SQL injection attack

SQL injection is a web security vulnerability that may allow an attacker to interfere with the queries that an application makes to its database. It generally allows to view data that in the normal circumstances cannot be retrieved. As a result, an attacker may be able to modify or delete data that can cause persistent changes to the application's content or behavior. Additionally, harm can be done to underlying server and backend infrastructure or denial of service attack may occur.

SQL injection usually occurs when a user is asked for an input and instead of proper input, the user enters an SQL statement that is unknowingly run on the system database.

For example, let us consider a simple SQL injection attack scenario. Suppose, for log-in purpose, a user has to enter his/her User ID. Hence, the SQL statement might be as below.

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

Now, let us assume in the field for User ID, the user enters `105 OR 1=1`

As a result, the statement will be as below that is always true and SQL injection attack can occur.

```
SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

Because of flaws in the input validation implementation, SQL injection occurs and it can have consequences like authentication bypass, gaining access to unauthorized data, unauthorized data manipulation, achieving administrative privileges etc.

There are some ways to prevent SQL injection attack. Few of them are mentioned below.

- **Input validation**: The validation process is aimed at verifying whether or not the type of input submitted by a user is allowed. Input validation makes sure it is the accepted type, length, format, and so on. Only the value which passes the validation can be processed. It helps counteract any commands inserted in the input string. Validation shouldn't only be applied to fields that allow users to type in input. Regular expressions should be used as whitelists for structured data to ensure strong input validation. In case of a fixed set of values, which value is returned, should be checked.
- **Avoiding administrative privileges**: Application should not be connected to the database using an account with root access. This should be done only if absolutely needed since the attackers could gain access to the whole system. Even the non-administrative accounts server could place risk on an application, even more so if the database server is used by multiple applications and databases. It is better to enforce least privilege on the database to defend the application against SQL injection. It

must be ensured that each application has its own database credentials and that those credentials have the minimum rights the application needs. Instead of trying to determine which access rights should be taken away, focus should be on identifying what access rights or elevated permissions the application needs. If a user only needs access to some parts, a mode could be created that strictly serves this function.

- **Character escaping**: Character-escaping functions should be used for user-supplied input provided by each database management system. This is done to make sure the DBMS never confuses it with the SQL statement provided by the developer.
- **Web application firewall**: To identify SQL injection attacks, a web application firewall can be used. It operates in front of the web servers and monitors the traffic which goes in and out of the web servers and identifies patterns that constitute a threat.