stmt->execute();
```

```
?>
```

Reflexión final:

- ¿Por qué no debemos confiar en los datos que vienen del cliente?
 - ¿Cómo se puede mejorar aún más la seguridad del formulario?
-

✓ Criterios de evaluación:

- ✓ Identificó correctamente las vulnerabilidades

- ✓ Aplicó correctamente consultas preparadas
 - ✓ Usó `htmlspecialchars()` para mitigar XSS
-