A Case Study on

SQL Injections

Alok Bhawankar
Final Year B.Tech CSE
1032170126

What is SQL Injection Attack?

SQL Injection attacks (or SQLi) alter SQL queries, injecting malicious code by exploiting application vulnerabilities.

Successful SQLi attacks allow attackers to modify database information, access sensitive data, execute admin tasks on the database, and recover files from the system. In some cases attackers can issue commands to the underlying database operating system.

This article focuses on the anatomy of SQL injection attacks. To learn more, including how to prevent attacks, see our complete guide to SQL injection.

Real-Life SQL Injection Attack Examples

Over the past 20 years, many SQL injection attacks have targeted large websites, business and social media platforms. Some of these attacks led to serious data breaches. A few notable examples are listed below.

Breaches Enabled by SQL Injection

- **GhostShell attack**—hackers from APT group Team GhostShell targeted 53 universities using SQL injection, stole and published 36,000 personal records belonging to students, faculty, and staff.
- **Turkish government**—another APT group, RedHack collective, used SQL injection to breach the Turkish government website and erase debt to government agencies.
- **7-Eleven breach**—a team of attackers used SQL injection to penetrate corporate systems at several companies, primarily the 7-Eleven retail chain, stealing 130 million credit card numbers.
- **HBGary breach**—hackers related to the Anonymous activist group used SQL Injection to take down the IT security company's website. The attack was a response to HBGary CEO publicizing that he had names of Anonymous organization members.

Notable SQL Injection Vulnerabilities

- **Tesla vulnerability**—in 2014, security researchers publicized that they were able to breach the website of Tesla using SQL injection, gain administrative privileges and steal user data.
- **Cisco vulnerability**—in 2018, a SQL injection vulnerability was found in Cisco Prime License Manager. The vulnerability allowed attackers to gain shell access to systems on which the license manager was deployed. Cisco has patched the vulnerability.
- **Fortnite vulnerability**—Fortnite is an online game with over 350 million users. In 2019, a SQL injection vulnerability was discovered which could let attackers access user accounts. The vulnerability was patched.

How Does a SQL Injection Attack Work?

There are several types of SQL injection—here are a few common variants.

SQL injection based on user input

A basic SQL injection attack uses user inputs. Web applications accept inputs through forms, which pass a user's input to the database for processing. If the web application accepts these inputs without sanitizing them, an attacker can inject SQL statements via form fields and delete, copy, or modify the contents of the database.

SQL injection based on cookies

Another approach to SQL injection is modifying cookies to "poison" database queries. Web applications often load cookies and use their data as part of database operations. A malicious user, or malware deployed on a user's device, could modify cookies, to inject SQL in an unexpected way into the backend database.

SQL injection based on HTTP headers

Server variables such HTTP headers can also be used for SQL injection. If a web application accepts inputs from HTTP headers, fake headers containing arbitrary SQL can inject code into the database.

Second-order SQL injection

These are possibly the most complex SQL injection attacks, because they may lie dormant for a long period of time. A second-order SQL injection attack delivers poisoned data, which might be considered benign in one context, but is malicious in another context. Even if developers sanitize all application inputs, they could still be vulnerable to this type of attack.

Impact of SQL Injection Attacks

Here are a few examples of the harm SQL injection attacks can cause to an organization, if successful:

- **Steal credentials**—SQL injections can be used to find user credentials. Attackers can then impersonate these users and use their privileges.
- **Access databases**—attackers can use SQL injections to gain access to the information stored in a database server.
- **Alter data**—attackers can use SQL injections to alter or add new data to the accessed database.
- **Delete data**—attackers can use SQL injections to delete database records, including drop tables.
- **Access networks**—attackers can use SQL injections to access database servers with operating system privileges. The attacker can then attempt to access the network.

SQL Injection Code Examples

Example 1: Injecting Malicious Statement into Form Field

This is a simple SQL injection attack based on user input. The attacker uses a form that requires first name and last name as inputs. The attacker inputs:

- First name: `malicious'ex`
- Last name: `Smith`

The SQL statement that processes the form inputs looks like this:

```
select id, firstname, lastname from authors
```

Once the attacker injects a malicious expression into the first name, the statement looks like this:

```
select id, firstname, lastname from authors where firstname = 'malicious'ex' and lastname ='newman'
```

The database identifies incorrect syntax due to the single apostrophe, and tries to execute the malicious statement.

Here is code from OWASP showing how to prevent this attack, by sanitizing the input.

```
String firstname = req.getParameter("firstname");

String lastname = req.getParameter("lastname");

String query = "SELECT id, firstname, lastname FROM authors WHERE firstname = ? and lastname = ?";
```

```
// Using a PreparedStatement to take the user’s query and sanitize
it

// by setting it as a string, instead of directly passing it to DB

PreparedStatement pstmt = connection.prepareStatement( query );

pstmt.setString( 1, firstname );

pstmt.setString( 2, lastname );

try

{

    ResultSet results = pstmt.execute( );
```

Conclusion

What makes SQL injection attacks even more menacing for businesses is the fact that these seemingly simple attacks can cost them not only hefty sums of money but may also lead to a lack of customer trust in the long run. And the fact that so many websites and web apps still face this vulnerability regularly reflects the seriousness of the issue. The next big question which must come into the minds of web app owners is how to prevent SQL injection? As a relief, the solutions to SQL injection are simple, yet immensely useful. You just have to follow some good security practices from the beginning itself. It might take longer or seem a bit tedious, but surely the rewards will be satisfying.

Plagiarism Report