ISSN:0975-9646

Attacks Due to SQL Injection & Their Prevention Method for Web-Application

Shubham Srivastava[1], Rajeev Ranjan Kumar Tripathi[2]
[1]*Department of Computer science and Engineering ,TMU, Moradabad (India)*
[2]*Department of Computer science and Engineering,ITM, GIDA, Gorakhpur (India)*

***Abstract*-The use of web application has become increasingly popular in our daily life as reading news paper, making online payments for shopping etc. At the same time there is an increase in number of attacks that target them. In this paper we present a detailed review on various types of SQL injection attacks and prevention technique for web application. Here we are presenting a new technique for prevention of SQL injection attack for web application. As we know that SQL injection attack can be easily prevented by applying more secure scheme in login phase. To address this problem we proposed a technique with highly secure login scheme which uses hash code with salt.**

***Keywords*-**

SQL injection, database security, authentication, Hash- code , Cryptographic salt and Final hash code.

1. INTRODUCTION

The use of web application has become increasingly popular in our daily life as reading news paper, making online payments for shopping etc. At the same time there is an increase in number of attacks that target them.SQL injection attacks are possible because web application code is not secured during application development. SQL injection is a hacking method that is based on the security vulnerabilities of web application.

It is categorized as one of the top-10 2010 Web application vulnerabilities experienced by Web applications according to OWASP (Open Web Application Security Project) [9].
One of the best ways to secure applications is by limiting access to those authorized to access the application[14]. In this paper we proposed a new technique to authenticate user for limiting access. In this technique we calculate a **Final hash code value** at run time and correct matching will authenticate the user. This technique uses hash function and salt, a **salt** consists of random bits creating one of the inputs to a one way function and a hash function is a subroutine that maps a large data set to a small data set.

1.1 Meaning & impact of SQL Injection

There are some malicious code that can be attach to the SQL called SQL Injection. SQL Injection is one of the many web attack mechanisms used by hackers to steal data from organizations. It is perhaps one of the most common application layer attack techniques used today. It is the type of attack that takes advantage of improper coding of our web applications that allows hacker to inject SQL commands into say a login form to allow them to gain access to the data held within our database.

In essence, SQL Injection arises because the fields available for user input allow SQL statements to pass through and query the database directly.SQL Injection is the hacking technique which attempts to pass SQL commands (statements) through a web application for execution by the backend database. If not sanitized properly, web applications may result in SQL Injection attacks that allow hackers to view information from the database. Once an attacker realize that a system is vulnerable to SQL injection, he is able to inject SQL query or commands through an input form field.
An attacker may execute arbitrary SQL statements on the vulnerable system. This may compromise the integrity of our database and/or expose sensitive information. Depending on the back-end database in use, SQL injection vulnerabilities lead to varying levels of data/system access for the attacker.

2. RELATED WORK

Most of existing techniques, such as filtering, information-flow analysis, penetration testing, and defensive coding, can detect and prevent a subset of the vulnerabilities that lead to SQLIAs. In this section, we list the most relevant techniques-

William G.J.Halfond et al.'s Scheme- [2]- This approach works by combining static analysis and runtime monitoring. In its static part, technique uses program analysis to automatically build a model of the legitimate queries that could be generated by the application. In its dynamic part, technique monitors the dynamically generated queries at runtime and checks them for compliance with the statically-generated model. Queries that violate the model represent potential SQLIAs and are thus pre- vented from executing on the database and reported.

SAFELI – [3] proposes a Static Analysis Framework in order to detect SQL Injection Vulnerabilities. SAFELI framework aims at identifying the SQL Injection attacks during the compile-time. This static analysis tool has two main advantages. Firstly, it does a White-box Static Analysis and secondly, it uses a Hybrid-Constraint Solver. For the White-box Static Analysis, the proposed approach considers the byte-code and deals mainly with strings. For the Hybrid-Constraint Solver, the method implements an efficient string analysis tool which is able to deal with Boolean, integer and string variables.

Thomas et al.'s Scheme - Thomas et al., in [11] suggest an automated prepared statement generation algorithm to remove SQL Injection Vulnerabilities. They implement their research work using four open source projects namely: (i) Net-trust, (ii) ITrust, (iii) WebGoat, and (iv) Roller. Based on the experimental results, their prepared statement code was able to successfully replace 94% of the SQLIVs in four open source projects.

