Inyección SQL

Nicolás A. Ortega Froysa

4 de octubre de 2021

1. Introducción

Una *inyección SQL* es una especie de ataque cibernético en que se inyecta código SQL malicioso en una entrada de datos de un formulario. De esta forma, el atacante puede correr peticiones arbitrarias en la base de datos, y así conseguir información de ella, insertar información errónea, corromper los datos, o incluso borrar información importante.

Es un ataque común en sistemas web que hacen uso de una base de datos, y especialmente cuando permiten interacción (indirecta) entre el usuario y la base de datos (e.g. formularios de entrada).

2. Peticiones SQL Erróneas

Cuando se escribe una petición SQL, se hace combinando en una misma sentencia los datos y los comandos. A menudo, cuando se crean aplicaciones que interactuan con una base de datos, las entradas de los formularios se insertan *verbatim* en la petición SQL. Cuando ocurre esto, un mal actor puede aprovecharse para inyectar un carácter de escape, y luego correr los comandos que quiera (i.e. ejecución de código arbitrario).

Por ejemplo, imaginemos un comando SQL que busque si existe un determinado usuario (asumimos que `PASSWD()` hace el *hash*):

```
SELECT * FROM Users WHERE
    username="nortega" AND password=PASSWD("MyPasswd");
```

Esto, normalmente sólo escogería la entrada del usuario *nortega*, y sus datos. Mas, si uno quisiera conseguir todos los datos de todos los usuarios, podría insertar en el formulario de usuario: `nortega" OR 1=1; --`. Al hacer esto, el comando terminaría siendo:

```
SELECT * FROM Users WHERE
    username="nortega" OR 1=1;
    -- AND password=PASSWD("MyPasswd");
```

Por lo tanto, la contraseña quedaría ignorada (como comentario), y como la condición `1=1` siempre es verdad, terminaría por devolver todas las entradas de la tabla `Users`.

Para evitar esto, se pueden user ciertas funciones en lenguajes como PHP (e.g. `mysqli_escape_string()`) para *escapar* aquellos caracteres que puedan usarse para hacer este tipo de inyecciones (e.g. añadiendo un `\` antes del carácter para escapar el *string*). Entonces el comando pasaría a ser lo siguiente:

```
SELECT * FROM Users WHERE
    username="nortega\" OR 1=1; --"
    AND password=PASSWD("MyPasswd");
```

3. Inyección SQL Ciega

Similarmente al caso anterior, la inyección ciega ocurre cuando la página o aplicación no muestra directamente la información de la base de datos, pero sí se mostraría de manera diferente cuando se cumple una condición. Por ejemplo, con el mismo código del caso anterior, si suponemos que el código se ejecuta en un formulario de *login*, si el número de entradas es igual a 1, entonces acepta entrada en la cuenta. En cuyo caso, con añadir la condición `LIMIT 1`, ya entraría en la primera cuenta que encuentra (que normalmente es la cuenta del administrador).

```
SELECT * FROM Users WHERE
    username="nortega" OR 1=1 LIMIT 1;
    -- AND password=PASSWD("MyPasswd");
```

Esto se evita de la misma manera que en la sección anterior.

4. Inyección SQL De Segundo Orden

Supongamos que todos los formularios vienen con salvaguardas para evitar las inyecciones SQL. Aún así, es posible implementar un ataque más sutil que no afecte inmediatamente la base de datos, sino que lo afecte luego. En estos casos, la SQL del formulario se ejecuta correctamente, guardando el código SQL inyectado apropiadamente como un dato, para que luego sea usado

y ejecutado erróneamente como código SQL. De este modo, podríamos decir que los casos anteriores eran inyecciones de primer nivel, ya que se ejecutaban inmediatamente del formulario, mientras que las inyecciones de segundo nivel son aquellos que se ejecutarán más tarde.