Research of SQL Injection Attack and Prevention Technology

Li Qian
Institute of Information Engineering of Anhui Xinhua University
University of Science and Technology of China
Hefei, China
E-mail: wulianchongjing@qq.com

Zhenyuan Zhu, Jun Hu, Shuying Liu
Institute of Information Engineering of Anhui Xinhua University
Hefei, China
E-mail: qianli2014@hotmail.com

***Abstract*—SQL injection attack is one of the most serious security vulnerabilities in Web application system, most of these vulnerabilities are caused by lack of input validation and SQL parameters use. Typical SQL injection attack and prevention technologies are introduced in the paper. The detecting methods not only validate user input, but also use type-safe SQL parameters. SQL injection defense model is established according to the detection processes, which is effective against SQL injection vulnerabilities.**

Keywords: SQL injection; input validation; defence model; prevention technology

I. INTRODUCTION

SQL injection was an attack in which malicious code was embedded in strings that were later passed to database backend for parsing and execution. The malicious data produced database query results and acquired sensitive information, such as account credentials or internal business data [1]. At present, the standard definition of SQL injection technique was not yet fully established. Chris Anley discussed the various ways in which SQL could be injected into the application and resolved some of the data validation and database lockdown issues [2]. Because most web applications were associated with database backend, there were possibilities of SQL injection attacks on its [3].Through analyzing the principle of SQL injection attacks, prevention method was proposed to solve the double defense through the browser and server ends [4]. Jang exhibited a novel scheme that automatically transformed web applications, rendering them safe against SQL injection attacks [5].

Mishear demonstrated two non-web-based SQL injection attacks, and presented XML-based authentication approach which could handle this problem in some way [6]. Alhuzali stated that SQL injection attacks occurred when developers had combined hard-coded strings with user input queries [7]. The causes of SQL injection vulnerabilities were relatively simple and well understood. SQL Injection Attack was a class of code injection that took advantage of a lack of validation of user input [8]. SQL injection was a code injection technique, and was used to attack data-driven applications, in which malicious SQL statements were inserted into an entry field for execution [9]. Actifed Taper referred that SQL injection was a type of attack which the attacker entered illegal SQL code into web application. SQL injection vulnerability allowed an attacker to a web application's underlying database and destroyed functionality or confidentiality [10]. Li Qian stated that input validation needed to verify the client, the server and the URL security testing [11].

The issue is that some current techniques cannot be applied in practice because they cannot prevent typical attack or have not been implemented yet. So they have limitations that influence their effectiveness and practicability. Some of them need to modify web application code or additional infrastructures. Typical SQL injection attack and prevention technologies are introduced in the paper. The detecting methods not only validate input values but also use type-safe SQL parameters, which is effective against SQL injection vulnerabilities.

II. SQL INJECTION ATTACKS PRINCIPLE I

SQL databases are attacked against by direct insertion of code into input variables, which are consisted of the primary form of SQL injection. Some attackers insert malicious code into a string, the server will not respond input values when the input string contains such an SQL statement. The method to directly attack is to inject the ill-disposed codes into strings, which are usually stored in a table or as metadata. And then the attacking code will run when the attacked strings are called in the form of concatenated subsequently into a dynamic SQL command.

A. SQL Injection Attack

The following example shows how SQL injection attacks realize. SQL injections utilize weakness of a bank's application to misguide the application into running a database backend query or command. Usually, an application of a bank's operation has a menu, which is used for searching customer's personal information, such as the telephone number. The application will execute an SQL query in the database backend.

- SELECT client_name, sex, address, date_of_birth
  WHERE tel_no=123456

If user enters the string "123456 or 1=1," then the SQL query passes to the database as follows:

- SELECT client_name, sex, address, date_of_birth WHERE tel_no=123456 or 1=1

The condition 1=1 is always true in database. The query will return all rows in the table, which is not the original intention. The application can be changed so that it accepts one numeric value only. SQL injection attacks can be mitigated by ensuring proper application design, especially in modules that require user input to run database queries or commands.

The below script case shows a typical SQL injection. The script creates an SQL query by concatenating hard-coded strings together with an input string. The user is prompted to input the name of a city. If the user enters Seattle, the query is assembled by the script looks like the follows:

- SELECT * FROM OrdersTable WHERE ShipCity = 'Seattle'

However, assume that the user enters the following script:

- SELECT * FROM OrdersTable WHERE ShipCity =

  ' Seattle '; drop table OrdersTable--'

The semicolon (;) states the end of one query and the start of another query, the double hyphen (--) denotes that the rest of the present line is a comment and should be disregarded. If the edited code is syntactically correct, it will be executed by the server side.

When SQL Server runs the statement, SQL Server will select all records in OrdersTable where ShipCity is Seattle, and then SQL Server will drop OrdersTable.

As long as injected SQL code is syntactically correct, tampering cannot be captured programmatically. Therefore, all users input and check code that implements constructed SQL commands in the server should be verified.

