Reporte de Vulnerabilidad

Archivo: producto.php

Código Analizado:

```
<?php
include "../../models/conexion.php"; // Asegúrate de que la ruta a este archivo es correcta

$id = $_GET["id"];
// Realizamos la consulta a la base de datos
$sql = $conexion->query("SELECT * FROM productos where $id");

$producto = $sql->fetch_assoc();


// Devolvemos los datos en formato JSON
echo json_encode($producto);
?>
```

Análisis: ```html
Vulnerabilidades de Seguridad

Inyección SQL

Descripción: El código es vulnerable a inyección SQL debido a que la variable `$id` proveniente de `$_GET["id"]` se inserta directamente en la consulta SQL sin validación ni sanitización. Un atacante puede manipular el valor de `id` en la URL para ejecutar código SQL malicioso, comprometiendo la base de datos.

Línea Aproximada:

```
                    $sql = $conexion->query("SELECT * FROM productos where $id");
```

Impacto: Un atacante podría obtener acceso no autorizado a la base de datos, modificar o eliminar datos, o incluso ejecutar comandos en el servidor de la base de datos.

Mitigación:

- **Usar consultas preparadas (prepared statements):** Las consultas preparadas permiten separar la definición de la consulta SQL de los datos que se insertan en ella. Esto evita que los datos sean interpretados como parte de la consulta SQL.
- **Validar y sanitizar la entrada:** Si el `id` debe ser un número entero, asegúrate de que lo sea usando `intval()` o `filter_var()`. Si debe ser una cadena, escapa los caracteres especiales con `mysqli_real_escape_string()` (pero las consultas preparadas son preferibles).

Métricas de Calidad del Código

Legibilidad

El código es relativamente corto y fácil de entender en su funcionalidad básica. Sin embargo, la falta de comentarios sobre el propósito del código y las validaciones podría afectar la legibilidad para otros desarrolladores.

Mejora: Añadir comentarios descriptivos. Nombrar las variables de manera más descriptiva.

Acoplamiento

El código está acoplado al archivo `conexion.php` a través de la inclusión `include "../../models/conexion.php";`. Si la forma de conexión a la base de datos cambia en `conexion.php`, este script también necesitará ser modificado.

Mejora: Considerar inyección de dependencias, aunque para este ejemplo simple podría ser excesivo.

Solución Propuesta

El siguiente código utiliza una consulta preparada para prevenir la inyección SQL y mejora la legibilidad con comentarios. También se asume que el `id` es un entero. Si no lo es, adapta la validación.

```
                <?php
                include "../../models/conexion.php";

                // Obtener el ID del parámetro GET y validarlo como entero
                $id = isset($_GET["id"]) ? intval($_GET["id"]) : 0; //Valor predeterminado
seguro

                // Verificar que el ID sea válido (ej: mayor que 0)
                if ($id = 0) {
                    echo json_encode(array("error" => "ID inválido"));
                    exit;
                }

                // Preparar la consulta SQL para evitar inyección SQL
                $sql = $conexion->prepare("SELECT * FROM productos WHERE id = ?");
                $sql->bind_param("i", $id); // "i" indica que $id es un entero

                // Ejecutar la consulta
                $sql->execute();

                // Obtener el resultado
                $result = $sql->get_result();
                $producto = $result->fetch_assoc();

                // Cerrar la consulta
                $sql->close();

                // Devolver los datos en formato JSON
                echo json_encode($producto);
                ?>
```

Explicación de los cambios:

- Se utiliza `intval()` para asegurar que `$id` sea un entero, previniendo algunos tipos de inyección SQL básica.
- Se usa una consulta preparada (`$conexion->prepare()`) para insertar el `$id` de manera segura, evitando que sea interpretado como código SQL.
- Se usa `bind_param()` para enlazar el parámetro `$id` a la consulta preparada, especificando su tipo como entero ("i").
- Se agrega validación básica para asegurar que el ID es un valor positivo, evitando consultas innecesarias o potencialmente dañinas. Se usa `isset()` para verificar si la variable `$_GET["id"]` existe.

```
```