International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal
ISSN No: 2456 - 6470 | www.ijtsrd.com | Volume - 2 | Issue – 4

SQL Injection Attack Detection and Prevention Techniques to Secure Web-Site

Mr. Vishal Andodariya
Assistant Professor
Vadodara Institute of Engineering, Kotambi, Vadodara, Gujarat, India

ABSTRACT

Structured Query Language (SQL) Injection is a code injection technique that exploits security vulnerability occurring in database layer of web applications [8]. According to Open Web Application Security Projects (OWASP), SQL Injection is one of top 10 web based attacks [10]. This paper shows the basics of SQL Injection attack, types of SQL Injection Attack according to their classification. It also describes the survey of different SQL Injection attack detection and prevention. At the end of this paper, the comparison of different SQL Injection Attack detection and prevention is shown.

Keywords: *SQL Injection, Types of SQL Injection, Tokenization*

I. INTRODUCTION

SQL Injection Attack is used to get unauthorized access to web application. It can be used to disclose database of web application also. Success of SQL Injection is due to lack of improper handling of input validation or bypass of user validation. Bypass of user validation can be of disabling the client side validation such as disabling java script in web browser.

II. TYPES OF SQL INJECTION [8]

SQL Injection can be categorized into SQL manipulation, code injection, functional call and buffer overflow. SQL manipulation means attacker attempts to modify the existing SQL statements. Code Injection attempts to add additional SQL statements or commands to the existing SQL statements. Functional call is the insertion of database function into user input. Buffer Overflow will make the loss of database connection.

A. SQL Manipulation

It can be classified as below:

*1) Tautology***:** Tautological attack will inject code in original SQL statement. The additional code is in the way that it always evaluated to be true.

For example :- Select name from user where id=”abc” or 1=1

*2) Interface***:** In this type of attack, the attacker will try different forms of injection query. According to different code injection, attacker will infer that what kind of queries is injectable. Also if application will display database error message, attacker could know which parameters are injectable.

3) Union Queries: Attacker will inject second query using the UNION keyword of SQL. Attacker uses Union query to retrieve data from table defined in second injected query.

4) Piggybacked Queries: In this attack, attacker will insert or inject second query using “;” (semicolon). They insert a new query which is not related to first query.

B. Code Injection

This attack can be classified as follow:

1) LIKE Queries: In this attack, user input is given in the LIKE parameter of the query, which subverts the query.

2) Column Number Mismatch.: These types of attacks are used to gain knowledge about tables. By trying different number of columns in the input query, the attacker will infer that what numbers of columns are present in database table.

3) Additional Where Clause: This means that there are many additional *where* condition statement that can be added using 'OR' or 'AND' keyword.

4) Insert Subselect Query: The use of advanced insert query can be used by attacker to access records of the database.

C. Functional Call Injection

Function call injection is the insertion of database function into a user input. These functions are used to manipulate data in database.

1) Role foundation: The query string of user request is manipulated by attacker to gain information from web application.

2) System Stored Procedures: Attacker tries to execute Database Stored Procedure. If the backend server (Database Server) is known, the attacker will use system stored procedure.

D. Buffer Overflow

In this type of attack, the attacker will overflow buffer of database server so that the more requests for database are blocked. This can be implemented by using inbuilt functions like DELAY of database server. Which are executed in system and creates delay in execution of server.

III. LITERATURE SURVEY

A. Preventing SQL Injection Attacks [5]

In this paper, authors have proposed a technique in which user authentication is carried out in different database level. First is Relational database level and second is XML database level.

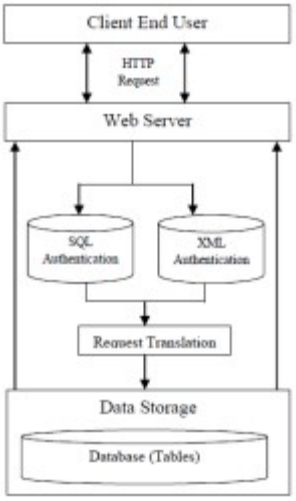

Fig 1. Two Level Authentication Architecture [5]

The query is passed to both database and processed. If both database results are same then the query is correct or legal database request and user is authenticated person. If both the results are not same then the query is malicious and user is attacker. The database is replicated in XML format.

