International Journal of Engineering & Technology, 7 (2.6) (2018) 213-216

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research Paper

Proposed Method for SQL Injection Detection and its Prevention

Ashish Kumar[1]*, Sumitra Binu[2]

[1]Department of Computer Science, Christ University, Bangalore, India
[2]Department of Computer Science, Christ University, Bangalore, India
**Corresponding author E-mail: ashishalbert9@gmail.com*

Abstract

SQL injection attack is a commonly used method to attack the database server. Injection attacks enable the attacker to bypass the validation and authorization mechanisms used by database server and gain access to the database. The easiest way to launch this attack is by exploiting the loopholes in the validation of user inputs provided through login pages. Each login page that a user visits can contribute towards revealing the identity of the user. Feedbacks given by the server while executing an SQL code can reveal information regarding the vulnerabilities in the validation process of the database server. This information can be misused by the attacker to launch an SQL injection attack. This paper discusses a technique for identifying and preventing SQL injection attack using tokenization concept. The paper discusses a function which verifies the user queries for the presence of various predefined tokens and thereby preventing the access to web pages in cases where the user query includes any of the defined tokens.

Keywords: *SQL, SQL Injection attacks, SQL Injection Vulnerability, Tokenization.*

1. Introduction

The SQL injection, is a type of database threat which is used to penetrate website and get access into the database. SQL Injection Attacks (SQLIA) are launched to gain access to databases containing sensitive data, by penetrating websites with security loopholes. This is a critical attack which can bypass many security layers like encryption & firewall and is launched by exploits the vulnerabilities in input validation. This attack can easily bypass the database components. SQL injection is a simple attack that can be launched without much effort by logging in via web pages that does not provided proper validation of user inputs. As part of the login and authentication process, the user gives his/her username and secret information such as password for verification. The provided user inputs are translated into an SQL statement by the web application and attack of SQL injection is carried out by SQL statements which can bypass the validation procedure [1].

SQL is a query based scripting language which will allow the users to access the database and SQL injection attack may provide unauthorized access to the database server. In this attacks which involves client's input is dealt with as SQL code. The means of an SQL can enable client to access database through PHP or PERL, by providing fundamentalqueries. If the provided knowledge given by the user, sent right to the database and which is not properly tested and verified, then the vulnerability can be misused by the client by inserting malicious SQL code. The attackers can directly execute and run SQL queries to the database which may lead to exploitation of the data, for example it may execute change or delete query which carried forward to irrecoverable and inaccessible of the data. In more critical condition the remote code can be executed and the data which is stored in the database can be accessed by the attackers.

Though, the system admins are less prior to know about the attacks of a user because Application Program Interface (API) can be utilized by the attacker, which help them to execute the undesirable query. This case represents the vulnerability situation which cause serious threats on web platform, it processes the user SQL queries which are used to retrieve information from a database. A vast majority of the web applications are susceptible to vulnerabilities in validation of user inputs and hence are defenseless against SQL injection. SQLIA is moderately simple to perform and hard to prevent.

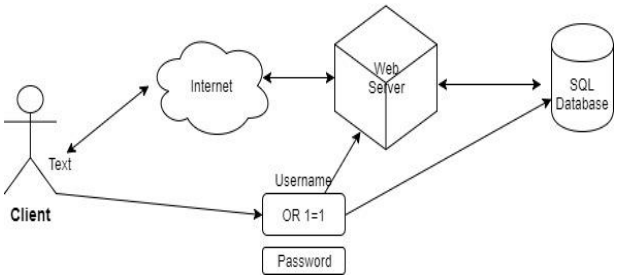

Fig 1. SQL injection Schema

The user inputs are utilized for making SQL query which is exchanged with the database. If the entered values are found as expected the user's access is allowed otherwise access will be denied [1]. The attacks of SQL injection attacks are in many forms which includes Error-based Injection, Tautology based Injection, Union-based Injection and Blind SQL injection. As shown in Fig 1 in tautology based attack user gets unauthorized access into the database by including a tautology that always evaluates to true in the query. In the example illustrated in Fig. 1, users get unauthorized access to the system as the tautology 1 = 1 is always true, hence if

inputs given by the users are not properly sanitized, then it may lead to a critical web security attack which describes the principle involved in SQL injection [2]. To understand how the attack can be successfully launched via a web application, consider that the application has a LOGIN page titled as "Login.html/aspx". The attacker will attempt to login to the application by providing "Admin or 1=1" or alternately any characters via the text box control meant for accepting username and password.

