Reporte de Vulnerabilidad

Archivo: productos.php

Código Analizado:

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

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

$productos = array();

// Recorremos el resultado de la consulta y lo agregamos al arreglo $productos
while ($producto = $sql->fetch_assoc()) {
    $productos[] = $producto;
}

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

Análisis: ```html
Vulnerabilidades de Seguridad

Inyección SQL (Potencial)

Aunque el código proporcionado no contiene directamente inyección SQL, el archivo incluido `conexion.php` es el punto débil. Si `conexion.php` no escapa correctamente los datos, o utiliza directamente datos provistos por el usuario para construir la query, entonces este código es vulnerable.

Línea: N/A (Posible vulnerabilidad en `conexion.php`)

Mitigación:

- Utilizar sentencias preparadas (prepared statements) con parámetros en el archivo `conexion.php` para escapar correctamente los datos antes de ser insertados en la consulta SQL.
- No incluir archivos de conexión directamente, sino llamar a funciones que retornen la conexión previamente configurada.
- Asegurarse de que la clase de conexión (si la hubiera) escape los datos al insertar.
- Implementar validación y saneamiento exhaustivos de cualquier entrada de usuario que se utilice para construir la consulta SQL. Sin embargo, esto no elimina la necesidad de sentencias preparadas.

Revelación de Información Sensible

Si el archivo `conexion.php` contiene credenciales de la base de datos (usuario, contraseña, nombre de la base de datos), la inclusión de este archivo implica que, en caso de acceso no autorizado, se estaría revelando información sensible.

Línea: N/A (Información sensible contenida en `conexion.php`)

Mitigación:

- Asegurar permisos de acceso adecuados al archivo `conexion.php` para evitar lectura no autorizada.
- Utilizar variables de entorno para almacenar las credenciales de la base de datos en lugar de incluirlas directamente en el código.
- Limitar los privilegios de la cuenta de base de datos al mínimo necesario para la operación.

Métricas de Calidad del Código

Complejidad Ciclomática

Baja. El código es sencillo y directo, sin ramificaciones complejas.

Duplicación

Baja. El fragmento de código proporcionado no muestra duplicación. Sin embargo, la duplicación podría existir en el archivo `conexion.php` o en otros archivos que lo utilicen.

Legibilidad

Buena. El código es fácil de entender y seguir. Los nombres de las variables son descriptivos.

Acoplamiento

Moderado. El código depende del archivo `conexion.php`. Un alto acoplamiento dificulta la reutilización y el testeo del código.

Mejora: Se podría desacoplar la lógica de acceso a la base de datos mediante la creación de una clase o interfaz para la gestión de la conexión. Esto permitiría intercambiar fácilmente la implementación de la base de datos sin afectar el resto del código. Además, inyectar la dependencia de la conexión, en lugar de incluir el archivo directamente, mejoraría la testabilidad.

Manejo de Errores

El código carece de manejo de errores. Si la consulta falla, no se realiza ninguna acción para registrar el error o notificar al usuario.

Mejora: Agregar manejo de errores (try-catch) y logueo de errores es esencial para la robustez de la aplicación. Considerar también retornar códigos de error HTTP apropiados en caso de fallas, en lugar de simplemente no hacer nada.

Solución Propuesta

La solución principal se centra en asegurar que el archivo `conexion.php` utilice sentencias preparadas para evitar la inyección SQL. Además, se recomienda externalizar la configuración de la base de datos utilizando variables de entorno y limitar los privilegios de la cuenta de base de datos. El manejo de errores debería ser implementado para mejorar la robustez y la calidad del código. Finalmente, una inyección de dependencia de la conexion en lugar de la inclusion directa aumenta la testabilidad y reduce el acoplamiento.

Ejemplo (`conexion.php`):

```
                    <?php

                    $servername = getenv('DB_HOST');
                    $username = getenv('DB_USER');
                    $password = getenv('DB_PASSWORD');
                    $dbname = getenv('DB_NAME');

                    try {
                        $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
                        // set the PDO error mode to exception
                        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
                        // Emulate prepared statements
                        $conn->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);

                        // Función para ejecutar consultas preparadas
                        function ejecutarConsulta($conn, $sql, $params = []) {
```

```
                            $stmt = $conn->prepare($sql);
                            $stmt->execute($params);
                            return $stmt;
                        }
                    } catch(PDOException $e) {
                        echo "Connection failed: " . $e->getMessage();
                        die(); // O manejar el error de conexión de manera más elegante
                    }

                    ?>
```

Código Principal (ejemplo):

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

                    try {
                         // Realizamos la consulta a la base de datos usando la función
ejecutarConsulta
                         $sql = "SELECT * FROM productos";
                         $stmt = ejecutarConsulta($conn, $sql);

                         $productos = $stmt->fetchAll(PDO::FETCH_ASSOC);


                         // Devolvemos los datos en formato JSON
                         echo json_encode($productos);
                    } catch (Exception $e) {
                        // Manejo de errores (ejemplo: loguear el error)
                        error_log("Error al obtener productos: " . $e->getMessage());
                        http_response_code(500); // Internal Server Error
                        echo json_encode(["error" => "Error al obtener productos."]);
                    }
                    ?>
```

```
```