B. SQL Injection Attacks: Technique and Prevention Mechanism [4]

This paper uses the query tokenization to validate the query containing the user input. Also the concept of double authentication of previous paper is implemented. In query tokenization process, the query is tokenized using query parser. Process is as below:

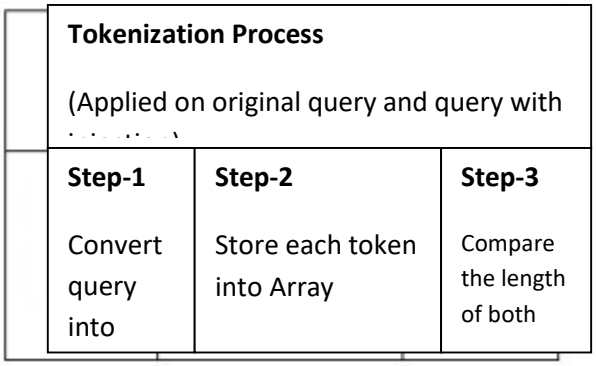

Fig 2. Tokenization [4]

I. Convert query into Tokens

II. Store each token into array

III. Compare the length of both

If lengths of both array is same, there is no SQL injection otherwise there is SQL injection. Next are the two level authentications. At first, query is passed

to relational database and at second query is passed to XML database. If both results returned from database are same then query is valid otherwise not.

C. Neutralizing SQL Injection Attack Using Server Side

Code Modification in Web Applications [3]

Step 1: From the SELECT query, all characters after the WHERE clause are extracted and stored in a string S1.

Step 2: Input parameters are accepted from the user. The

parameters are checked for their appropriate type. If the input type matches the required type, the input parameters are added to the query. Otherwise, the parameters are rejected, and the page is reloaded with a warning message of "Invalid Parameters."

Step 3: The query string is normalized to convert it into a simple statement by replacing the encoding if any.

Step 4: Using the string extraction method all characters after the WHERE clause are extracted.

Step 5: The input parameters from the extracted string are removed sequentially as they were added. For numeric parameters, we remove the numbers and, for alphanumeric parameters, we remove the characters enclosed in single quotes. The new string is named as S2.

Step 6: Strings S1 and S2 are compared if they match and then it is considered that there is no injection attack, and the query is sent to the database server for execution. Otherwise, the query is dropped and the page is reloaded with a warning

message of "The user is trying for SQL Injection!!!"

D. A System to Detect and Block SQL Injection with the help of Multi-Agent System using Artificial Neural Network [2]

Whole system is based on the last paper Multi-Agent system for Detecting and Blocking SQL Injection. The system is extended with the use of Artificial Neural Network. Artificial Neural Network is used in the process of classification. It contains Retrieve, Reuse, Revise and Retain phases. Retrieve phase will retrieve query from the cache memory, which were used for a similar query in accordance with attributes of the new case. There are 12 fields of previously fired query are stored in database table. Condition is that at least 8 fields must be match with rows in database. Output of Retrieval phase is passed to Reuse phase. Reuse phase gives input to ANN which is passed by Retrieval phase. In Revise phase, if output generated by the ANN is greater than 0 then the query is declared as legal. However if the output is less than 0 then the query is declared as suspicious. At last, Retain phase is responsible to store the output generated by ANN to database for training purpose.

E. The Multi-Tier Architecture for Developing Secure Website with Detection and Prevention of SQL Injection Attacks[1].

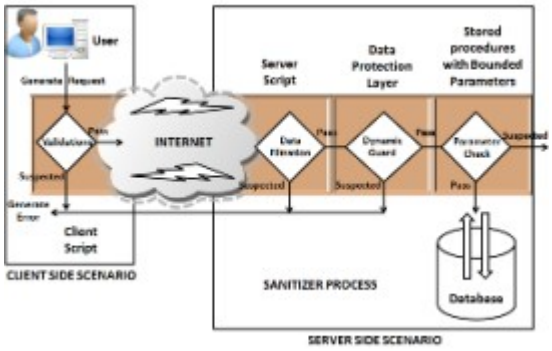

Fig 4. Multi Tier Architecture [1]

Proposed technique of this paper consists of four stages to detect and prevent SQL Injection. First stage is Validation Stage. At this stage, the client side validation is carried out. It detects and regular expression mechanism to validate user input. Second stage is Data Filtration Stage. This prevents the use of special characters. It uses the stage is responsible for analyzing code written behind the design page. This stage detects the malicious SQL code inject by other techniques like query strings, URLs. Third stage is Dynamic Guard Stage (Sanitizer). Parameterized query is parameterized database access API provided by development platform. This parameterized statement can be of Java such as PrepareStatement or SQLParameter in .NET. Instead of concatenating string with parameter in SQL query, we can define a place holder for user input separately. Fourth stage is Parameterized Check Stage. This stage keeps track of parameter and query separately. Parameterized query separates both query and input data. This technique helps to not change structure of SQL statement.