B. Common Detection Method

SQL injection attacks are usually accessed to page port, which looks like ordinary Web page login. The general firewall cannot detect SQL injection attacks and therefore needs some artificial means to enhance the detection of injection attack. The detection methods are as following:

- Check IIS log

SQL attackers usually access particular page files, and then increase many log files. As IIS logs record information of user's IP address and access files, we can judge whether attackers has SQL injection attacks by looking at the log file size and content.

- Check database

Don't put the sensitive data into the database, if attacker uses SQL software tools to inject database, which will generate some temporary tables. We can check whether SQL injection attacks are happened by checking database table structure and content.

- Check user's input

Use regular expression or limit the length to validate user input string, and always check user's input by testing format, length, type, and range. When software testers are implementing precautions against malicious code input, consider the deployment and architecture scenarios of their application.

III. SQL INJECTION ATTACK PREVENTION METHOD

When software testers are implementing precautions against malicious input, not only validate user input by testing type, format, length and range, but also consider the architecture and deployment scenarios of their application. The following suggestions should be considered, and the validation items are listed in Table 1:

TABLE I. VALIDATION ITEM

No	validation item
1	Make no assumptions about the type, size or content of the data that is received by the application.
2	Check the content of string variables.
3	Accept only expected data, escape sequences, reject binary value, and comment characters.
4	Verify the size and value type of input and enforce appropriate limits.
5	Do not build Transact-SQL statements directly from user input.
6	Against its schema as it is input, when use with XML documents.
7	Use stored procedures to validate user input.
8	Do not concatenate user input because it is not validated.

A. Validate User Input

Filtering input maybe also helps to protect against SQL injection, such as: "insert| select| and|…" The following example removes escape characters.

Parameter Comment:

<param name = "InputWord"> Filter string </ param>,
<returns> If exists illegal characters, returns true </ returns>

Filter Code shows as follows:

```
Public static bool SqlFiltering (string InputWord)
{
  string
  inputdata="insert|delete|update|select|and|exec|chr|mid|
  master|or|truncate|declare|join|char";
  if(inputdata ==null){
        return false;
    }
  foreach(string i in inputdata.Split('|')){
        if((InputWord.ToLower().IndexOf(i+"")>-1)
        ||(InputWord.ToLower().IndexOf(" "+i)>-1)){
            return true;
        }
    }
        return false;
}
```

B. *Use SQL Parameters*

SQL Parameters provide length validation and type checking. One benefit of using the Parameters collection is that the input is regarded as a literal value rather than the code to be run. Another benefit is that type and length validation is enforced by client server. Values outside this range will trigger an exception. The following code fragment demonstrates the use of the Parameters collection:

- SqlDataAdapter myCommand = new SqlDataAdapter("StudentLogin", conn);
- myCommand.SelectCommand.CommandType = CommandType.StoredProcedure;
- SqlParameter parm = myCommand.SelectCommand.Parameters.Add ("@stu_id", SqlDbType.VarChar, 11);
- parm.Value = Login.Text;

In this example, the @stu_id parameter is treated as a literal value instead of as executable code.

This value is checked for length and type. If the value of @stu_id does not comply with the specified length and type constraints, an exception will be thrown.

C. *Use Parameterized Input with Stored Procedures*

Stored procedures are susceptible to SQL injection if they use unfiltered input. The following code is vulnerable:

- SqlDataAdaptermyCommand = new SqlDataAdapter("LoginStoredProcedure '" + Login.Text + "'", conn);

No matter whether or not to use stored procedures, the parameters should be used as input check. The code example is as follows:

- SqlDataAdaptermyCommand = new SqlDataAdapter("SELECT stu_lname, stu_fname FROM Authors WHERE stu_id= @ stu_id ", conn);
- SQLParameter parm = myCommand.SelectCommand.Parameters.Add("@stu_id ", SqlDbType.VarChar, 11);
- Parm.Value = Login.Text;

IV. SQL Injection Attack Defence Model

According to the above prevention technology, a defense model is established to prevent SQL injection attacks. The model not only validates input form information, but also detects web address bar information, especially for sensitive character detection. Firstly, server side checks IP address legitimacy. The user is denied to access the login server when the input values are illegal. Secondly, server side checks input values by testing format, length, type, and range. If the input string matches SQL prevention rule, then the user is allowed to access the page. Finally server side verifies the user privilege. If the user's number exceeds the access permissions, then the user is blocked timely and the system sends a message to system administrator. Server records the injection attack when all verifications are invalid.

