ISSN No. 0976-5697

Volume 2, No. 5, Sept-Oct 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Key Generator based secured system against SQL-Injection attack

Romil Rawat
M.Tech scholar,
Department Of Information Technology
SATI Vidisha M.P
rawat.romil@gmail.com

Sumit Dhariwal*
M.Tech scholar,
Department Of Information Technology
SATI Vidisha M.P
sumitdhariwal22@rediffmail.com

Nikhil Patearia
M.Tech Scholar,
Department of Computer Science & Engineering,
SATI Vidisha M.P, India
nikhil_sati29@rediffmail.com

Abstract: As the internet has been grown, user has been tremendously addicted to use web application. Almost every sectors like banking, Reservation, traveling and many sectors has been completely dependent on internet functionalities, SQL-Injection has been ranked at the top most attack in web application. It breaks private and confidential secured systems by applying different vulnerable tricks used by attacker. It uses bypassing techniques to access the restricted information in unauthorized manner. So, for protecting them various security has been implemented, but as security grown, attacks also grown, many systems has already crashed and stolen for maliciously purposes. Here we have used a technique, which uses key-generator and comparator function. Key is generated by permutation and combination of username and password values. It is compared by the saved value of key, if it matches, there is no SQL-Injection, otherwise there is Injection.

Keywords: SQL Injection, Key Generator, Comparator.

I. INTRODUCTION

SQL-Injection Most threatening attack has been found on web application, which bypasses the direction of execution without any valid authentication[1,2] .The Structural Query Language Injection (SQLI) attack occurs when an attacker changes the logic, semantics or syntax of a SQL query by inserting new SQL keywords or operators. SQL databases are attractive targets [4, 5, 7] because they often contain valuable information such as user names, passwords, e-mail addresses, personal data, and financial records. SQL Injection Attack is a class of code injection attacks that happens when there is no input validation. In fact, attackers can shape their illegitimate input[3,4,15] as parts of final query sting which operate by databases. Banking web applications or secret information systems could be the victims of this vulnerability because attackers by abusing this vulnerability can threat their authority, integrity and confidentiality [4,516,17]. So, developers addressed some defensive and secured coding practices to eliminate this vulnerability but they are not sufficient. SQLIAs can also escape traditional tools such as firewalls and Intrusion Detection Systems because they performed through ports used for regular web traffic. SQL injection attacks can be carried out easily using only a web browser going through port 80 which is frequently left open by firewalls.

Attackers have the opportunity to put additional SQL instructions into places where the programmer expected only benign data. With carefully structured inputs, an attacker could remotely execute arbitrary instructions in a database. With some trial and error probing, attackers can learn about the database and gain complete control by causing malicious string inputs to be executed by the database.[7,8,18,19] Because there is no strict separation

between program instructions and user data, it is possible that user data could be interpreted as instructions. Successful attacks can lead to the extraction, modification, addition, or deletion of sensitive data. SQL injection vulnerabilities occur when a web application does not properly validate [9, 10, 20] user input (for instance, fields in a web form) and then includes that input as part of a SQL statement.

In research contributes the methodology generating a key from username and password , by the permutation and combinations of username and password, and before input query is embed ,key is generated by user input and compared by saved Key ,if found equal, query is forwarded with positive access. If found unequal it is diverted to service page which generates a error report. Here service page converts the Error report to a form, which is unusable for attacker to retrieve the information form the appeared error page.

II. RELATED WORK

McClure and Kruger [8] in their work used a completely different query development platform by changing the so-called unregulated query generation process that uses string concatenation, to a new systematic one[21]. This solution consists of two parts. The first is an abstract object model. The second is an executable which is executed against a database and the output is a Dynamic Link Library (DLL) containing classes that are strongly-typed to the database schema. This DLL is used as a concrete instantiation of the abstract object model. However, as they provide a completely new paradigm for query development process which is not as easy as previous one, the developer need to learn before its use. Swaddler [3], analyzes the internal state of a web application. It works based on both single and multiple variables and shows an impressive way against complex attacks to web applications. First the approach describes the normal values for the application's state variables in critical points of the application's components. Then, during the detection phase, it monitors the application's execution to identify abnormal states[21]. SQL rand [9] is a of Instruction-Set Randomization. The SQL standard keywords are manipulated by appending a random integer to them. The attacker is not aware about that random integer. Thus if any malicious user attempting to SQL injection attack would immediately be thwarted as the injective codes in the randomized query are treated as non- keywords.

IDS [6] use an Intrusion Detection System (IDS) to detect SQLIAs, based on a machine learning technique. The technique builds models of the typical queries and then at runtime, queries that do not match the model would be identified as attack. This tool detects attacks successfully but it depends on training seriously. Else, many false positives and false negatives would be generated.

In [10] Valeur et al. gave a learning-based Intrusion Detection System (IDS) to detect SQLIA. The IDS is trained using a set of typical application queries. The technique builds statistical models of the typical queries and then monitors the application at run-time to identify the queries that do not match the model. However, the fundamental limitation is that the success of such system completely depends on the quality of the training set used. Poor training set would result large number of false positive and false negative.

Scott and Sharp, [11] propose a solution to provide an application level security including SQLIA for web-based applications. They use a security policy description language (SPDL) to specify a set of validation constraints and transformation rules to be applied to application parameters as they flow from the web page to the application server. The compiled SPDL codes are kept on a security gateway which acts as application level firewall.