F. Random4: An Application Specific Randomized Encryption Algorithm to Prevent SQL Injection [6]

Table 1. Lookup Table for Random4 [6]

	R[1]	R[2]	R[3]	R[4]
a	;	l	x	W
…				
z	7	k	@	U
A	i	J	)	0
…				
Z	M	6	f	.
0	9	B	g	"
…				
9	c	R	j	l
@	(	a	5	0
…				
-	6	a	H	K

This paper at basic suggests to use client side validation such as Java script to validate user input. But it does not prevent attack to the application. So they suggest Encryption technique for the database. Random4 algorithm uses lower case, upper case, numeric and 10 special characters for encryption of the database. The algorithm chooses one of four different values for single character. The encryption character is chosen from the lookup table based on the next input character in input string. The encrypted string and original string are concatenated and stored in database. This encrypted key is used to give table name and column name of database. Also this system encrypts the input given by client. So the encrypted input would be in the format that is not executable code of SQL language.

IV. COMPARISON

The method explained in section III(A) with the technique of two level authentications is slower process. Moreover it can't handle UNIOUN QUERIES [7]. This system passes the SQL query to both relational and XML database. So we need two results from database. It will take time of about two requests. Next method explained in section III(B) uses the tokenization and two level authentications. It is also slower technique as uses two level authentications. Also uses the tokenization method which also adds some overhead in response time. Section III(C) mainly works on input given by user, it filter out the input and checks its original query after getting input from user but before executing the query. Section III(D) expands the system of III(C) with the artificial neural network. The neural networks need training to operate. Requires high processing time for large Neural Networks, which is slower process [9]. Section III(E) with multi-tier architecture gives technique at different layer of defense. This system is faster than previous techniques as it includes client side scripting and prepared statement. Section III(F) is also powerful technique as it encrypts the database names. This process is carried out when database is created. And also ensures that input passed from users is in encrypted form and this encrypted form will not be SQL query or SQL code.

V. CONCLUSION

In the web application system the response time of web request is more important. The request sent by the client should be processed quickly. By studying the different methods we can conclude that the system explained in section 3.6 is more powerful as it prevents attack with encrypted database. And encryption is carried out when the database is created. And also the input is encrypted and which would be in the format that cannot be executable code of SQL language.

VI. REFERENCES

1) Praveen Kumar, "The Multi-Tier Architecture for Developing Secure Website with Detection and Prevention of SQL Injection Attacks," International Journal of Computer Applications (0975 – 8887) Volume 62– No.9, January 2013.

2) Niraj Kulkarni, D R Anekar, Mayur Ghadge, Rohit Garde," A System to Detect and Block SQL Injection with the help of Multi-Agent System using Artificial Neural Network," International Journal of Computer Applications (0975 – 8887), Volume 71– No.12, February 2013.

3) Asish Kumar Dalai, Sanjay Kumar Jena, "Neutralizing SQL Injection Attack Using Server Side Code Modification in Web Applications", Security and Communication Networks Volume 2017, February 2017.

4) Gaurav Shrivastava, Kshitij Pathak, "SQL Injection Attacks: Technique and Prevention mechanism," International Journal of Computer Applications (0975 – 8887), Volume 69– No. 7, May 2013.

5) Asha. N, M. Varun Kumar, Vaidhyanathan G, "Preventing SQL Injection Attacks", International Journal of Computer Applications (0975 – 8887), Volume 52– No.13, August 2012.

6) Srinivas Avireddy, Varalakshmi Perumal, Narayan Gowraj ,Ram Srivatsa Kannan,Prashanth Thinakaran, Sundaravadanam Ganapathi, Jashwant Raj Gunasekaran and Sruthi Prabhu, "Random4: An Application Specific Randomized Encryption Algorithm to Prevent SQL Injection," IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications,IEEE-2012.

7) Kai-Xiang Zhang, Chia-Jun Lin, Shih-Jen Chen, Yanling Hwang, Hao-Lun Huang, Fu-Hau Hsu," TransSQL: A Translation And Validation based Solution for SQL-injection Attack", First International Conference on Robot, Vision and Signal Processing, IEEE-2011.

8) Khaleel Ahmad, Jayant Shekhar and,K.P. Yadav, " Classification of SQL Injection Attacks", Vol. I (4), 235-242, VSRD-TNTJ-2010.

9) ANN,http://www.learnartificialneuralnetworks.com/introduction-to-neural-networks.html-13/04/2018

10) "Top Ten Most Critical Web Application Vulnerabilities,"OWASP Foundation, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project-.10/12/2013