International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 10, Issue 2, March – April 2021 ISSN 2278-6856

Preventing Web Database from SQL Injection Attacks

Mr. Omkar Singh[1], Dr. S. K. Singh[2], Mr. Haroon Khan[3]

[1] Thakur College of Science and Commerce, Thakur Village, Kandivali – E, Mumbai – 400101

[2] Thakur College of Science and Commerce, Thakur Village, Kandivali – E, Mumbai – 400101

[3] Thakur College of Science and Commerce, Thakur Village, Kandivali – E, Mumbai – 400101

Abstract : SQL Injection Attack causes a very serious security issue over web applications or websites. In this attack, Attacker is able to take benefit of poorly coded Web application software to put malicious or unwanted code into the organization's systems and network. The vulnerability exists within web application when a Web application does not provide proper validation or filtering for the input data entered by the user in the Input fields. In today's world there are large numbers of web application which are having many input fields where Hacker can get chance to attack as a SQL Injection (E.g. To dump the database contents to the attacker). So Attacker can access the confidential data of the organization. We are going to present a project of SQL Injection attack, detection and prevention techniques in this web form project .we have two pages one is vulnerable and second one is secure It Targets the back end data stores through web application inputs like forms, URLs etc.

Keywords*- Tautology, Injection attack, vulnerability, prevention.*

1 INTRODUCTION

The Internet has brought about many changes in the way organizations and individuals conduct business, and it would be difficult to operate effectively without the added efficiency and communications brought about by the internet. In the last few years, the popularity of web-based applications has grown tremendously.

A number of factors have led an increasing number of organizations and individuals to rely on web-based applications to provide access to a variety of services. Today, web-based applications are routinely used in security critical environments, such as medical, financial, and military systems.

Because of the popularity of these types of applications many techniques to exploit their security vulnerabilities are potentially quite dangerous. One such technique is called SQL injection.

SQL Injection attack has been one of the major threats to the security of web applications and attackers can trick server into executing malicious SQL code which is occurs when user input is parsed as SQL tokens, thus changing the semantics of the underlying query. SQL injection attacks have been used to extract customer and order information from e-commerce databases or bypass security mechanisms. The intuition behind such attacks is that predefined logical expressions within a predefined query can be altered simply by injecting operations that always result in true or false statements.

This paper, presents a runtime technique to prevent SQL injection observe that all SQL injections alter the structure of the query intended by the programmer and by capturing this structure at runtime, we can compare it to the parsed structure after inserting user-sup plied input, and evaluate similarity . Evaluated the proposed system based on user input on a set of real-world applications without requiring a call to the database, thus lowering runtime costs and satisfy the following three criteria:

1.1 Prevent the possibility of the attack.
1.2 Minimize the effort required by the programmer.
1.3 Minimize the runtime overhead.

2 Related Work

SQL Injection Attack causes a very serious security issue over web applications or websites. In this attack, Attacker is able to take benefit of poorly coded Web application software to put malicious or unwanted code into the organization's systems and network. The vulnerability exists within web application when a Web application does not provide proper validation or filtering for the input data entered by the user in the Input fields. In today's world there are large numbers of web application which are having many input fields where Hacker can get chance to attack as a SQL Injection (E.g. to dump the database contents to the attacker. So Attacker can access the confidential data of the organization. We are going to make a project of SQL Injection attack, detection and prevention techniques in this paper .It Targets the back end data stores through web application inputs like forms, URLs etc.

Now a day's most web applications are being hacked using SQL Injection attacks method. This comes under top ten security threat in web applications. SQL Injection Attack is most serious attack because it is categorized Under Open Web Application Security Project(OWASP) Top 10 Attack with high risk, Fix ability as medium having the impact on server as data loss or corruption of DB, lack of responsibility or denial of access to DB. It can sometimes lead to the complete host take over. SQL Injection Attack is

most effective method for accessing the data from DB (Backend), with the help of this attack attacker can reach to the database and lift or steal sensitive information. The cause of SQL Injection is Malicious User input being treated as real DB Query. Commonly this attack is generated from web input so we can say it also an input validation attacks. With the help of this attack Hacker can get table details, database schema, and also Hacker can employ DML statement from the supplied input filed to web application to the database server which resulting a corrupt/ modified database.

