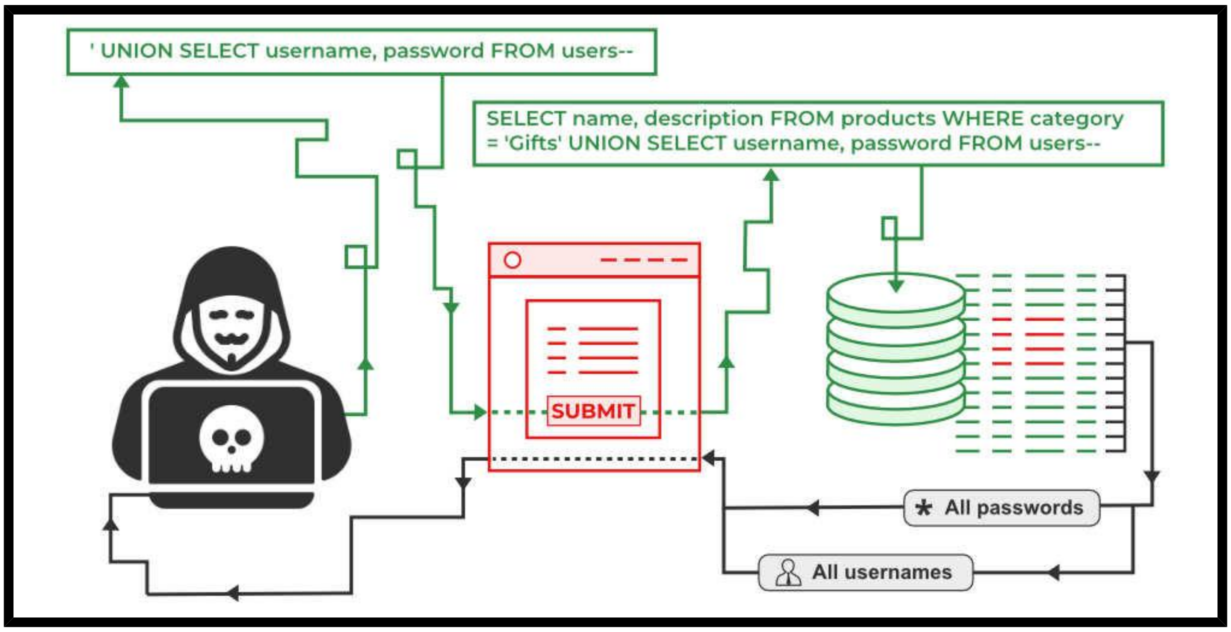
Injection

SQL injection is a code injection technique attackers use to gain unauthorized access to a database by injecting malicious SQL commands into web page inputs.

Attackers can extract sensitive information, modify database data, execute administration operations on the database (such as shutdown DBMS), recover the content of a given file present on the DBMS file system, and in some cases, issue commands to the operating system.

In this article, we will discuss what is SQLi(SQL Injection), Types of SQL injection, SQL injection in web pages, how to prevent SQL injection attacks, and many more.


' UNION SELECT username, password FROM users--
SELECT name, description FROM products WHERE category = 'Gifts' UNION SELECT username, password FROM users--
SUBMIT
All passwords
All usernames

What is SQL Injection?

SQLi or **SQL Injection** is a web page vulnerability that lets an attacker make queries with the database. Attackers take advantage of web application vulnerability and inject an SQL command via the input from users to the application.

Attackers can SQL queries like SELECT to retrieve confidential information which otherwise wouldn't be visible. SQL injection also lets the attacker to perform a **denial-of-service (DoS) attacks** by overloading the server requests.

The Exploitation of SQL Injection in Web Applications

Web servers communicate with database servers anytime they need to retrieve or store user data. SQL statements by the attacker are designed so that they can be executed while the web server is fetching content from the application server.

SQL in Web Pages

SQL injection typically occurs when you ask a user for input, such as their username/user ID, and instead of their name/ID, the user inputs an SQL statement that will be executed without the knowledge about your database.

For example,

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

The above code is constructing an SQL query by directly concatenating a user input (txtUserId) into the query string. Attackers can easily exploit this by giving an input that is always true, like x=x,1=1, etc.

If the attacker gave input as " 105 OR 1=1 " in the UserId field, the resulting SQL will be:

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

This resulting query will return data of all users, not just the user with UserId ="105″.

SQL Injection Example

For a better understanding of how attackers do a SQL injection attack, let's learn **how to do an SQL injection attack** ourselves. In this example, we will perform a basic SQL injection attack and learn the process behind it.

Suppose we have an application based on student records. Any student can view only his or her records by entering a unique and private student ID.

