SQL Injection

Simple to Defend Against Yet Nefarious and Ubiquitous

By:

Dovid Budman

Dovid Samuels

The Threat

Most websites and apps use databases

The SQL Injection attacks vulnerabilities in databases

Attacking Confidentiality, Integrity and Accessibility

Introducing SQLI

SQL Injection (SQLI) involves:

Typing SQL code into submission text boxes

in order to:

Manipulate the data in the database

Short Example

Website has login screen for user to enter username and password

The code behind the website then passes the values to a query to its database

db.rawQuery('''SELECT ID FROM users WHERE username="$username" AND password="$password"''');

What's the problem?

What is the vulnerability here?

The Injection

A malicious user can enter into the username sql code that will be passed on to the query!

For example, if the attacker enters **' OR '1'='1'; --**

The user can then see all users’ info since 1 always equals 1,
and without entering a password since -- comments out the rest of the query.

Exploring Vulnerabilities

Test every input element (form fields, URL parameters, cookies, etc.)

Simplest injection method is to submit a single apostrophe

- Helps reveal errors in input handling
- May provide correct syntax needed
- Can help you construct more effective SQL injection query

What is vulnerable?

SQL Server, relational databases, MySQL, and IBM DB2 are vulnerable to SQL injection attacks

SQL injection attacks target websites or apps without secure coding, rather than vulnerable software

Good to know which sql server

Slight differences among the SQL servers regarding syntax, so good to first try to discover which one is being used so that can tailor the attack to it

For example, trying to see if is mysql, enter normal input (hammer) and “and 1 = sleep(2);-- yielding:

Select * from products where pname like ‘%hammer’ and
1 = sleep(2);--%;

If sleeps know is mysql. If not, then not and can try wait for delay to see if SQL Server and so on. This is a type of Blind SQLI.

Stacking Commands

Some APIs allow for multiple commands, leading to ability to add any command wish.

For example,

```
'; DROP TABLE users;--
';exec master..xp_cmdshell "net localgroup administrators /add hacker";--
```

Using Error Messages

A developer might be overly eager to make clear error messages, leaving the database vulnerable to attack

An attacker can use error messages shown by the database server to get information about the structure of the database such as names and types

In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database

UNION

A Union operation combines into one command showing the results of two or more SELECT statements which can merge data from tables not intended for public consumption.

Only works when queries have same number and type of columns and know the actual column names (which can be guessed or discovered through injection)

Union Example

Many websites, especially where e-commerce is involved, use tables.

Can test if table information is outputted to the screen by guessing how many columns and using **union 1,2,3 from dual**, where guess is using 3 columns and dual is a dummy table used for testing purposes. Now, if see 1 2 3 outputted we know that there are 3 columns and very importantly that table information is being outputted to the screen which makes it very vulnerable to attack.

Union example continued

If know that using mysql for example, there is a meta-data table called information_schema and has a table called information_schema.tables

Append: UNION(SELECT TABLE_NAME, TABLE_SCHEMA, 3 FROM information_schema.tables);--

Then can see if there is a users table for example. Now want to know column names in the users table, so append:

Union (Select column_name, 2 , 3 fromm information_schema.columns where table_name = ‘users’);--

Union Example Continued

Now, can add 3 columns that interested in from users table. Say found column names of u_username, u_pass, u_type. Append:

```
UNION (Select u_username, u_pass, u_type  from users);--
```

With this can find usernames and their password, which are probably hashed, and their type - allowing us to possibly find admins, in which case would probably want to work specifically on his password.

Blind SQL Injection

Learning about the database based on true or false results as opposed to actual data

Either getting error message or not, or based on the time taken to return a result using wait type commands

Blind SQLI

When App is vulnerable to SQLI but results aren’t visible

May display differently based on the results of logical statement injected into the legitimate SQL statement for the page

Was time-intensive, but recent automation has helped with this difficulty

Blind SQLI Example - Content Based

See if the following return different results: (where 23 alone worked already by submitting a simple request saw id in url)

Enter for id: 23 and (or) 1=1, and separately, 23 and 1=2 yielding:

Select * from products where id=23 and (or) 1=1

Select * from products where id=23 and 1=2

If different results have learned is open to SQLI - can see true or false answers to attempted queries

Blind SQLI - Time-Based

As used in previous example to find out which sql server the web app is using, by seeing if the sleep syntax worked or not.

Another use - to check if table with a guessed name exists by selecting for the table and sleeping so if sleeps know that the name is correct.

Second-Order Injections

Even if the web app is defending against SQL Injections as input, a dangerous query can sneak in and be able to cause damage later, where not set up to defend against it.

For example enter as username a dangerous query, so that even if treated as string, later when the attacker changes password for example, the query can run then.

Automation

There are programs to automate the process of searching out vulnerabilities across different websites and what kind of vulnerability is open.

Example software: BSQL Hacker, SQLmap, SQLninja - given a website explore potential vulnerabilities

Automation to perform the actual attack

There is also automation to perform the actual attack. For example a python script to attack websites in bulk.

Some more SQL Injection Tools: Metasploit, BSQLHacker, Marathon Tool, SQL Power Injector, Havij, SQL Brute, Sqlninja, sqlget, Absinthe, Blind Sql Injection Brute Forcer, sqlmap, Darkjumper, Pangolin, SQLPAT, Fj-Injector Framework, safe3si, SQLler, Sqlsus, SQLEXEC() Function, Automagic SQL Injector, SQL Inject-Me, NTO SQL Invader, The Mole, Sql Poizon;
And for mobile: DroidSQLi, sqlmapchik

Some Advantages of SQLI

SQL Injections are relatively simple to perform which leads to more attacks.