2. Literature review

Xiang Fu, Kai Qian, Lixin Tao, in their work discusses a static analysis framework that identifies the SQLIA vulnerabilities during compilation time. This static analysis tool will detect SQLIA on web applications developed using ASP.NET. The tool (SAFELI) uses a hybrid constraint solver at each hot spot to inspect the submitted user query and to identify the user inputs that could contribute to an SQL injection attack [3].

A mapping model was proposed by Pandurang and Karia [4] to detect and prevent injection attack. In the discussed model each client has its own container which will restrict the damage caused by an attack.

An Intention oriented approach on SQLIA detection was proposed along with an intention description language by Chenyu and Fan [5]. Deterministic Finite Automat (DFA) is used to represent SQL strings and for checking whether the SQL query belongs to the string.

Abirami J., Devakunchari R. and Valliyammai [6] in their work, they discussed about the major key characteristics of SQL injection and how to prevent such attacks by implementing proper validation of inputs. The authors have proposed a defensive model for prevention SQL injection attack, which deals in to detect and identify the attacks by the method of comparing input size of a string intended queries with original values. This web application technique can be utilized for preventing attacks.

Gudipati V., Venna T., Subburaj S. and Abuzaghleh [7] in their work have done the survey and analysis of SQL injection attacks, and according to the author no existing solution is primarily utilized in real time for detecting SQL injection attack.
The Raja P.K [8], have proposed a dynamic technique of query matching which quickly detects the attack. The SQLI (Structured Query Language injection) sanitizer is the process to minimize the processing time by the approach of dynamic query matching. The Sanitizer also do the future analysis which verify the unwanted SQL attacks and dump them.

The XSS (Cross Site Scripting) attacks mainly target on view side i.e. client side of the web app/site and tries to hoax the internet users which lead them to a security breach. In Paper [9], the authors consider a three-tier web app for the security with dynamic and static behavior. These behavior is created to detect the anomalies in SQL Injection class and XSS attacks.

Methods used for gathering information to launch SQLI attacks, are explained by Sonewar and Thosar [10]. Exploiting vulnerabilities in web applications using payloads are explained by the authors from the perspective of kali.

3. SQL Attack- An overview

SQL injection is a type of attack in which the attacker provides inputs that cause malicious Structured Query Language code to be executed on the database of the web application. These queries can bypass the defensive mechanism provided by firewall and permit the attacker to gain access to the database or make changes to the database contents. This attack is launched by the unauthorized users to get the sensitive information stored in the database. The attacker adds SQL statements via the input fields of the web application. The lack of implementation of proper validation mechanisms causes hackers to be successful in penetrating the database. Through the login page of the web application the hackers try to input malicious SQL queries to get unauthorized access to restricted data.

SQL Injection are classified into major categories they are: Firstly Union Query, it is the technique were attackers join injected queries through the UNION and they can get the data from the stored tables in the database. SQL manipulation method is done by changing "where" clause of SQL statement. Out-of-band attack is used as a secondary channel to dump the output. Piggy-Backed attack works by modifying data. Secondly Boolean-based SQL Injection, it is a technique that depends on sent queries to the database, which will force the system to response a various results whether query returns False or True. Thirdly Time-Based-Blind SQL injection, it is the technique which is similar to the Boolean-based SQL injection but before response it forces the database to wait for certain amount of time. The response time will indicate the result to the attacker in the form of TRUE or FALSE query. The lack of the implementation of proper validation causes hackers to be successful in penetrating the database. Through the login page of the web application the hackers try to input malicious SQL queries to get the access for the web application.
If the condition mentioned in the query is true it will be executed. SQL injection is the utilization of the user input fields to gain entry to the database. The blind SQL attack is a form of threat that checks for the true or false condition in the database and determine the solution based on the program response. To illustrate this attack consider a shopping website address which display items for sale. viz http://www.ashishshop/item.php?id=24 To choose an item with id = 24 we use SQL statement "Select name, price FROM Item_table where id = 24". To manipulate this query the attacker modifies the URL as http://www.ashishshop/item.php?id=24 and 1=2 and the corresponding SQL statements will be "Select name, price FROM Item_table where id = 24 and 1=2" The execution of this query will return the value False and hence, no items are displayed in the list. The attacker repeats the attack by changing the request to http//:www.ashishshop/item.php?id = 24 and 1=1. This will return True and the details of the item with ID = 24 will be displayed proving the successful execution of blind SQL injection attack.