SQL Injection has become a common issue with database driven web application. The attack is easily detected, and exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind. Essentially, the attack is carried out by placing a Meta character into data input fields to then place SQL commands in the control plane, which does not exist there before. SQL injection attacks allow hacker to spoof identity of system, make the changes within existing data, which causes repudiation issues such as changing balances or tampering to transaction, allow the complete disclosure of all data on the system, delete the data or make it otherwise unavailable, and become administrators of the database tier. SQL Injection is very common with various kinds of languages like PHP and ASP.

Applications due to the prevalence of older functional interfaces. Due to the interfaces available in the JAVA and .NET applications are less vulnerable to have easily exploited SQL injections. Depending up the attacker's skill and imagination, such as low privilege connections to the Database server and so on, the severity of SQL Injection attacks is limited.

We reviewed a number of electronic journal articles from IEEE journals and from ACM, and gathered some information from web sites to gain sufficient knowledge about SQL injection attacks. Following are the papers from which we covered different important strategies to prevent SQL injection attacks.

2.1 From "Using Parse Tree Validation to Prevent SQL Injection Attacks" ACM, I covered the techniques for sqlinjection discovery. This paper also covered very well the SQL parse tree validation that we mentioned in report (Gregory T. Buehrer Bruce W. Weide, and Paolo A. G. Sivilotti, 2005).

2.2 From "The Essence of Command Injection Attacks in Web Applications" ACM, they covered the techniques to check and sanitize input query using SQLCHECK, it use the augmented queries and SQLCHECK grammar to validate query. (Zhendong Su and Gary Wassermann, 2006).

2.3 From "Using Automated Fix Generation to Secure SQL Statements" IEEE CNF, they covered brief background, SQL statement, and vulnerability replacement methods (Stephen Thomas and Laurie Williams, 2007).

2.4 From "Automated Protection of PHP Applications against SQL-injection Attacks" they covered an original method to protect application automatically from SQL Injection attacks. The original approach combines static analysis, dynamic analysis, and automatic code re-engineering to secure existing properties (Ettore Merlo, Dominic Letarte, Giuliano Antoniol, 2007).

2.5 From "Preventing SQL Injection Attacks in Stored Procedures", they also provided a Novel approach to shield the stored procedures from attack and detect SQL injection from sit (Ke Wei, M. Muthuprasanna, Suraj Kothari, 2007). This method combines runtime check with static application code analysis so that they can eliminate vulnerability to attack. The key behind this attack is that it alters the structure of the original SQL statement and identifies the SQL injection attack. The method is divided in two phases, one is offline and another one is runtime. In the offline phase, stored procedures use a parser to pre- process and detect SQL statements in the execution call for runtime analysis. In the runtime phase, the technique controlled all runtime generated SQL Queries related with the user input and check these with the original structure of the SQL statement after getting input from the user. Once this technique detects the malicious SQL statements it prevents the access of these statements to the database and provides details about attack.

3 Methodology

3.1 Tautologies:

Tautology-based attack is to inject code in one or more conditional statements so that they always evaluate to true. The most common usages of this technique are to bypass authentication pages and extract data. If the attack is successful when the code

either displays all of the returned records or performs some action if at least one record is returned. Example: In this example attack, an attacker submits

Enter haroon as username

Enter xxx') OR 1 = 1 --] as password

3.2 Union Query:

In union-query attacks, Attackers do this by injecting a statement of the form: UNION SELECT <rest of injected query> because the attackers completely control the second/injected query they can use that query to retrieve information from a specified table. The result of this attack is that the database returns a dataset that is the union of the results of the original first query and the results of the injected second query.

