SQL injection is a web app security vulnerability that allows an attacker to tamper with the SQL query that an application makes in its database. Through this attack, an attacker can get hold of the data that they cannot normally see or get access to. It might include data that belongs to other users or the data that an application is itself able to access. With the help of this attack, an attacker can

* delete or modify data in the database
* escalate an attack to compromise the server and other underlying backend infrastructure
* perform a DoS attack

Retrieving a data from the database: An application when prompted about data, sends a HTTP request to the (URL) to the server and the server issues a SQL query to the database which represents the same data but in different forms- one in the form of HTTP Request URL and the other in the form of a SQL query.

URL: <https://insecure-website.com/products?category=Gifts>

Is equivalent to

SQL: SELECT \* FROM products WHERE category=’Gifts’ and released=1

Restriction ‘released’ is used to specify products that that released which means that released=1 denotes the products that are released. Released keyword denotes the data that can be accessed or are authorized to access.

Attack query:

URL: <https://insecure-website.com/products?category=Gifts> ‘--

SQL: SELECT \* FROM products WHERE category=’Gifts’ --' and released=1

Thus, the part after the “ --' ” is considered as comment since in SQL -- means comment out and dropped in the SQL query.

URL: <https://insecure-website.com/products?category=Gifts> ‘ +OR+1=1 --

SQL: SELECT \* FROM products WHERE category=’Gifts’ OR 1=1 --' this query returns items that are under the category of wither gifts or where 1=1 which means all items in a category.

SQL query is vulnerable to ‘--' which the attacker takes full advantage of with login username “administrator’--’

Compromised query to retrieve data from other tables of the database using ‘union’:

SELECT name, description FROM products WHERE category=’Gifts’ UNION SELECT username, password FROM users --’

Using these query attackers can retrieve the table of users containing their usernames and passwords. Using SQL injection, it is also possible to know about the database as a whole and all the tables it contains. Using the SELECT \* FROM information\_schema.tables it is possible to do so.

Blind SQL injection vulnerabilities are those sql queries in which the application does not include any query results in the responses. As such, it is less vulnerable to regular sql injection and is more complex to execute. Attackers can use multiple techniques to execute blind SQL injection vulnerabilities by:

* Changing the logic of the query to trigger detectable difference in the app’s response depending on the truth of a single condition
* Conditionally triggering a time delay in the processing of the query
* Triggering out of band network interaction

SQL injection vulnerabilities can be detected using systematic tests for checking all the entry points in the application.

SQL injections are made in where clause of UPDATE statement, inserted values of the INSERT statement, within the table and column names of the SELECT statement and in select statements within the ORDER BY clause.

First Order SQL injections are those in which the application processes a user input in a SQL query from a HTTP request in an unsafe way such that the input is altered by the attacker making the query reveal unauthorized data. On the other hand, second order SQL injections also known as stored SQL injection occur when the application saves user input in the database from an HTTP request and no vulnerability arises at that moment but later when incorporated with another SQL query it reveals data in an unsafe way.

SQL UNION attacks:

SELECT a, b FROM table1 UNION SELECT c, d FROM table2

For SQL Union attack to work, the following conditions must be met:

* Between the individual queries, the columns of the queries should be of the same data type
* The individual queries must return the same number of columns