The SQL automatically defense model shows as Figure1:

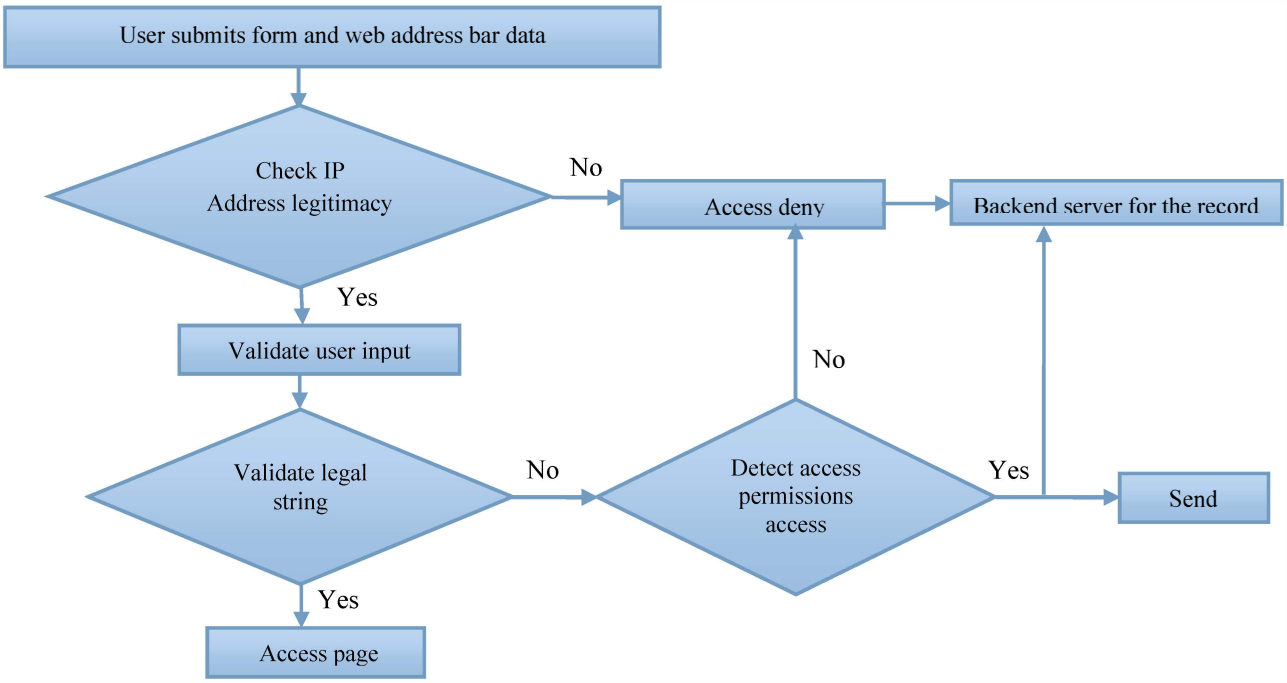

Fig. 1. SQL defense model

V. CONCLUSIONS

SQL injection attack is a serious security problem in Web application system, most of which are caused by lack of input validation and SQL parameters use. And SQL injection attacks make use of such software defect to access the target databases from user's client. Moreover, the flaws during an input validation of web components can also be utilized for an attacking chance.

This paper shows the characteristics of the SQL injection attack and the prevention methods. The SQL injection attack can be prevented by input validation and type-safe SQL parameters methods. Based on this, a defense model is established to prevent SQL injection attacks, which detects attacks by comparing the input size of intended queries with the actual ones. The technique can be utilized for protecting web application.

ACKNOWLEDGMENT

This work was supported by the Education Department of Anhui Province Natural Science Key Research Project (No. KJ2014A100); by the Anhui Xinhua University project (No.2013jgkcx05, No.2013xqjdx02)

REFERENCES

[1] Kim, Mi-Yeon; Lee, Dong Hoon.Data-mining based SQL injection attack detection using internal query trees.[J]Expert systems with applications. 2013,9:416-430

[2] Anley C. Advanced SQL injection SQL sever application.[EB]. http://www.creangel.com/papers/advanced_sql_injection.pdf.

[3] Mittal, Piyush. A fast and secure way to prevent SQL injection attacks.[C]2013 IEEE Conference on Information and Communication Technologies, ICT 2013, p 730-734

[4] Meng Ting. MySQL Injection Attacks and Defense Methods. [J]Information Security and Technology,2013.11:26-38

[5] Jang, Young-Su; Choi, Jin-Young,Detecting SQL injection attacks using query result size[J]COMPUTERS & SECURITY,2014.44: 104-118

[6] Mishra, Nitin; Chaturvedi, Saumya; Sharma, Anil Kumar.XML-Based Authentication to Handle SQL Injection.[J]Advances in Intelligent Systems and Computing.2014,236.: 739-749

[7] A. Alhuzali, H. Tora, Q. Nguyen. Analysis of SQL injection methods and its prevention.[C]Final project of ISA 564 Security Lab (2012)

[8] A. Tajpour, S. Ibrahim, M. Sharifi. Web application security by SQL injection detection tools[J]Int J Comp Sci Issues, 2012.9: 332–339

[9] Microsoft."SQLInjection". Retrieved 2013-08-04. http://technet.microsoft.com/en-us/library/

[10] Atefeh Tajpour, Maslin Massrum. Comparison of SQL Injection Detection and Prevention Techniques.[C]2010, the 2nd International Conference on Education Technology and Computer, ICETC 2010: 174–179

[11] Li Qian, Jiahua Wan, Lu Chen, Xiuming Chen. Complete Web Security Testing Methods and Recommendations.[C]2013, the 1st International Conference on Computer Sciences and Applications, CSA2013: 86–89