Example: Inject this Query

- 1' order by 7-- -
- 1' union select 1,2,3,4,5,6,7-- -
- 1'union select 1,@@version,3,4,5,6,7-- -
- 1' union select 1,table_name,3,4,5,6,7 from information_schema.tables-- -
- 1'union select 1,column_name,3,4,5,6,7 from information_schema.columns where table_name="users"-- -
- 1' union select 1,column_name,3,4,5,6,7 from information_schema.columns where table_name=char(117,115,101,114,115)-- -
- 1' union select 1,password,3,4,5,6,7 from users-- -

3.4 Proposed Solution

A web application is vulnerable to an SQL injection attack if an attacker is able to insert SQL statements into an existing SQL query of the application. This is usually achieved by injecting malicious input into user fields that are used to compose the query, login page prompts the user to enter her username and password into a form typically used for checking the user login credentials therefore, are prime targets for an attacker. In this example, if the login application does not perform correct input validation of the form fields, the attacker can inject strings into the query that alter its semantics. For example, consider an hacker entering “OR 1=1”-- one of sql injection string as in the tabel(1), the “--” command indicates a comment in Transact-SQL. Hence, everything after the first “--” is ignored by the SQL database engine with the help of the first quote in the input string, the user name string is closed, while the “OR 1=1” adds a clause to the query which evaluates to true for every row in the table. When executing this query, the database returns all user rows, which applications often interpret as a valid login. So in the proposed system all client-supplied data needs to be cleansed of any characters or strings that could possibly be used maliciously as shown in the figure(1).

In this section list proposed methods can be applied to minimize the risk of a SQL injection attack

3.5 Using escape string

Developers usually use php function mysql_real_escape_string to filter input data. As of PHP 5.5.0 mysql_real_escape_string and the mysql extension are deprecated. So we shall use mysqli extension and mysqli::escape_string function instead. Where the unnecessary extra characters are removed and then passed to the execution.

3.6 Using prepared statements

When prepared statements and parameterized queries are used, these SQL statements that are sent to and parsed by the database server separately from any parameters. In this way it is impossible for an attacker to inject malicious SQL.

3.7 Validation Input

The first step in any form processing script should check the syntax of input for validity to verify that the input is a valid input in the language. The validation could be anything such as checking whether the entered value for “password" is a number, and is 16 or over, or making sure the username column has only valid characters such as A to Z, a to z and many classes of input have fixed languages such Email addresses, dates etc.

3.8 Limit the Length of User Input

Make use of the maxlength attribute of the textbox controls will be restricting the number of characters the hacker can type. In proposed system all text boxes and form fields should be always as short as possible for example, if the firstname textbox should only accept 40 characters, then enforce the length in the maxLength attribute. This will restrict the hacker to send small set of commands back to the server

<input type="text" id="txtFirstname" MaxLength="40" runat="server "/>

3.9 Error messages

These should not reveal sensitive information and where exactly an error occurred. Simple custom error messages such as “Sorry, we are experiencing technical errors. The technical team has been contacted. Please try again later” can be used instead of display the SQL statements that caused the error.

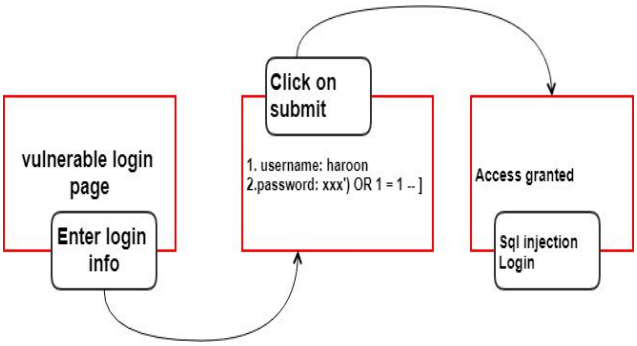
3.10 Password encryption

SQL Server automatically encrypts the passwords that you assign to logins and application roles. Even if you look directly into the system tables in the master database, you won't find actual passwords. You don't need to do anything to enable this feature; in fact, you can't disable it.

3.11 Result and Discussion

Suppose user supplies admin and 1234 as the password. The statement to be executed against the database would be

SELECT * FROM users WHERE email = 'admin' AND password = md5('1234');



The above code can be exploited by commenting out the password part and appending a condition that will always be true. Let's suppose an attacker provides the following input in the email address field.

admin' OR 1 = 1 LIMIT 1 -- ']

Xxx for the password.

The generated dynamic statement will be as follows.

