# SQL Injection Attacks: Techniques and Protection Mechanisms

Nikita Patel
Department of Info. Tech.
Patel College of Science & Technology
Bhopal, India

Fahim Mohammed
Department of Computer Science
Research Scholar NIT
Bhopal, India

Santosh Soni
Department of Computer Science
Patel College of Science & Technology
Bhopal, India

**Abstract-- When an internet user interacts in web environment by surfing the Net, sending electronic mail messages and participating in online forums lot of data is generated which may have user's private information. If this information is captured by third party tools and techniques; it may cause a breach in end user privacy. In the Web environment, end user privacy is one of the most controversial legal issues. In this paper issues related to information leakage through SQL injection attacks are presented and protection mechanisms are also discussed.**

**Keywords: - Privacy, Security, Code Injection, SQL Injection, web application security, Malicious Code, Vulnerability.**

## I. INTRODUCTION

As the Internet is growing day by day, most of the people are not aware of security and privacy. Internet is a widespread information infrastructure; it is basically an insecure channel for exchanging information. Web security is the set of rules and measures taken against web security threats. Web privacy is the ability of hiding end user's information. Nowadays most of the applications have the vulnerability (weakness) that makes a threat possible. An attack may be possible due to poor design, configuration mistakes, or poor written code of the web application. A threat can be harmful for database, control of web application, and other components of web application, that are needed to be protected from all types of threat. All types of code injection or SQL injection are very dangerous for these components of the web application.

To build secure applications, security and privacy must be considered, and developer must be aware about it. The main goals of information security are Confidentiality, Integrity and availability. Confidentiality means the information available on a system should be safe from unauthorized people; Integrity means the information available in an organization should be complete and whole. It shouldn't be altered by any unauthorized person. Availability is as important as Confidentiality and Integrity. It means the information requested or required by the authorized users should always be available.

## II. CODE INJECTION ATTACKS

Code Injection is a term used when malicious code/script is injected into a program/web application from an outside source, for example input field which is provided by the web application to take input from the end-user. This attack makes use of lack of accurate input/output data validation. The injected malicious code executes as a part of the application. The consequences of a successful code injection attack may result in either damage to an asset, or an undesirable operation. Attack can be performed within software, web application etc in which the weakness is present. Weakness contribute to the introduction of vulnerabilities within that software or web applications, vulnerability can be used by the attacker to exploit the web applications to gain unintended access to data, denial of services, or perform incorrect operations. HTML Injection Attack, Cross Site Scripting Attack, SQL Injection Attack, Shell Attack, Content Spoofing, HTTP Response Splitting, HTTP Request Splitting and XML Poisoning Attack are some examples of the code injection attack.

SQL Injection: - SQL injection is an attack technique which can be used by the attacker to exploit the web application; as a result the attacker may gain unauthorized access to a database or to retrieve information directly from the database. Attacker can exploit SQL injection vulnerabilities remotely without any database or application authentication. SQL injection attacks are straightforward in nature – an attacker just passes malicious string as an input to an application for stealing confidential information.

### *I. SQL Injection using Dynamic Strings*

Most of the web based application takes input from the end user for constructing dynamic SQL statement [2].

```
Query = "SELECT * FROM student WHERE sname = 'studentname' ";
```

### *II. Example 1 - Dynamically built SQL command string*

Consider a web application that takes input from the students and displays the result of the student, with the logic of the above SQL query, the result of the above query is as follows

Suppose an attacker submits a student name that looks like the following:

```
Student Name: nikita' OR '1'='1
```

The SQL command string built from this input would be as follows:

```
SELECT * FROM student WHERE sname = 'nikita' OR '1'='1
```

This query will return all rows from the student's database, regardless of whether "nikita" is a real user name. This is due to the OR statement appended to the WHERE clause. The comparison '1'='1' will always return a "true" result, making the overall WHERE clause evaluate to true for all rows in the table. If this is used for authentication purposes, the attacker will often be logged in as the first or last user in the table.

