Reporte de Vulnerabilidad

Archivo: ecommerce.php

Código Analizado:

```
<!doctype html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.2/css/all.min.css">


    <title>Hello, world!</title>
</head>

<body>

    <div class="container-fluid-row ">

        <div class="row px-4">
            <h1>Prueba Tecnica Con JavaScript</h1>
            <div class="col-4">
                <div class="card">
                    <div class="card-body">
                        <form id="formProducto">
                            <input type="hidden" id="id">
                            <div class="mb-3">
                                <label for="nombre" class="form-label">Nombre</label>
                                <input type="text" class="form-control" id="nombre"
name="nombre" required>
                            </div>
                            <div class="mb-3">
                                <label for="cantidad" class="form-label">Cantidad</label>
                                <input type="number" class="form-control" id="cantidad"
name="cantidad" required>
                            </div>
                            <div class="mb-3">
                                <label for="precio" class="form-label">Precio</label>
                                <input type="number" class="form-control" id="precio"
name="precio" step="0.01" required>
                            </div>
                            <button type="submit" class="btn btn-success">Guardar</button>
                        </form>
                    </div>
                </div>
            </div>
            <div class="col-8">
                <div class="card">
                    <div class="card-body">
                        <h2>Lista de Productos</h2>

                        <table id="productosTable" class="table table-success table-striped">
                            <thead>
                                <tr>
                                    <th>ID</th>
                                    <th>Nombre</th>
                                    <th>Cantidad</th>
                                    <th>Precio</th>
                                    <th>Acciones</th>
                                </tr>
                            </thead>
                            <tbody>
                                <!-- Los productos se cargarán aquí mediante JavaScript -->
                            </tbody>
                        </table>
```

```
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"
integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"
crossorigin="anonymous"></script>
    -->
    <script src="ecommerce.js"></script>
</body>

</html>
```

Análisis: ```html
Análisis de Vulnerabilidades y Calidad del Código

Vulnerabilidades de Seguridad

El código proporcionado, al ser solo la parte HTML y haciendo referencia a un archivo Javascript externo, no presenta vulnerabilidades inherentes directamente en este fragmento. Las vulnerabilidades más probables estarán en el archivo Javascript externo `ecommerce.js`, específicamente en el manejo de datos del formulario y la interacción con el servidor. Asumiendo que existe una lógica en `ecommerce.js` para interactuar con un backend, las siguientes vulnerabilidades podrían estar presentes:

- **Cross-Site Scripting (XSS):** Si los datos ingresados por el usuario en los campos del formulario (nombre, cantidad, precio) no se sanitizan correctamente antes de ser mostrados en la tabla o enviados a un servidor, podrían ser vulnerables a XSS. Por ejemplo, si el nombre del producto contiene código JavaScript malicioso, este podría ejecutarse en el navegador del usuario cuando se muestre el nombre del producto.
- **Inyección SQL (si interactúa con una base de datos):** Si el backend utiliza los datos del formulario directamente en consultas SQL sin la debida parametrización o escape, podría ser vulnerable a inyección SQL.
- **Cross-Site Request Forgery (CSRF):** Si el backend no implementa protección contra CSRF, un atacante podría falsificar solicitudes en nombre de un usuario autenticado para modificar datos o realizar acciones no deseadas.
- **Falta de validación en el lado del servidor:** Confiar únicamente en la validación del lado del cliente (HTML `required`, type=`number`, step=`0.01`) no es suficiente. Un atacante podría omitir la validación del cliente y enviar datos inválidos directamente al servidor.
- **Divulgación de información sensible:** Si el backend expone información sensible (por ejemplo, detalles de configuración, claves API) en la respuesta a las solicitudes, esto podría ser una vulnerabilidad.

Métricas de Calidad del Código

Analizando el fragmento HTML:

- **Legibilidad:** El HTML está bien formateado y es legible. El uso de clases de Bootstrap facilita la comprensión de la estructura y el diseño.
- **Acoplamiento:** El HTML está acoplado a la librería Bootstrap. Esto no es inherentemente malo, pero significa que un cambio en Bootstrap podría afectar al código HTML.

- **Complejidad:** El HTML en sí es simple y no presenta una complejidad significativa. La complejidad estará principalmente en el Javascript `ecommerce.js` que maneja la lógica.
- **Duplicación:** No hay duplicación significativa en el fragmento HTML proporcionado.

Solución Propuesta

Mitigación de Vulnerabilidades

- **Sanitización de entradas (XSS):** Implementar la sanitización de entradas en el lado del servidor para todos los datos recibidos del formulario. Utilizar bibliotecas específicas para la sanitización de HTML para prevenir XSS. En el lado del cliente, utilizar `textContent` en lugar de `innerHTML` al mostrar los datos.
- **Consultas parametrizadas (Inyección SQL):** Utilizar consultas parametrizadas o ORM (Object-Relational Mapping) para interactuar con la base de datos. Esto evita que los datos del usuario sean interpretados como código SQL.
- **Tokens CSRF (CSRF):** Implementar tokens CSRF para proteger las solicitudes contra ataques CSRF. Generar un token único por sesión y verificarlo en cada solicitud.
- **Validación en el servidor:** Realizar validación exhaustiva en el lado del servidor para todos los datos recibidos del formulario. Verificar tipos de datos, rangos de valores y formatos.
- **Gestión adecuada de errores:** Evitar exponer información sensible en los mensajes de error. Registrar los errores en un archivo de registro en lugar de mostrarlos directamente al usuario.
- **HTTPS:** Asegurar que toda la comunicación entre el cliente y el servidor se realiza a través de HTTPS para proteger los datos en tránsito.

Mejora de la Calidad del Código

- **Abstracción de la lógica de presentación:** Considerar el uso de un framework JavaScript como React, Angular o Vue.js para separar la lógica de la interfaz de usuario del manejo de datos. Esto mejoraría la mantenibilidad y testabilidad del código.
- **Modularización del código JavaScript:** Dividir el archivo `ecommerce.js` en módulos más pequeños y cohesivos para mejorar la legibilidad y reducir la complejidad.
- **Documentación:** Agregar comentarios claros y concisos al código JavaScript para explicar la lógica y el propósito de cada función y variable.
- **Testing:** Implementar pruebas unitarias y de integración para verificar el correcto funcionamiento del código JavaScript y prevenir regresiones.

```
```