SQL INJECTION

- SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database.
- It generally allows an attacker to view data that they are not normally able to retrieve.
- This might include data belonging to other users, or any other data that the application itself is able to access.
- In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

- The SQL Injection is a code penetration technique that might cause loss to our database.
- It is one of the most practiced web hacking techniques to place malicious code in SQL statements, via webpage input.
- SQL injection can be used to manipulate the application's web server by malicious users.

- SQL injection is a type of security exploit in which an attacker adds malicious SQL code to a web form input box to gain access to resources or make changes to data.
- It is one of the most common web application security vulnerabilities.
- SQL injection attacks can be prevented by using parameterized queries, which ensure that user input is treated as data rather than as code.
- It’s important to note that SQL injection attacks can have serious consequences and can result in data loss, data corruption, or even complete system compromise.

What is the impact of a successful SQL injection attack?

- A successful SQL injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information.
- Many high-profile data breaches in recent years have been the result of SQL injection attacks.
- In some cases, an attacker can obtain a persistent backdoor into an organization's systems, leading to a long-term compromise that can go unnoticed for an extended period.

SQL injection examples

There are a wide variety of SQL injection vulnerabilities, attacks, and techniques, which arise in different situations. Some common SQL injection examples include:

- ✓ Retrieving hidden data, where you can modify a SQL query to return additional results.
- ✓ Subverting application logic, where you can change a query to interfere with the application's logic.
- UNION attacks, where you can retrieve data from different database tables.
- ✓ Examining the database, where you can extract information about the version and structure of the database.
- ✓ Blind SQL injection, where the results of a query you control are not returned in the application's responses.

Retrieving hidden data

Consider a shopping application that displays products in different categories. When the user clicks on the Gifts category, their browser requests the URL:

```
https://insecure-website.com/products?category=Gifts
```

This causes the application to make a SQL query to retrieve details of the relevant products from the database:

```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

This SQL query asks the database to return:

- all details (*)
- from the products table
- where the category is Gifts
- and released is 1.

The restriction `released = 1` is being used to hide products that are not released. For unreleased products, presumably `released = 0`.

The application doesn't implement any defenses against SQL injection attacks, so an attacker can construct an attack like:

```
https://insecure-website.com/products?category=Gifts'--
```

This results in the SQL query:

```
SELECT * FROM products WHERE category = 'Gifts'--' AND released = 1
```

The key thing here is that the double-dash sequence `--` is a comment indicator in SQL, and means that the rest of the query is interpreted as a comment. This effectively removes the remainder of the query, so it no longer includes `AND released = 1`. This means that all products are displayed, including unreleased products.

Retrieving data from other database tables

In cases where the results of a SQL query are returned within the application's responses, an attacker can leverage a SQL injection vulnerability to retrieve data from other tables within the database. This is done using the `UNION` keyword, which lets you execute an additional `SELECT` query and append the results to the original query.

For example, if an application executes the following query containing the user input "Gifts":

```
SELECT name, description FROM products WHERE category = 'Gifts'
```

then an attacker can submit the input:

```
' UNION SELECT username, password FROM users--
```

This will cause the application to return all usernames and passwords along with the names and descriptions of products.