6. REVISIÓN DE “OJS-3.5.0-1”

6.1. Vulnerabilidades críticas

6.1.1. IDX-019: Inyección SQL

INYECCIÓN SQL					
ID	IDX-019	SEVERIDAD	CRITICA	CVSS V3.1	9.9
VECTOR CVSS	AV:NETWORK/AC: LOW/PR: LOW/UI: NONE/S: CHANGED/C: HIGH/I: HIGH/A: HIGH				
TIPO DE VULNERABILIDAD	A03 2021 - Inyección				

Descripción

Se invocan consultas SQL construidas con datos de entrada que provienen de una fuente no fiable. Lo cual podría permitir a un atacante modificar el significado de la sentencia o ejecutar comandos SQL arbitrarios.

Estas llamadas se realizan desde múltiples ficheros en la aplicación, y en la mayoría de ellos no se realizan filtrados de estos datos.

Los errores de inyección SQL se producen cuando:

1. Los datos entran en un programa desde una fuente no fiable.
2. Los datos se utilizan para construir dinámicamente una consulta SQL.

En un ataque de inyección SQL, el usuario puede enviar una consulta SQL directamente a la base de datos, obteniendo acceso sin proporcionar las credenciales apropiadas. Los atacantes pueden entonces ver, exportar, modificar y borrar información confidencial; cambiar contraseñas y otra información de autenticación; y posiblemente obtener acceso a otros sistemas dentro de la red. Esta es una de las categorías de vulnerabilidad más comúnmente explotadas, pero puede evitarse en gran medida mediante buenas prácticas de codificación.

Evidencias

```
);
    $connection->prepare("set search_path to {$searchPath}")->execute();
}
}
```

Se encontraron evidencias en las siguientes rutas:

- lib/pkp/lib/vendor/laravel/framework/src/Illuminate/Database/Connectors/MySqlConnection.php (Líneas 29, 97 y 121)
- lib/pkp/lib/vendor/laravel/framework/src/Illuminate/Database/Connectors/PostgresConnector.php (Líneas 125, 141, 159 y 184)

- lib/pkp/lib/vendor/laravel/framework/src/Illuminate/Database/Connectors/SqlServerConnector.php (Línea 54)

Recomendaciones

El mejor remedio, es sanitizar los parámetros en estas clases, en vez de ir por todo el código realizando las validaciones.

De forma general, la causa principal de una vulnerabilidad de inyección SQL es la capacidad de un atacante de cambiar el contexto en la consulta SQL, haciendo que un valor que el programador pretendía que se interpretara como datos se interprete en su lugar como un comando. Cuando se construye una consulta SQL, el programador sabe qué debe interpretarse como parte del comando y qué debe interpretarse como datos. Las sentencias SQL parametrizadas pueden imponer este comportamiento al no permitir cambios de contexto dirigidos a datos y evitar casi todos los ataques de inyección SQL. Las sentencias SQL parametrizadas se construyen utilizando cadenas de SQL normales, pero cuando es necesario incluir datos proporcionados por el usuario, incluyen parámetros de enlace, que son marcadores de posición para los datos que se insertan posteriormente. En otras palabras, los parámetros de enlace permiten al programador especificar explícitamente a la base de datos lo que debe tratarse como un comando y lo que debe tratarse como datos. Cuando el programa está listo para ejecutar una sentencia, especifica a la base de datos los valores en tiempo de ejecución a utilizar para cada uno de los parámetros *bind* sin el riesgo de que los datos sean interpretados como una modificación del comando.

Cuando se conecta a MySQL se puede utilizar sentencias SQL parametrizadas (en lugar de concatenar cadenas suministradas por el usuario) de la siguiente manera:

```
...
$stmt = $mysqli->prepare($query);
$stmt->bind_param('ss',$nombre_usuario,$nombre_artículo);
$stmt->ejecutar();
...
```

La extensión MySQL Mejorada (mysqli) está disponible para los usuarios de MySQL en PHP5. El código que dependa de una base de datos diferente debería comprobar si existen extensiones similares.

Escenarios más complicados, a menudo encontrados en código de generación de informes, requieren que la entrada del usuario afecte a la estructura de la sentencia SQL, por ejemplo añadiendo una restricción dinámica en la cláusula **WHERE**. No utilice este requisito para justificar la concatenación de la entrada del usuario para crear una cadena de consulta. Evite los ataques de inyección SQL en los que la entrada del usuario deba afectar a la estructura de la instrucción con un nivel de indirección: cree un conjunto de cadenas legítimas que correspondan a los distintos elementos que podría incluir en una instrucción SQL. Cuando

construya una sentencia, utilice la entrada del usuario para seleccionar de este conjunto de valores controlados por la aplicación.

Referencias

https://owasp.org/Top10/es/A03_2021-Injection/
<https://cwe.mitre.org/data/definitions/89.html>