Suppose we have a field like the one below:

Student id: The student enters the following in the input field: **12222345 or 1=1**.

Query

```
SELECT *
FROM STUDENT
WHERE STUDENT-ID == 12222345
OR 1 = 1;
```

SQL Injection based on 1=1 is always true. As you can see in the above example, **1=1** will return all records for which this holds true. So basically, all the student data is compromised. Now the malicious user can also similarly use other SQL queries.

Consider the following SQL query.

Query - 1

```
SELECT *
FROM USER
WHERE USERNAME = ''
AND PASSWORD = '';
```

Now the malicious attacker can use the '=' operator cleverly to retrieve private and secure user information. So following query when executed retrieves protected data, not intended to be shown to users.

Query - 2

```
SELECT *
FROM User
WHERE
(Username = ''
OR 1 = 1)
AND
(Password = ''
OR 1 = 1);
```

Since **1=1** always holds true, user data is compromised.

SQL Injection Types

There are different types of SQL injection attacks:

1. In-band SQL Injection

It involves sending malicious SQL queries directly through the web application's interface and allows attackers to extract sensitive information or modify the database itself.

2. Error-based SQL Injection

Attackers exploit error messages generated by the web application by analyzing error messages to gain access to confidential data or modify the database.

3. Blind SQL Injection

Attackers send malicious SQL queries and observe the application's response. By analyzing the application's behavior, attackers can determine the success of the query.

4. Out-of-band SQL Injection

Uses a different channel to communicate with the database. Allows attackers to exfiltrate sensitive data from the database.

5. Inference-based SQL Injection

Uses statistical inference to gain access to confidential data. Attackers create queries that return the same result regardless of input values.

Impact of SQL Injection

The hacker can retrieve all the user data present in the database such as user details, credit card information, and social security numbers, and can also gain access to protected areas like the administrator portal. It is also possible to delete user data from the tables.

Nowadays, all online shopping applications and bank transactions use back-end database servers. So in case the hacker is able to exploit SQL injection, the entire server is compromised.

SQL Injection Prevention

Developers can use the following prevention measures to prevent SQL injection attacks.

- **User Authentication** − Validating input from the user by pre-defining the length, type of input, and authenticating the user.
- **Restricting access privileges** − Defining how much data any outsider can access from the database. Users should not be granted permission to access everything in the database.
- **Do not use system administrator accounts** − Avoid using system administrator accounts for routine tasks to minimize security risks.

SQL Injection Based on Batched SQL Statements

- **Most databases support batch SQL statements** − Many database systems allow the execution of multiple SQL statements in a single batch.
- **A batch of SQL statements** − It is a collection of two or more SQL statements separated using semicolons.

The SQL declaration underneath will return all rows from the "users" desk after which delete the "Employees " table.

Query

```
SELECT * FROM Users;
DROP TABLE Employees;
```

Look at the following example:

Syntax

```
txtEmpId = getRequestString("EmpId");
txtSQL = "SELECT * FROM Users WHERE EmpId = " + txtEmpId;
```

The valid SQL statement would look like this:

Query

```
SELECT * FROM Users WHERE EmpId = 116;
DROP TABLE Employees;
```

SQL Injection – FAQs

What is SQL injection?

SQL injection is a technique used to extract user data by injecting web page inputs as statements through SQL commands. It allows malicious users to manipulate a web application's web server by injecting malicious code into SQL statements via web page inputs.

How common is SQL injection as a hacking technique?

SQL injection is one of the most common web hacking techniques, posing a significant threat to web applications and databases.

How does SQL injection exploit web applications?

Web servers communicate with database servers to retrieve or store user data. Attackers craft SQL statements that can execute while the web server fetches content from the application server, compromising the security of the web application.

Can you provide an example of SQL injection?

Certainly. Let's say there's an application for student records. An attacker enters "12222345 or 1=1" into a student ID field, which modifies the SQL query to retrieve all student records. This compromises all student data. Similar manipulations can delete records or gain unauthorized access to data.

What are the impacts of SQL injection?

The impact can be severe. Attackers can retrieve user data such as details, credit card information, and social security numbers, as well as access protected areas like administrator portals. It's also possible to delete user data. If successful, the entire server can be compromised.

How can SQL injection be prevented?

To prevent SQL injection, you should:

- Use user authentication to validate input and define input field characteristics.
- Restrict user access privileges to limit database access.
- Avoid using system administrator accounts.
- For more details, refer to the "How to Protect Against SQL Injection Attacks?" article.

When does SQL injection typically occur in web applications?

SQL injection typically occurs when a user provides input like a username or user ID, and the application executes it as an SQL statement without proper validation or knowledge of the database structure.