Ruse et al.'s Approach - In [12], Ruse et al. propose a technique that uses automatic test case generation to detect SQL Injection Vulnerabilities. The main idea behind this framework is based on creating a specific model that deals with SQL queries automatically. Adding to that, the approach identifies the relationship (dependency) between sub-queries. Based on the results, the methodology is shown to be able to specifically identify the causal set and obtain 85% and 69% reduction respectively while experimenting on few sample examples.

Ali et al.'s Scheme - [14] adopts the hash value approach to further improve the user authentication mechanism. They use the user name and password hash values SQLIPA (SQL Injection Protector for Authentication) prototype was developed in order to test the framework. The user name and password hash values are created and calculated at runtime for the first time the particular user account is created

Roichman and Gudes's Scheme – [13]suggests using a fine-grained access control to web databases. The authors develop a new method based on fine-grained access control mechanism. The access to the database is supervised and monitored by the built-in database access control. This is a solution to the vulnerability of the SQL session traceability. Moreover, it is a framework applicable to almost all database applications.

SQL-IDS Approach - Kemalis and Tzouramanis in [16] suggest using a novel specification-based methodology for the detection ofexploitations of SQLinjection vulnerabilities. The proposed query-specific detection allowed the system to perform focused analysis at negligible computational overhead without producing false positives or false negatives.

SQLrand Scheme - SQLrand approach [18] is proposed by Boyd and Keromytis. For the implementation, they use a proof of concept proxy server in between the Web server (client) and SQL server; they de-randomized queries received from the client and sent the request to the server. This de-randomization framework has 2 main advantages: portability and security. The proposed scheme has a good performance: 6.5 ms is the maximum latency overhead imposed on every query.

SQLIA Prevention Using Stored Procedures - Stored procedures are subroutines in the database which the applications can make call to[15] . The prevention in these stored procedures is implemented by a combination of static analysis and runtime analysis. The static analysis used for commands identification is achieved through stored procedure parser and the runtime analysis by using a SQL Checker for input identification.

Parse Tree Validation Approach - Buehrer et al. [20] adopt the parse tree framework. They compared the parse tree of a particular statement at runtime and its original statement. They stopped the execution of statement unless there is a match. This method was tested on a student Web application using SQLGuard. Although this approach is efficient, it has two major drawbacks: additional overheard computation and listing of input (black or white).

Dynamic Candidate Evaluations Approach - In [10], Bisht et al. propose CANDID. It is a Dynamic Candidate Evaluations method for automatic prevention of SQL Injection attacks. This framework dynamically extracts the query structures from every SQL query location which are intended by the developer (programmer). Hence, it solves the issue of manually modifying the application to create the prepared statements.

3. PROPOSED TECHNIQUE

In this paper we proposed a new technique for preventing Database against SQL injection attack. In this approach one extra column is required in user account table to store **Final hash value**. This value is created at the time of new user registration and stored in user account table together with user name and password as shown in table.

Table1- User account table

USER NAME	PASSWORD	FINAL HASH-CODE

At the time of login Final hash-code is calculated using stored procedure at run time and authentic user is identified by exact matching of username, password and final hash-code. For calculation of Final hash code we will proceed according to architecture given in next section.

3.1 ARCHITECTURE

Architecture of proposed technique is shown in figure below-

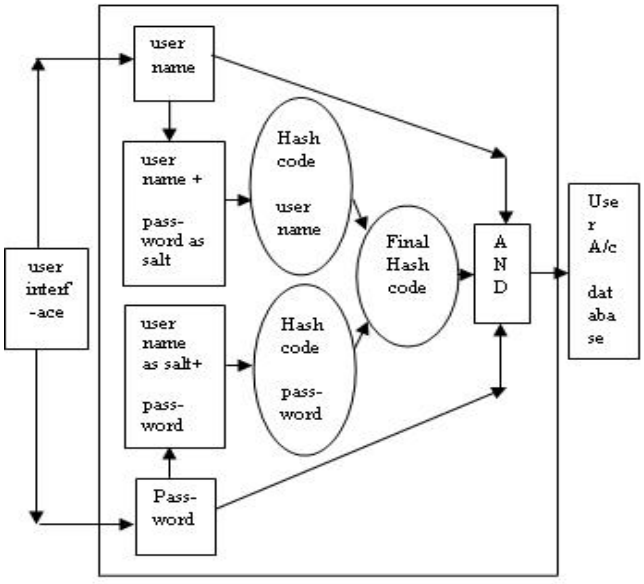


fig.1 -Proposed Architecture

3.2 WORKING METHODOLOGY

The working of proposed technology can be divided into two parts-

1-Registration of new user:- when a new user wants to register he/she will fill the login form with a unique name and password at user interface. At middle tier this unique name and password is processed according to the proposed architecture.