SELECT * FROM users WHERE email = 'admin' OR 1 = 1 LIMIT 1 -- '] AND password = md5('1234');

Here,
xxx@xxx.xxx ends with a single quote which completes the string quote

OR 1 = 1 LIMIT 1 is a condition that will always be true and limits the returned results to only one record.

-- ' AND … is a SQL comment that eliminates the password part.

We have a login page that is vulnerable to SQL Injection attacks. The php form code above is taken from the login page. The application provides basic security such as sanitizing the username field. This means our above code cannot be used to bypass the login.

To get round that, we can instead exploit the password field. The diagram below shows the steps that you must follow
Let's suppose an attacker provides the following input

Step 1: Enter haroon as username

Step 2: Enter xxx') OR 1 = 1 --] S

tep 3: click on sign in button

Step 4: you will directed to the welcome page

4. Conclusion

Web applications have become an essential and integral part of internet usage today as they provide the convenience of business and personal transactions anywhere anytime. However, SQLIAs pose a serious threat to web applications hence; the primary purpose of this project was to present a new model to protect and detect web applications against SQLIAs with least modification of the Web architecture. Our proposed model was developed based on negative tainting and SQL syntax-aware methods, and was evaluated through SQL penetration testing in web application and database server. The negative tainting and SQL syntax-aware that we use gives our technique several significant advantages over techniques based on other mechanisms.
Evaluations have been performed using three different applications. We were able to successfully distinguish between legitimate SQL queries and malicious ones that had adopted various evasion methods such as encoding, comments and white space evasion methods as well as logical expressions and string techniques that were not captured by commercially available detection engines.

REFERENCES

[1] Atefeh Tajpour ,Suhaimi Ibrahim, Mohammad Sharifi"Web Application Security by SQL Injection DetectionTools", IJCSI International Journal of ComputerScience Issues, Vol. 9, Issue 2, No 3, March 2012.
[2] Mayank Namdev, Fehreen Hasan, Gaurav Shrivastav"Review of SQL Injection Attack and Proposed Methodfor Detection and Prevention of SQLIA", InternationalJournal of Advanced Research in Computer Scienceand Software Engineering, Volume 2, Issue 7, July2012.
[3] Sitharthan. S, Sankaran. S "SQL Injection Attack-Types and Classification A Review", InternationalJournal of Computer Science and InformationTechnology Research ISSN 2348-120X (online) Vol. 2,Issue 3, pp: (33-38), Month: July September 2014.

[4] Srinadh Swamy, Pavan Kumar & Vas U Dev,"Improved Authentication Technique To Protect WebApplications", International Journal of Computer Science and Engineering (IJCSE) ISSN(P): 2278-9960;ISSN(E): 2278-9979 Vol. 3, Issue 3, May 2014.

[5] Atefeh Tajpour,Maslin Masrom, Mohammad JojorZadeh Shooshtari, Hossein Rouhanizeidanloo,"Comparison of SQL Injection Detection andPrevention Tools based on Attack Type andDeployment Requirements", International Journal of Advancements in Computing Technology Volume 3,Number 7, August 2011.

[6] Kamlesh Kumar Raghuvanshi, DeenBandhu Dixit,"Prevention and Detection Techniques for SQLInjection Attacks", International Journal of Computer Trends and Technology (IJCTT) – volume 12 number 3– Jun 2014.

[7] William G.J. Halfond, Jeremy Viegas, and AlessandroOrso, "A Classification of SQL Injection Attacks andCountermeasures", IEEE, 2006.

[8] Hossain Shahriar , Sarah North , and Wei-Chuen Chen,"Early Detection of SQL Injection Attacks",International Journal of Network Security & ItsApplications (IJNSA), Vol.5, No.4, July 2013.

[9] Atefeh Tajpour, Suhaimi Ibrahim, Maslin Masrom,"SQL Injection Detection and Prevention Techniques",al Journal of Innovative Research in Science,Engineering and Technology (An ISO 3297: 2007Certified Organization) Vol. 3, Issue 4, April 2014.

[10] Jignesh Doshi, Bhushan Trivedi, "Assessment of SQLInjection Solution Approaches", International Journalof Advanced Research in Computer Science andSoftware Engineering, Volume 4, Issue 10, October2014.