ISSN 2320 - 2602
Volume 8 No.3, March 2019
International Journal of Advances in Computer Science and Technology
Available Online at http://www.warse.org/IJACST/static/pdf/file/ijacst01832019.pdf
https://doi.org/10.30534/ijacst/2019/01832019

Proposed Methods for Prevention of SQL Injection Attacks

Tariq Jamal
Department of Computer Systems Engineering
University of Engineering and Technology Peshawar Pakistan.

ABSTRACT

SQL injection is that kind of strategy in which SQL code is inserted into web-based applications that uses server-side database. Such web applications settle for user input like form then place these user inputs in the database requests. SQL statements are executed in such a manner that wasn't supposed or anticipated by the applying developer that tries to subvert the link between a webpage and its supporting database, therefore the database is tricked into execution malicious code due to the poor design of application. The proposed system depends on protection site at run time, before inclusion of user input with database by validating, encoding, filtering the content, escaping single quotes, limiting the input character length, and filtering the exception messages. The proposed answer is effectiveness and measurability additionally it's simply adopted by application programmers. For empirical analysis, we offer a case study of our answer and implement in hypertext mark-up language, PHP, My Sql, Apache Server and Jmeter application.

Key words: SQLIA, tautology, union queries, Database server.

1. INTRODUCTION

With the advent of internet, it brought a revolution in every sphere of life. The methods used by organization for business is changed due to internet. Nowadays many businesses which have worth of billions are carried out online under the umbrella of e-commerce. Equally, website applications are commonly used jobs portal, university portals. Due wide spread use of internet, many techniques and methods are introduced by hackers to exploit the security vulnerabilities in order to gain personal advantages. Hackers access to an application or a network over internet is very dangerous as they may even destroy the whole sit up by injecting malicious codes. Among these techniques, one is SQL injection.

In this paper, we will discuss different SQL injection attacks; how they are carried out and what is its goals. In the later section, we will propose a model and give some methods in order to prevent SQL injection attacks.

2. SQL Injection Attacks

SQL is a popular language that is utilized in designing of several database tools like Oracle, My SQL, Microsoft SQL Server etc. SQL aided as prone to attacks and hacker may use it for launching attacks, the attacks which are launched through SQL queries are termed as (SQLIA) SQL Injection attacks. In this method hacker insert SQL quarries into web-application to hack victim's database to view all database contents to the hacker.

A shocking increased is recorded in the recent times of SQL injection attacks which had affected thousands of websites around the globe. Following are some of the techniques by the help of which SQLIA could be performed.

2.1 Tautologies

In this method SQL code is inserted in conditional statements. The reason for this to make these conditional statements true. Through such method hacker want to bypass authentication or inject some parameters to get desired data from database.

For Instance in figure 1, if the hacker inject "or 1=1--" in the user field, it will bypass the login and password field and will let you log in as admin in the database, the query looks like as

```
Select login_id from users_id
Where
User_id="or 1=1--"
```

Figure 1: Tautologies

2.2 Logically incorrect queries

In such type of attack the goal of attacker to collect information of the backend database like number of tables, columns, rows etc by injecting incorrect queries.

For example, in figure 2, if we insert the below query it will possibly extract the information of first table and then it will convert table name into integer.

```
Select login_id from users_id
Where
Login_id=”and password=”
And
Id= convert (int(select top 1 user_id from sys objects where id=u))
```

Figure 2: Logically incorrect queries

2.3 Union query

In this attack the hacker uses union operators with original query and the hacker get desired data from a table. The main purpose of this attack to bypass authentication and get data.

For instance in figure 3, below code will first produce null set and then will return passwords for the user names and in the end both portions are made union to give the required result.

```
Select * from users-id
Where
Login_id= “union select password from user_infor
Where
User_id=’abc’—and password=”
```

Figure 3: Union query

2.4 Alternate encoding

The main goal of alternate coding (figure 4) is to flee from detection, the injected code is changed in such way that it's unobserved by defensive secret writing practices and so the attack is launched without being detected.

For example,

```
$login = mysql_query(“SELECT * FROM user WHERE (User_ID= ‘” . mysql_real_escape_string($_POST[‘User_ID’]) . “’)and (pass_word = ‘” .mysql_real_escape_string($_POST[‘pass_
word’]) . “’)”);
```

Figure 4: Alternate encoding

2.5 Piggy-backed query

In this type of attack the hacker attach additional query to original query. The main purpose of this attack is that attacker can modify data of victim’s database.

For example in figure 5, hacker attach the below query in order to modify the data:

```
SELECT Login_ID FROM users_ID WHERE login_ID=’john’
and pass_word=’’; DROP TABLE users – ‘AND ID=1223
```

Figure 5: Piggy-backed query

2.6 Stored procedure

Stored procedures as shown in figure 6 are mainly used for the purpose of escaping detection. In stored procedure, the text is injected in such a manner that its detection is not possible. Similarly, the code is modified in such a way that it is escaped from defensive coding practices.

