Reporte de Vulnerabilidad

Archivo: actualizar.php

Código Analizado:

```
<?php
include "../../models/conexion.php"; // Asegúrate de que la ruta a este archivo es correcta
$data = json_decode(file_get_contents('php://input'), true);

$id = $data['id'];
$nombre = $data['nombre'];
$cantidad = $data['cantidad'];
$precio = $data['precio'];

$sql = "UPDATE productos SET nombre='$nombre', cantidad='$cantidad', precio='$precio' WHERE
id=$id";

if ($conexion->query($sql) == true) {
    echo json_encode(["message" => "Producto actualizado exitosamente"]);
} else {
    echo json_encode(["message" => "Error al actualizado producto"]);
}

?>
```

Análisis: ```html
Vulnerabilidades de Seguridad

Inyección SQL

Descripción: El código es vulnerable a inyección SQL debido a que los datos recibidos del usuario (`$nombre`, `$cantidad`, `$precio`, `$id`) son directamente concatenados en la consulta SQL sin ningún tipo de sanitización o escapado. Esto permite a un atacante inyectar código SQL malicioso y manipular la base de datos.

Línea Afectada (Aproximada):

```
$sql = "UPDATE productos SET nombre='$nombre', cantidad='$cantidad', precio='$precio' WHERE
id=$id";
```

Impacto: Un atacante podría modificar, borrar o robar información de la base de datos. Podría incluso ejecutar comandos en el servidor subyacente, dependiendo de los permisos de la cuenta de la base de datos.

Cómo Mitigar:

- **Usar Sentencias Preparadas (Prepared Statements):** Las sentencias preparadas permiten separar la consulta SQL de los datos, previniendo la inyección SQL. PDO o MySQLi son las opciones recomendadas.
- **Escapado de Datos:** Si bien no es la solución ideal, usar funciones de escapado específicas de la base de datos (como `mysqli_real_escape_string`) puede ayudar a mitigar la vulnerabilidad. Sin embargo, es menos seguro que las sentencias preparadas.
- **Validación de Datos:** Validar que los datos de entrada coincidan con el tipo esperado y el rango de valores permitidos es esencial (e.g., `$cantidad` y `$precio` deben ser numéricos).
- **Principio de Privilegio Mínimo:** Asegurarse de que la cuenta de la base de datos usada por la aplicación tenga solo los permisos necesarios.

Métricas de Calidad del Código

Complejidad

Análisis: El código es relativamente simple en términos de complejidad ciclomática. Sin embargo, la falta de validación de entradas y el manejo de errores más robusto podrían aumentar la complejidad.

Duplicación

Análisis: No hay duplicación evidente en este pequeño fragmento, pero en un contexto más amplio, la lógica de acceso a la base de datos podría repetirse, sugiriendo la necesidad de una capa de abstracción.

Legibilidad

Análisis: El código es relativamente legible, aunque el uso de sentencias preparadas mejoraría la legibilidad al separar la consulta SQL de los datos.

Acoplamiento

Análisis: El código está fuertemente acoplado a la base de datos. Una capa de abstracción de datos reduciría el acoplamiento y facilitaría las pruebas y el mantenimiento. La dependencia directa de `conexion.php` también incrementa el acoplamiento. Inyectar la conexión, en lugar de incluirla directamente, sería mejor.

Solución Propuesta

El siguiente fragmento de código usa sentencias preparadas para prevenir la inyección SQL y mejora la validación de datos.

```
<?php
include "../../models/conexion.php"; // Asegúrate de que la ruta a este archivo es correcta
$data = json_decode(file_get_contents('php://input'), true);

$id = filter_var($data['id'], FILTER_VALIDATE_INT);
$nombre = trim($data['nombre']);
$cantidad = filter_var($data['cantidad'], FILTER_VALIDATE_INT);
$precio = filter_var($data['precio'], FILTER_VALIDATE_FLOAT);

if ($id === false || $cantidad === false || $precio === false) {
    echo json_encode(["message" => "Error: Datos inválidos."]);
    exit;
}


$sql = "UPDATE productos SET nombre=?, cantidad=?, precio=? WHERE id=?";
$stmt = $conexion->prepare($sql);

if ($stmt === false) {
    echo json_encode(["message" => "Error preparing statement: " . $conexion->error]);
    exit;
}

$stmt->bind_param("sidi", $nombre, $cantidad, $precio, $id); //sidi: string, integer, double,
integer

if ($stmt->execute()) {
    echo json_encode(["message" => "Producto actualizado exitosamente"]);
} else {
    echo json_encode(["message" => "Error al actualizar producto: " . $stmt->error]);
}

$stmt->close();
$conexion->close();
?>
```

Mejoras:

- **Sentencias Preparadas:** Usa sentencias preparadas con `mysqli` para prevenir la inyección SQL.
- **Validación de Datos:** Valida los tipos de datos y usa `filter_var` para asegurar que los valores de entrada son del tipo esperado.

- **Manejo de Errores:** Incluye un manejo de errores más robusto para detectar problemas con la preparación y ejecución de la consulta.
- **Cierre de Conexión y Statement:** Cierra la conexión y la declaración preparada para liberar recursos.
- **Sanitización Basica** Usa `trim` para remover espacios innecesarios en el `nombre`.

```
```