steps are as given below-

1. Find hash code of login name using password as salt.
2. Find hashcode of password using login name as salt.
3. Find **Final hashcode** by concatenating output of step1 and step 2
4. Store login name,Password and Finalhashcode (output of step3) to user a/c table.

2- Login and verification:- when a user wants to logon he/she will fill the login form .

steps are as given below-

1. Enter a unique name and password at user interface.
2. Entered user name is matched with the name stored in user a/c table.
3. when user name matches correctly , process user name and password according to proposed technique to find Final hash code at run time.
4. verify this final hashcode and password with stored values.
5. If he/she is valid user then can access database to retrieve information from there otherwise error message is displayed .

Working can be diagramatically shown as below-

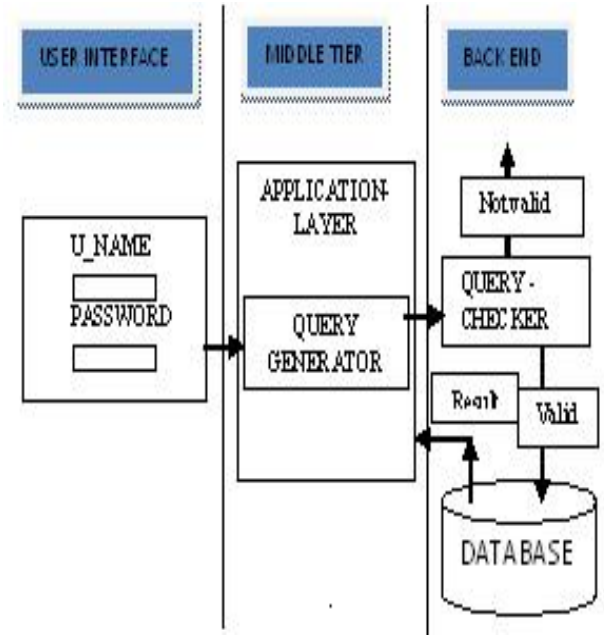


fig.2 –Three tier proposed Architecture

4. EVALUATION OF TECHNIQUE

The performance of proposed technique has been evaluated on a table having different number of user records. We computed response time of the system with and without embedded our proposed technique.

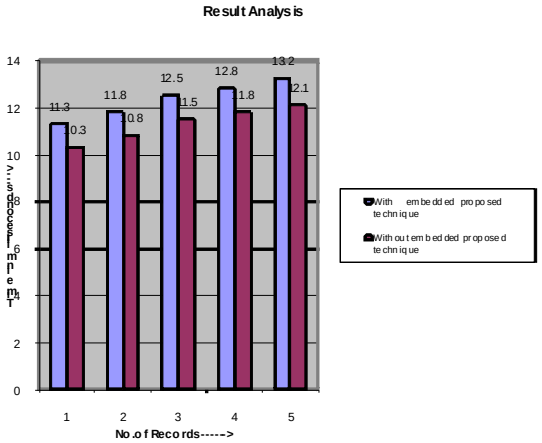
For evaluation of our proposed technique we consider a dummy data table with varying number of records 10, 20, 30, 40, 50. Result demonstrate that our proposed method put insignificant overhead onto the server in terms of time required in milliseconds.

Processing overhead for different number of users is shown into the table given below:-

Total Records	With embedded proposed technique	Without embedded proposed technique
10	11.3	10.3
20	11.8	10.8
30	12.5	11.5
40	12.8	11.8
50	13.2	12.1

Table2 – Performance analysis of proposed technique

Following graph shows the comparative study of system with and without embedded proposed technique-



5.CONCLUSIONS AND FUTURE WORK

It is obvious from above description that SQL injection attacks are one of the largest classes of security problems. Most existing technique either require developers to manually specify the interfaces to an application or, if automated, are often inadequate when applied to modern, complex web applications.