## III. CATEGORIES OF SQL INJECTION ATTACKS

There are four main kinds of SQL Injection attacks

1. SQL Manipulation
2. Code Injection
3. Function Call Injection
4. Buffer Overflows

SQL manipulation usually involves modifying the SQL query through altering the WHERE clause. In this class of attack, amend the WHERE clause of the statement so the WHERE clause constantly results in TRUE [2]www.integrigy.com/.../Integrigy_Oracle_SQL_Injection_Attacks].

In the case of Code injection an attacker introduces new SQL statements into the input field instead of valid input. The classic code or statement appends a SQL Server command to make SQL statement vulnerable. Code injection only works when multiple SQL statements per database request are supported or keywords like AND, OR are supported by the database.

Function call injection is the addition of database functions or user defined functions into a vulnerable SQL queries. These function calls can be used to make internal calls or modify data in the database that can be harmful to the users [2].

SQL injection of buffer overflows is a subset of function call injection. In several commercial and open-source databases, vulnerabilities exist in a few database functions that may result in a buffer overflow.

## IV. SQL INJECTION METHODS

There are four types of SQL Injection attacks

### *A. SQL Manipulation*

The most common type of SQL Injection attack is SQL manipulation. The attacker attempts to modify the present SQL statement by adding elements to the WHERE clause. An example of SQL manipulation can be given by a simple search application. This application takes student roll number as input and displays its result. The web application may run the following query [4].

```
SELECT * FROM student WHERE rollnum = '<user_input>'
```

This query will return the result of the student, but if the attacker attempts to manipulate the SQL statement to execute as –

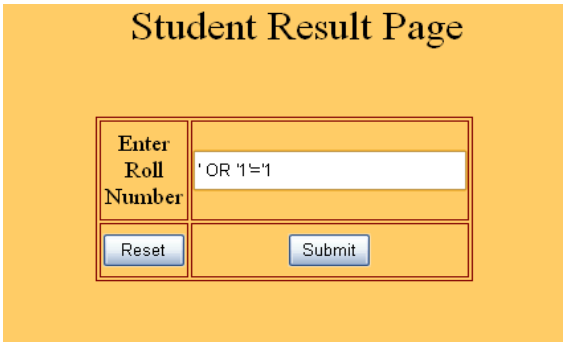


Figure 1 Taking Input from User

```
SELECT * FROM student WHERE rollnum = '' or '1' = '1'
```

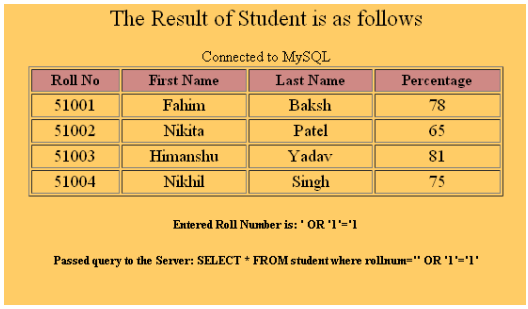


Figure 2 Result of the Malicious Code

The WHERE clause becomes true for every row and as a result it fetches all entries of the database, in this way the attacker gains access to the application.

## V. METHODS FOR PROTECTION AGAINST SQL INJECTION ATTACKS

SQL Injection attacks can be protected with simple changes in server site programming as well as client side programming. Developers must be aware of all types of attacks and take care for all possible attacks. Each and Every dynamic SQL statement must be sanitized. A single unprotected query can be harmful for the application, data, or database server.

### *B. Taking User Input From Predefined Choices*

In this way the web application can be secured from malicious attacks. The attacker cannot insert custom queries or any type of harmful script which can disturb the integrity of the database. This is a simple yet effective way to curb web application attacks. This can be established by making simple changes into the server site code.