4. Proposed Method

This paper proposes a technique which can prevent Blind-SQL injection attack. Here we are using tokenization concept where in a user defined function like ('/[\'^£$%&*()}{@#~?><>,|=_+¬-]/',$query)) will detect a blind SQL injection attack on a web application and prevent the attack by blocking the IP address of attacker for a specific time period. The proposed mechanism is tested on the Login/Registration page of few selected websites and it has been successful in detection of injection attacks.

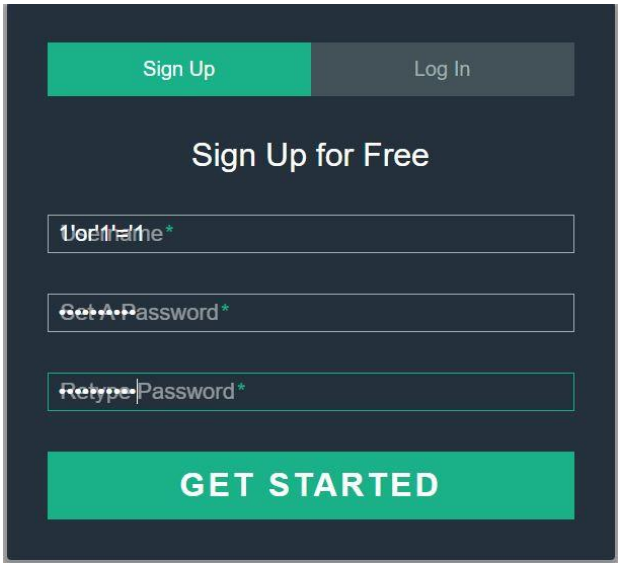

Fig 2. Login/Registration page

In this method if the attackers give an invalid input as shown in Fig.2 via the registration page, the application will display the message "Are trying to do attack (SQL Injection)" The attack is prevented by blocking the corresponding IP address for a certain period and the user will be notified about the blocking by an appropriate message as shown in Fig.3.

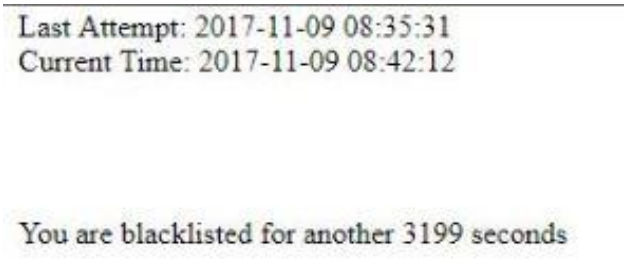

Fig 3. Blacklisted System Output

5. Implementation

The proposed method includes tokenization concept where in a user defined function will check the queries submitted by the user with a pre-defined set of tokens. If there is a match then the application will display the alert message that the attacker is attempting an SQL injection attack. The application will block the IP address of the attacker for a certain time period. The following code describes the mechanisms for detecting SQL injection and for blocking IP address. The Script is written in the Sublime Text which is cross-platform source code editor. Its open source tool which contain 22 visual themes.

```
<?php
function sqlia_real($query)
{
try
{
if(preg_match('/[\'^£$%&*()}{@#~?><>,|=_+¬-]/',$query))
{
include('connect.php');
$ip_address=$_SERVER['REMOTE_ADDR'];
$current_date=new DateTime( 'NOW' );
$current_timestamp= $current_date->format('Y-m-d-H-i-s');
$sql = "INSERT INTO ip(ip, attackTime) VALUES ('$ip_address','$current_timestamp')";
mysqli_query($conn, $sql);
echo 'You are trying to do sql injection attack on this web site';
?>
```

In this PHP code, line number six of this code is having defined function which will compare the Injection queries with the defined tokens. The defined function will match with attacker query when attacker gives any kind of SQL query that includes the pre-defined tokens. When the tokens defined in the function matches with those in the query posed by the attacker then it will show injection attack.