In this paper we have reviewed the most popular existing SQL Injections related issues. We proposed a new technique based on hash function which is simple and highly secure from attackers. This paper presents an authentication method for preventing SQL injection attack and limiting access to those authorized to access the application. The proposed scheme is efficient and overhead is negligible. The future evaluation work should focus on efficiency of the system.

5. ACKNOWLEDGMENT

We are feeling so glad to express our sincere thanks to our Parents, supervisor , colleagues and all those persons who helped us directly and indirectly to complete this paper.

6. REFERENCES

[1] Indrani Balasundaram, E.Ramaraj "An Authentication Scheme for Preventing SQL Injection Attack Using Hybrid Encryption PSQLIA-HBE"(ISSN 1450-216X Vol.53 No.3 (2011),pp.359-368)

[2] William G.J.Halfond and Alessandro Orso "AMNESIA:Analysis and Monitoring for Neutralizing SQL-Injection Attacks"

[3] X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian, and L. Tao. A StaticAnalysis Framework for Detecting SQL Injection Vulnerabilities,COMPSAC 2007, pp.87-96, 24-27 July 2007

[4] William G.J.Halfond ,JeremyViegas, Alessandro Orso "AClassification of SQL injection Attacks And Countermeasures"

[5] Romil Rawat , Chandrapal Singh Dangi,Jagdish Patil " Safe Guard Anomalies against SQL Injection Attacks"

[6] Indrani Balasundaram, Dr.E.Ramaraj "An Approach to Detection of SQL Injection Attacks in Database Using Web Services"(IJCSNS ,VOL. 11 No.1,January 2011

[7] Debasish Das,Utpal Sharma & D.K. Bhattacharyya "An Approach to Detect and Prevent SQL Injection Attack Based on Dynamic Query Matching"

[8] A. Tajpour; M. Masrom; M. Z. Heydari.; S. Ibrahim; "SQLinjection detection and prevention tools assessment," Proc. Of ICCSIT 2010, vol.9, no., pp.518-522, 9-11 July 2010

[9] [http://www.owasp.org/index.php/Top_10_2010-A1-Injection, retrieve on 13/01/2010

[10] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan. CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks. ACM Trans. Inf. Syst. Secur., 13(2):1–39, 2010

[11] S. Thomas, L. Williams, and T. Xie, On automated prepared statement generation to remove SQL injection vulnerabilities. Information and Software Technology 51, 589–598 (2009).

[12] M. Ruse, T. Sarkar and S. Basu. Analysis & Detection of SQL Injection Vulnerabilities via Automatic Test Case Generation of Programs. 10th Annual International Symposium on Applications and the Internet pp. 31 – 37 (2010)

[13] Roichman, A., Gudes, E.: Fine-grained Access Control to WebDatabases. In: Proc. of 12th SACMAT Symposium, France (2007)

[14] Shaukat Ali, Azhar Rauf, Huma Javed "SQLIPA:An authentication mechanism Against SQL Injection"

[15] K. Amirtahmasebi, S. R. Jalalinia, S. Khadem, "A survey of SQLinjection defense mechanisms," Proc. Of ICITST 2009, vol., no., pp.1-8, 9-12 Nov. 2009

[16] K. Kemalis, and T. Tzouramanis (2008). SQL-IDS: A Specification-based Approach for SQLinjection Detection. SAC'08. Fortaleza, Ceará, Brazil, ACM: pp. 2153 2158.

[17] Shubham srivastava,"A Survey On: Attacks due to SQL injection and their prevention method for web application" (IJCSIT) Vol. 3 (1) , 2012, 3225-3228

[18] S. W. Boyd and A. D. Keromytis. SQLrand: Preventing SQL Injection Attacks. In Proceedings of the 2nd Applied Cryptography and Network Security Conference, pages 292–302, June 2004.

[19] Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee and S.-Y. Kuo , "Securing Web Application Code by Static Analysis and Runtime Protection," 13th International Conference on World Wide Web, New York, NY, 2004, pp. 40-52.

[20] G. Buehrer, B.W. Weide, P.A.G. Sivilotti, Using Parse Tree Validation to Prevent SQL Injection Attacks, in: 5th International Workshop on Software Engineering and Middleware, Lisbon, Portugal, 2005, pp. 106–113.