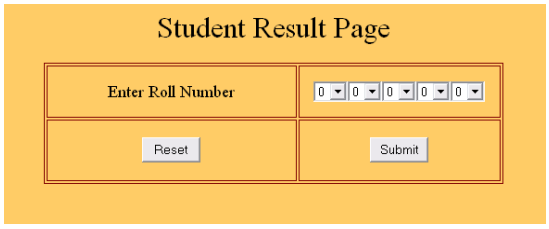


Figure 3 Taking Input From Predefined Choices

### C. *Bind Variables Mechanism*

Bind variable is another technique to control SQL injection attacks. Using bind variables helps in improving web application performance. The web application developer should use bind variables in all SQL statements. In java language there is a mechanism called prepared statement, this implements the concept of bind variables mechanism.

```
PreparedStatement pstate;

pstate=con.prepareStatement("select * from student where rollnum = ?");

pstate.setString(1, "sroll");
```

### D. *Parameterized Statements*

To defend SQL injection attacks, user input must not be directly passed in SQL queries. Instead, parameterized statements must be preferred, or else user input should be sanitized or filtered carefully [17].

To sanitize the given user input it must be assigned (bound) to a parameter and passed through a filtering or sanitizing function like one present in PHP (mysql_real_escape_string($user_input)). The use of this function adds a back slash (\) against all of the escape characters such that the malicious script present is not executed. The result of that function can be viewed in figure 4.

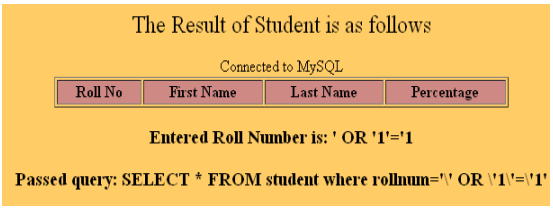


Figure 4 Result of Using Predefined Function

### E. *Input Validation*

This is the simplest method for defence against SQL injection attacks. Every passed string parameter ought to be validated. Many web applications use hidden fields and other techniques, which also must be validated. If a bind variable is not being used, special database characters must be removed or escaped.

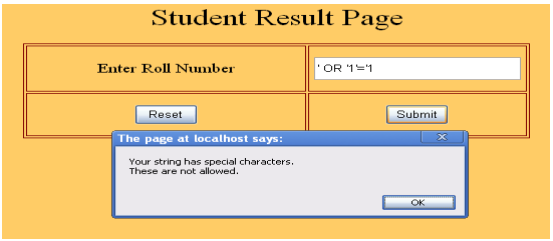


Figure 5 Input Validation Using Client Site Script

In most databases the single quote character and other special characters are a big issue, the simplest method to avoid them is to escape all single quotes. This can be established by using client side scripting language

## VI. CONCLUSION

Code injection attacks, especially SQL injection attack is one of the infamous issues. Controlling the malicious SQL code/script on the web application and maintaining the end privacy is still a key challenge for the web developer. These issues must be considered seriously by the web developers involved in developing websites using databases. This paper describes how an attacker can exploit the web application by using SQL injection attack to get confidential information from a database. Different protection mechanisms against SQL injection attack are also proposed.

ACKNOWLEDGMENT

The research presented in this paper would not have been possible without our college, PCST, Bhopal. We wish to express our appreciation to all the people who helped turn the World-Wide Web into the useful and popular distributed hypertext and providing information as it is anywhere. We also wish to thank the anonymous reviewers for their valuable suggestions, who helped in improving our paper content.

REFERENCES

[1] "An Introduction to SQL Injection Attacks for Oracle Developers," [Online]. Available: www.integrigy.com/Integrigy_Oracle_SQL_Injection_Attacks [Accessed: Oct.02, 2010].
[2] "SQL Injection" [Online] Available: http://projects.webappsec.org/w/page/13246963/SQL-Injection [Accessed: Oct 12,2010]
[3] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki [Accessed: Oct.5,2010].
[4] "SQL Injection Prevention," [Online] Available: http://www.owasp.org/index.php/SQL_Injection_Prevention [Accessed: Nov.10, 2010].
[5] "Prepared Statements" [Online] Available: http://www.owasp.org/index.php/SQL_Injection_Prevention [Accessed: Nov 12, 2010].
[6] "Blind SQL Injection " Kavin Spet White Paper. [Online] Available:
[7] "Second Order Code Injection Attack" Gunter Ollmann [Online] Available:
[8] "Web Application Attack Prevention for Tiered Internet Service " Susanta Nanda, Lap Chung Lam, Fourth Intenational Conference IEEE 2008.