The proposed mechanism recognizes the malicious queries given by the attacker as an input during the login/registration process using the tokenization concept. Then this code $ip=$_SERVER ['REMOTE_ADDR']; will store the IP address of the machine and keep IP address in the database. If again the user types the same or another malicious query then it will again capture the IP address of the machine and will compare with the stored IP in the database. If both IP addresses match then application will display a message that the user is attempting to do an SQL injection and the IP address of the user will be blocked. The verification of the process which is performed twice before allowing access to the site is shown in Fig.4.

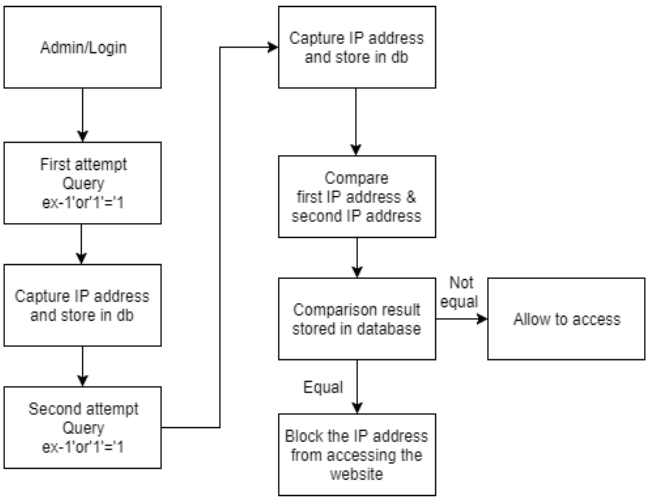

Fig 4. Process Diagram

6. Conclusion

SQL Injection attack is launched by attackers to gain unauthorized access to the database thereby accessing the sensitive information stored in the tables. In this paper, we have suggested a technique to detect Blind SQL injection attack with the help of defined function that makes use of tokenization concept and prevent the attack by blocking the IP address of the attacker. In future this can be helpful for protecting the website.

References

[1] Voitovych O.P, Yukovetskyi O.S., "SQL Injection Prevention System", IEEE International Conference Radio Electronics & Communication, 2016

[2] Srinivas A., Varalakshmi P., "An Application Specific Randomized Encryption Algorithm to Prevent SQL Injection", International Conference on Trust, Security and Privacy in Computing and Communication, IEEE.

[3] Xiang Fu, Xin Lu Boris, PeltsvergerShijunChen,"A Static Analysis Framework for Detecting SQL Injection Vulnerabilities", International Computer software and Applications conference, 2007.

[4] Pandurang R. and Karia D., “Impact analysis of preventing cross site scripting and SQL injection attacks on web application”, IEEE Bombay Section Symposium (IBSS), 2015.
[5] Chenyu M. and Fan G.,”Defending SQL injection attacks based-on intention-oriented detection”, 11th International Conference on Computer Science & Education (ICCSE), 2016.
[6] Abirami J., Devakunchari R. and Valliyammai C.,”A top web security vulnerability SQL injection attack”, Seventh International Conference on Advanced Computing (ICoAC),2015.
[7] Gudipati V., Venna T., Subburaj S. and Abuzaghle-hO.,”Advanced automated SQL injection attacks and defensive mechanisms”,Annual Connecticut Conference on Industrial Electronics, Technology & Automation (CT-IETA), 2016.
[8] Karuparthi R. and Zhou B.,”Enhanced Approach to Detection of SQL Injection Attack”, 15th IEEE International Conference on Machine Learning and Applications (ICMLA), 2016.
[9] Li Qian, Zhenyuan Zhu, Jun Hu and ShuyingLiu,”Research of SQL injection attack and prevention technology”, International Conference on Estimation, Detection and Information Fusion (ICEDIF), 2015.
[10] Sonewar P. and ThosarS.,”Detection of SQL injection and XSS attacks in three tier web applications”, International Conference on Computing Communication Control and automation (IC-CUBEA), 2016.