What is SQL Injection (SQLi)

SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.

An SQL Injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL Server, or others. Criminals may use it to gain unauthorized access to your sensitive data: customer information, personal data, trade secrets, intellectual property, and more. SQL Injection attacks are one of the oldest, most prevalent, and most dangerous web application vulnerabilities. The OWASP organization (Open Web Application Security Project) lists injections in their OWASP Top 10 2017 document as the number one threat to web application security.

How and Why Is an SQL Injection Attack Performed

To make an SQL Injection attack, an attacker must first find vulnerable user inputs within the web page or web application. A web page or web application that has an SQL Injection vulnerability uses such user input directly in an SQL query. The attacker can create input content. Such content is often called a malicious payload and is the key part of the attack. After the attacker sends this content, malicious SQL commands are executed in the database.

SQL is a query language that was designed to manage data stored in relational databases. You can use it to access, modify, and delete data. Many web applications and websites store all the data in SQL databases. In some cases, you can also use SQL commands to run operating system commands. Therefore, a successful SQL Injection attack can have very serious consequences.

- Attackers can use SQL Injections to find the credentials of other users in the database. They can then impersonate these users. The impersonated user may be a database administrator with all database privileges.
- SQL lets you select and output data from the database. An SQL Injection vulnerability could allow the attacker to gain complete access to all data in a database server.
- SQL also lets you alter data in a database and add new data. For example, in a financial application, an attacker could use SQL Injection to alter balances, void transactions, or transfer money to their account.

- You can use SQL to delete records from a database, even drop tables. Even if the administrator makes database backups, deletion of data could affect application availability until the database is restored. Also, backups may not cover the most recent data.
- In some database servers, you can access the operating system using the database server. This may be intentional or accidental. In such case, an attacker could use an SQL Injection as the initial vector and then attack the internal network behind a firewall.

There are several types of SQL Injection attacks: in-band SQLi (using database errors or UNION commands), blind SQLi, and out-of-band SQLi. You can read more about them in the following articles: Types of SQL Injection (SQLi), Blind SQL Injection: What is it.

To follow step-by-step how an SQL Injection attack is performed and what serious consequences it may have, see: Exploiting SQL Injection: a Hands-on Example.

Simple SQL Injection Example

The first example is very simple. It shows, how an attacker can use an SQL Injection vulnerability to go around application security and authenticate as the administrator.

The following script is pseudocode executed on a web server. It is a simple example of authenticating with a username and a password. The example database has a table named `users` with the following columns: `username` and `password`.

```
# Define POST variables
uname = request.POST['username']
passwd = request.POST['password']


# SQL query vulnerable to SQLi
sql = "SELECT id FROM users WHERE username='" + uname + "' AND password='" + passwd + "'"


# Execute the SQL statement
database.execute(sql)
```

These input fields are vulnerable to SQL Injection. An attacker could use SQL commands in the input in a way that would alter the SQL statement executed by the

database server. For example, they could use a trick involving a single quote and set the `passwd` field to:

```
password' OR 1=1
```

As a result, the database server runs the following SQL query:

```
SELECT id FROM users WHERE username='username' AND password='password' OR 1=1'
```

Because of the `OR 1=1` statement, the `WHERE` clause returns the first `id` from the `users` table no matter what the `username` and `password` are. The first user `id` in a database is very often the administrator. In this way, the attacker not only bypasses authentication but also gains administrator privileges. They can also comment out the rest of the SQL statement to control the execution of the SQL query further:

```
-- MySQL, MSSQL, Oracle, PostgreSQL, SQLite
' OR '1'='1' --
' OR '1'='1' /*
-- MySQL
' OR '1'='1' #
-- Access (using null characters)
' OR '1'='1' %00
' OR '1'='1' %16
```

Reference:

https://www.acunetix.com/websitesecurity/sql-injection/

https://portswigger.net/web-security/sql-injection