For instance:

```
$login = mysql_query(“SELECT * FROM user WHERE (User_
ID= ‘” .mysql_real_escape_string($_POST[‘User_ID’]) . “’)
and (pass_word = ‘” .mysql_real_escape_string($_POST[‘pass_
word’]) . “’)”);
```

Figure 6: Stored Procedure

3. PROPOSED SYSTEM

The system should be modelled as when the user wants to run an application, it should be first checked for sql queries and then it should be allowed to interact with the database. If the sql queries which is entered by the user contains malicious codes and special characters then it should immediately be blocked in order to make the database safer.

4. PREVENTIVE MEASURES

Following are some of the preventive measure which when adopted the risk of SQLIA could be minimized.

4.1 Validation input

In any form process the first step is to check input data or syntax to check that the input data is valid. Verification may be checking whether or not the entered data for password is in the range of required range, is it a number or alphabet or guaranteeing the username column has exclusively substantial characters like a-z, A-Z and bounty of categories of input have mounted dialects such as dates, e-mails, etc.

4.2 Inputs Encoding

Inputs in some cases would possibly contain some ineligible characters, which might be feasible to check the entered inputs of users. For instance, in an exceptionally search field the user might sort something which the user is looking for, likewise especial tags i.e <script>, the valid technique to prevent harmful data from user input is encoding, since encoding make able to display the harmful characters. For Instance the less than character i.e < is translated to <. Machine-readable text mark-up language cypher

methodology of the server object is also conversant in encode the harmful characters.

4.3 Filtering input

Vulnerability mostly occurs when web application does not check data entered by user before it is used in a SQL query. Due to not checking the user input properly the server may receive incorrect inputs from untrusted persons. By using efficient language filtering system the malicious inputs could be blocked. for instance, assume to separate uncommon characters like [ ] ; &.to achieve this practicality use the Replace technique of the String object, this code can swap all the undesirable characters from the customers input.

4.4 Limiting length of user input

The quantity of charactertics in textbox can be controlled by using maxlength attribute. This is planned in proposed framework that all kind of fields and text boxes must be as short as possible. For instance, if the first name textbox is fifty characters, then limit the length by victimization the maxLength attribute. this could restrain the application to send very little set of commands back to the server.

<input type="text" id="txtFirstname" MaxLength="50" ru

4.5 Modification of Errors Reports

When the hacker has no clues regarding the underlying SQL queries or the contributory tables. In such circumstances the hacker uses false statements to get error messages from wherever assailant will gain some vital information referring to SQL statements and should begin intrusive with the SQL statement. In proposed framework error reports are properly processed in such means that error cannot be shown to outside users and show of errors ought to be limited to internal customers solely like

- ✓ **DISPLAY ERRORS= OFF**
- ✓ **DISPLAY ERRORS STARTUP= OFF**
- ✓ **LOG ERRORS= ON**
- ✓ **LOG ERRORS MAX LENGTH= ZERO**
- ✓ **IGNORE REPEATED ERRORS= OFF**
- ✓ **TRACK ERRORS= OFF**
- ✓ **ERROR LOG= SYSLOG**

4.6 Two Factor Authentication System

The authentication of database should be a two or multiple factors as it reduces the risk of exploitation of the vulnerability of database. For this purpose, password could be used in combination with finger print, retina scanning etc which will make the database more secure.

4.7 Avoid Privilege escalation

The avoidance of Privilege escalation should be ensured all the time; users should not be granted permission to run the database as database administrator by limiting the database permissions so that the vulnerabilities could not be exploited. For example, an editor should be limit to his own role; editing and he should not be given the authority to access the database as administrator.

5. CONCLUSION

A hacker can hack any system and might harm a network in several ways. it's up to web application developer that up to how much extent he could design a secure application. Generally, proposed system isn't restricted to any specific platform since it doesn't depend on any specific technology. This strategy could also be instantiated for any existing web application framework.

REFERENCES

1. Goheer,J. (2009), SQL Injection and Cros Site Scripting , KualitatemPvt Ltd.

2. Cova, M.; Felmetsger,V. and Giovanni Vigna,(2007), Vulnerability Analysis of Webbased Applications, University of California.

3. Buehrer, G.; Weide,W.; Paolo,A. and Sivilotti, G. (2010), Using Parse Tree Validation to Prevent SQL Injection Attacks" ,Computer Science and Engineering of Ohio State University .

4. Xiang, F.; Qian,K. (2008), SAFELI – SQL Injection Scanner Using Symbolic Execution,School of Computing and Software Engineering Southern Polytechnic State University .

5. Stephen,W. and Keromytis, D. (2010), SQLrand: Preventing SQL Injection Attacks, Department of Computer Science Columbia University .

6. Ronald,L. (2008), SQL Injection, Hong Kong Computer Emergency Response Team Coordination Centre.

7. Kruegel ,K. ; Valeur ,F. and Barbara,S. (2007), An Anomaly-driven Reverse Proxy for Web App lications, University of California.

8. Mrowton,K. (2005) , Introduction to SQL Injection, Lee Lawson

9. www.OWASP.com