SQL Injection in Web Applications: Vulnerabilities
Innovative Attacks and Remedies

Authors:
Author 1: Manish Gupta
Email: gupta.manish@fiserv.com
Mobile: +91-8527135580

Fiserv India Pvt. Ltd.
DLF IT Park, 6th, 7th and 8th Floor, Tower C&D, A-44/45, Sec-62, Noida, U.P – 201301
ABSTRACT .......................................................................................................................... 3

SQL INJECTION OVERVIEW .......................................................................................... 3

SQL INJECTION VULNERABILITY VERSUS SQL INJECTION ATTACK................................. 3
KEY CONCEPTS OF A SQL INJECTION ATTACK................................................................... 3
WHAT A HACKER CAN DO WITH SQL INJECTION ATTACK?.................................................. 4
HOW CAN SQL INJECTION HAPPEN? ................................................................................... 4

SYSTEM DESIGN ................................................................................................................. 5

SQL INJECTION VULNERABILITY ASSESSMENT FLOW CHART ....................................... 6

TYPES OF SQL INJECTION ATTACKS ................................................................................ 7

1) TAUTOLOGY ......................................................................................................................... 7
   String SQL Injection ............................................................................................................ 7
   Numeric SQL Injection ....................................................................................................... 7
   Comments Attack .............................................................................................................. 7

2) INFERENCE ........................................................................................................................ 8
   Blind SQL Injection ........................................................................................................... 8
   Timing Attacks .................................................................................................................. 8
   Database Backdoors ......................................................................................................... 8
   Command SQL Injection .................................................................................................. 8

PREVENTING SQL INJECTION ............................................................................................ 9

1) PARAMETERIZED STATEMENTS....................................................................................... 9
2) PATTERN CHECK ............................................................................................................... 9
3) DATABASE PERMISSIONS ............................................................................................... 9
4) SANITIZE THE INPUT ..................................................................................................... 9
5) USE STORED PROCEDURES FOR DATABASE ACCESS ...................................................... 9

CONCLUSION ...................................................................................................................... 10

BIBLIOGRAPHY ..................................................................................................................... 10

NAME: .................................................................................................................................... 10
EXPERIENCE: ....................................................................................................................... 10
CURRENT COMPANY: ........................................................................................................... 10
CERTIFICATIONS: .................................................................................................................. 10
SUMMARY: ........................................................................................................................... 10
REFERENCES .......................................................................................................................... 11
ABSTRACT
In today’s world, web application plays an important role in one’s life. As these applications are used by numbers of people, the hackers can find the way to enter into the application by using malicious attack.

Many web Applications accepts user inputs and use them to form SQL statement at runtime. This practice would allow attacker to use malicious SQL query segment as user input which can result in a different request to database. By using SQL injection attacks, an attacker can obtain or modify confidential and sensitive information.

There should be some techniques that checks for the unauthorized access and without proof of his/her credentials user should not be allowed to enter into the application. In spite of the development of different approaches to mitigate SQL injection, it still remains an alarming threat to Web Application.

This paper aim is to collate some of the existing knowledge, introduce new techniques and demonstrate how to get complete control over the database management systems through SQL injection vulnerability.

SQL INJECTION OVERVIEW
SQL Injection refer to when the user supplied data to the web application in the form of SQL statement and access important data such as social security number, credit card number and other bank details from database. Using SQL Injection, attacker can create, read, update, alter or delete any information stored in the Database.

SQL INJECTION VULNERABILITY VERSUS SQL INJECTION ATTACK
Vulnerability is defined as any defect or flaw existing in the application that can be used by any unauthorized user to access unlimited data.

An SQL Injection, a user adds malicious keywords into a SQL query to enter the application and access all the information stored in database.

KEY CONCEPTS OF A SQL INJECTION ATTACK
* SQL Interpreter is used to interpret the SQL query sent by user
* Attackers adds malicious data to the SQL interpreter in such a way that interpreter fails to differentiate between the actual commands and data crafted by user.
What a hacker can do with SQL Injection attack?

* ByPassing Logins
* Accessing secret data
* Modifying contents of website
* Shutting down the MySQL server

**HOW CAN SQL INJECTION HAPPEN?**

The above example shows the online store that can access by using the valid username and password. An SQL Injection attack occurs when the user causes the system to generate SQL queries differently from what the user interface programmer projected.

For instance, an attacker may attempt to gain root privileges by manipulating the user name or password string.

Consider the following code:
SQLQuery="Select * from accounts where login='" + request.getParameter("login") + '" 
and password='" + request.getParameter("passwd") + '";

In the above code, the web application retrieves the username and password that entered by user in the given fields and concatenates these input values into the SQL query.

This helps the user to generate a query for the purpose of user authentication.

However, If a user has provided value ‘username’ in the login field and value ‘password’ ‘OR ‘1=1’ into the password field, then the query will be constructed as:

Select * from accounts where name='username' and password='password' OR ‘1=1’

Since under all circumstances one will always be equal to one, and hence the query will return all the data from the database. In this way, an unauthorized user will be able to view sensitive information.

**SYSTEM DESIGN**

The above figure shows SQLProb has four main components:

1. The Query Collector executes SQL queries during the data collection phase
2. The User Input Extractor identify user input data based on a global pairwise alignment algorithm
3. The Parse Tree Generator generates the parse tree for the incoming queries
4. The User Input Validator evaluates user input whether it is normal or malicious

Based on user input validation algorithms.

The shaded area shows the proxy. SQLProb uses two phases:

a) Data Collection Phase
During the data collection phase, user inputs validator stores queries that cover all the functionalities of the application in a repository.

b) Query Evaluation Phase.