This combined with how common databases and text submission boxes are leads to a high number of attacks.

And even though the attacks are relatively simple, the results can be devastating.

What is SQL Injections' Story?

The SQL injection vulnerability has been known for 22 years, but it continues to be a major issue to this day.

It was used in an estimated two-thirds of web app attacks in 2017.

Recent attacks include the 2017 hack on more than 60 universities and governments worldwide.

What is SQL Injections’ Story?

The vulnerability was first documented by “rain.forest.puppy”, a well respected security expert whose real name is Jeff Forristal, in the December 1998 issue of Phrack magazine which described

a Microsoft SQL server yielding possibly sensitive data through the use of commands in normal user inputs like “name” or “phone number”

Still, SQL injection didn’t hit the mainstream until 2002.

Some Real-World Examples

- Guess.com (2002)
- Microsoft SQL Server (2007,2009)
- 10’s of thousands of computers (2008)
- Kaspersky’s Malaysian site (2008)

Some Real-World Examples

- Automated search for SQL Injection Vulnerabilities (2008)
- Novel use as vehicle to deliver malware (2008)
- 130 Million Credit Card numbers (2009), reportedly the biggest case of identity theft in American history
- Mysql.com (2011)

Yahoo and China attacked

In 2012 Yahoo was attacked using Union-Based SQLI

In 2013 Chinese Government databases were attacked by hacktivist group “RaptorSwag” and published publicly with Anonymous

Massive Confidentiality Loss

In 2014 a security company revealed that almost 420,000 websites suffered loss of confidentiality information by SQLI.

A Seven Year String of Attacks

One string of high-profile attacks took place over a seven-year period affecting popular organizations like:

**7-Eleven, JCPenney Inc., NASDAQ,
Dow Jones Inc., and JetBlue Airways**

Different Underlying Vulnerabilities

Tautology-Based Attack (‘ OR 1=1)

Reconnaissance (Descriptive Errors)

Union-Query Attack (UNION SELECT …)

Piggybacking (code in input fields)

Stored Procedures

Inference (Blind attacks)

Alternate Encoding (masking commands)

The Two Most common SQL Injection Attacks

The most common attacks can be grouped into two major groups:

Tautology-Based and Inference Attacks

Defense

Response to the threat began around 2005

AMNESIA

SQL-IDS

Three Pronged Approach

Client Side and Server Side Protection

In 2015 new technique:

Use application layer detection at both the client and server

Countering the Threat

Defending against SQLI can be relatively simple.

Treat input data as untrusted or dangerous

Detect if multiple attempts being done, OR or UNION being present

Countermeasures

Parameterized statements

Countermeasures - Sanitization

Sanitize Inputs

For example, in PHP it is usual to escape parameters using the function `mysqli_real_escape_string();` before sending the SQL query:

```
$mysqli = new mysqli('hostname', 'db_username', 'db_password', 'db_name');

$query = sprintf("SELECT * FROM `Users` WHERE UserName='%s' AND Password='%s'",

                  $mysqli->real_escape_string($username),

                  $mysqli->real_escape_string($password));

 $mysqli->query($query);
```

Countermeasures - Database Permissions

```
For example,on Microsoft SQL Server:
```

```
    deny select on sys.tables to webdatabaselogon;
```

Countermeasures

Pattern Matching

Whitelisting

Blacklisting

Prepared Statements

Stored Procedures

Don’t Make it Easy for an Attacker

Use unusual table names and column names

Intrusion Prevention System

Don’t just code with in mind what should do, but also what *could do*

Code Reviews and Testing

Don’t use too informative error messages

Don’t Make it Easy for an Attacker

Use strong hashes for passwords

Enforce only returning one user on login

Limit number of queries allowed by individual users within certain time period to prevent search algorithm execution

Some SQLI Detection Tools

dotDefender, IBM Security AppScan, WebCruise, Snort Rule, HP WebInspect, SQLDict, SQLiX, SQL Block Monitor, Acunetix Web Vulnerability Scanner, GreenSQL Database Security, Microsoft Code Analysis Tool.NET (CAT.NET), NGS SQirreL Vulnerability Scanners, WSSA – Web Site Security Scanning Service, N-Staler Web Application Security Scanner

It Just Makes Sense

The time, money and effort involved in using the techniques to seal off vulnerabilities is well worth it considering how easy, common and damaging the attacks are

Resources

[Wikipedia](https://en.wikipedia.org/wiki/SQL_injection) https://en.wikipedia.org/wiki/SQL_injection
[W3Schools](https://www.w3schools.com/sql/sql_injection.asp) https://www.w3schools.com/sql/sql_injection.asp
https://owasp.org/www-community/attacks/SQL_Injection
https://owasp.org/www-community/attacks/SQL_Injection
[Offence Cheat Sheet](https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/)
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
[Defence Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html)
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Blind SQLi
https://owasp.org/www-community/attacks/Blind_SQL_Injection#:~:text=Description,based%20on%20the%20applications%20response.&text=This%20makes%20exploiting%20the%20SQL,difficult%2C%20but%20not%20impossible.%20.
History
https://commons.erau.edu/cgi/viewcontent.cgi?article=1475&context=jdfsl#:~:text=Even%20though%20it%20was%20first,information%20security%20community%20until%202002.
SQFLITE https://www.youtube.com/watch?v=xrCBVwYjcvQ
SQLITE Examples
https://www.exploit-db.com/docs/english/41397-injecting-sqlite-database-based-applications.pdf