During the query validation phase, the proxy captures all application generated query and forward it to user input extractor and the parse tree generator simultaneously.

The user input extractor extracts the user input data based on a global alignment algorithm. The user input validator validates the extracted user inputs in the parse tree, which is generated by the parse tree generator. If the user inputs are validated to be normal, the generated query will be sent to database directly; otherwise, the query will be discarded as a malicious query.

**SQL INJECTION VULNERABILITY ASSESSMENT FLOW CHART**
TYPES OF SQL INJECTION ATTACKS

There are a number of categorized SQL injection types that can be executed with a web-browser. They are:

1) Tautology

SQL injection codes are injected into one or more conditional statements so that they are always evaluated to be true. Under this technique, we may have the following types and scenarios of attacks:

**String SQL Injection**

This type of injection is also referred to as AND/OR Attack. The attacker injects the SQL tokens or strings to a conditional query statement that always evaluates to a true statement. This type of injection targeted all the rows in the database table to be returned. This injection includes the following: (a) Bypassing authentication, (b) Identifying parameters that can be injected, and (c) Extraction of data

**Example:**

Injected Statement: SELECT * FROM users WHERE name= 'Lucia01' OR '1'='1'

**Numeric SQL Injection**

Numeric values are used instead of strings. The attacker injects numeric values to a conditional query statement that would always evaluate to a true statement.

**Example:**

Injected Statement: SELECT * FROM users WHERE name= '101' OR '1' = '1'

**Comments Attack**

The malicious code and comments can be inserted after the "—" in the Where Clause. Anything written after the comment will be ignored. Comments Attack can be combined with either String or Numeric SQL Injection so that it performs as a tautology which always evaluates to a true statement.

**Example:**
Generated SQL Query: SELECT username, password FROM clients WHERE username = 'user1 OR '1'='1 — AND password = 'whatever'.

In the above case, the password condition will be ignored as it is written after the comment and hence only the condition username will be checked.

2) Inference
An attacker derives logical conclusions by asking a series of True/False questions through SQL statements. Through a successful inference, attacker changes the behavior of the database.

Blind SQL Injection
Blind injection refers to the Web Application vulnerability to an SQL injection but the results of the injection are not visible to the attacker. For example, an unsuccessful injection navigates attacker to the main page whereas a successful injection redirects to a blank page.

Timing Attacks
An attacker gathered the information by observing the response time or the behavior of the database. This would help the attacker to decide wisely on the appropriate injection approach.

Database Backdoors
Databases are not only used to store data but also to keep malicious activity like a trigger. An attacker can set a trigger in order to get the user input and get it directed to his or her e-mail, for example.

Example:

101; CREATE TRIGGER myBackDoor BEFORE

INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='hacker@me.com'WHERE userid = NEW.userid.

Command SQL Injection
This injection is used to inject and execute commands specified by the attacker in the vulnerable application. The application executes the unwanted system commands. An OS command injection attack occurs when an attacker attempts to execute system level commands through a vulnerable application. Applications are considered vulnerable to OS command injection attack if they utilize user input in a system level command.
PREVENTING SQL INJECTION

There are a number of ways to prevent MySQL injections

1) **Parameterized statements**

   Now for most of the development platforms, parameterized statement rather than embedding user input in the statement can be used. Always preferred strongly typed parameterized query APIs with placeholder substitution markers, even when calling stored procedures.

2) **Pattern check**

   Boolean, float, Integer parameters can be checked if their value is valid representation for the given type. Strings that must follow some strict pattern (alphanumeric, date, UUID, etc.) can be checked if they match this pattern.

3) **Database permissions**

   User should be given least permission to access database. Limiting the permission on the database would reduce the effectiveness of any SQL injection attacks exploit any bugs in the web Application.

4) **Sanitize the input**

   SQL injection can be prevented if you adopt an input validation technique in which user input is authenticated against a set of defined rules for type, length and syntax and also against business rules.

   It is important to sanitize user inputs to ensure that they do not contain any malicious codes within SQL query. For example: - an email address can contain only these characters:

   abcdefghijklmnopqrstuvwxyz
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
   0123456789
   @.-_.+

   Therefore early detection of malicious code can be done easily and throwing error message in case of any

5) **Use stored procedures for database access**

   Always use stored procedures to avoid SQL injection for performing any database access.
Single procedure can be used to update, insert, delete the data from the database. For example, the "add order" procedure might reject that order if the customer were over his credit limit.

Procedures are more useful for bigger application where transactions of data are huge. Procedures are more robust and easier to maintain.

CONCLUSION
This paper explained how to exploit a single vulnerability in a web application at its best to get complete control of the server that runs the database, not only the data stored in the database as usually intended: the SQL injection itself can be considered as a stepping stone to the actual target for this research, which is the complete control of its server: operating system access, file system access and use of the compromised database server as a foothold in the internal network.

Bibliography

Name: Manish Gupta

Experience: 4.6 months experience in IT industry

Current Company: Fiserv Pvt. Ltd

Certifications:
- HP QTP 11 certified in the year 2012
- ISTQB in the year 2012
- DOEACC ‘O’ Level in 2004

Summary:
- Designed Automation Testing framework architecture using VB Script in QTP
- Experience in testing a Financial Domain project of Thomson Reuters
- Experience in all phases of Software Development Life Cycle (Requirements Analysis, Design, Testing, Implementation and Support) to perform QA activities
- Experience in Functional Testing, Database Testing using SQL queries, Regression Testing, GUI testing
REFERENCES


[7] An Introduction to SQL Injection Attacks for Oracle Developers - This also includes recommended defenses.