However, the developers are completely responsible for this. They have to know not only which data needs to be filtered, but also what patterns and filters need to apply to data. Many testing techniques [12], [13], [14] have been proposed to test whether the web applications are vulnerable to SQLIA. In [12] the proposed technique is based on black-box approach, whereas, [13], [14] describe white-box approach.

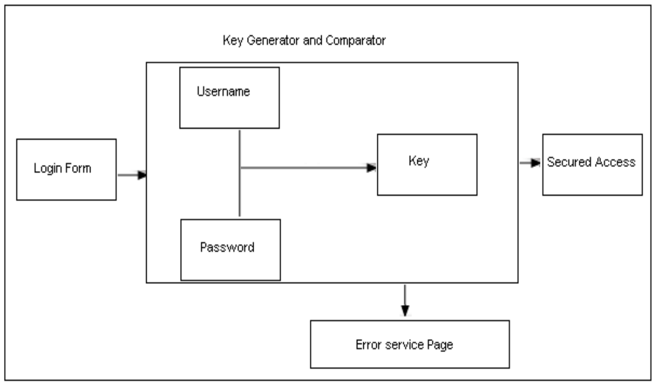
Another approach in this category is SQL-IDS [8] which focus on writing specifications for the web application that describe the intended structure of SQL statements that are produced by the application, and in automatically monitoring the execution of these SQL statements for violations with respect to these specifications.

III. PROPOSED TECHNIQUE

This paper proposes a new technique, for preventing database against SQL Injection. In the given approach there is need of one extra column in U_account for saving the key, which is generated at the key generator phase. When the same user Logins, the key is again created by key generator (permutations and combinations) and compared by the saved keys of corresponding User, if both the keys matches, the user input values are diverted towards dynamic query generator for finally execution. But if the keys doesn't matches the link is diverted towards the Error page, which reflects error page for attacker by hiding information and unusable for hacker**.**

IV. ARCHITECTURE

Architecture of proposed technique consists of 4 components .These are Login form, Key generator and comparator, service page and secured process. Shown in fig 1. And fig 2. Shows a key generator.



Key Generator Based Secure System

Figure.1 Key generator based Secure System

Key Generator

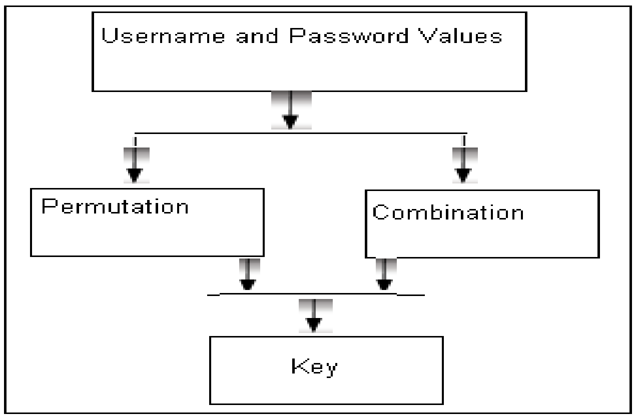


Figure.2 Key Generator

Here, the login form is just the user entry form for username and password. Secured process is the authentic actions happens after validating the user info. Key generator and comparator is the main component which generates and compares the saved and generated key. service page is activated when the both keys does not matches and ,generates a error page.. Some of these error messages carry some useful information about the Meta data. So, the attacker flouted in the SQL statement with logical error through which the attacker can able to know the Meta data information. If the attacker is able to get the Meta data information, then the attacker can insert/ delete/modify the table information through SQL Injection. To avoid such attack, a web service module is implemented. This module get the error message from the database server, and makes it as a generalized error message and send back to the application server. So that the attacker wont get any valuable information from the error message.

V. TECHNIQUE

A key generator and comparator has been created for comparing the authenticity of queries .here two stored procedures are used which checks the validity of queries if they matches, means no sql-injection,the query is not effected,Otherwise there is injection attack.Our system is implemented using Microsoft Sql-Server as a DBMS.When a user wants to login checking procedures starts

VI. RESULTS

Our concept has been checked on a number of recoreds.Dummy records of tables has been used here ,table consists of no. of records 150,300,450,600,750,900 was tested and whose results are shown in fig[3].

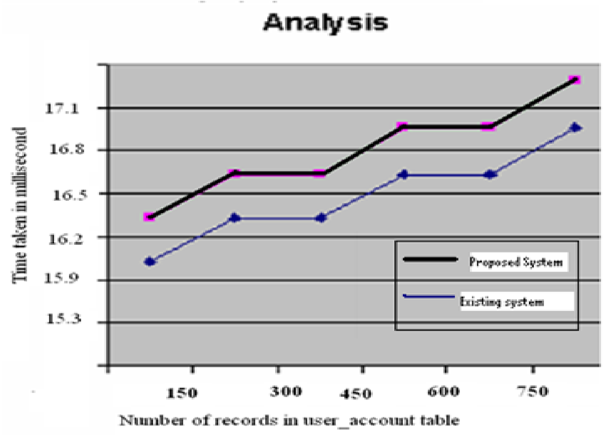


Figure.3 Analysis

The Graph looks linear as there is very minute difference of 1.5 ms and it doesnot puts any overhead on the existing system.

VI. CONCLUSION

The proposed approach presents a new technique Key generator based secured system to secure the authentication process of the database. It uses username and password to generate the key by permutations and combinations, and the key is saved to check the login process of the respected users .and the error page provides the safe technique to generate the report. And if key is match secured access performs the action.

VII. REFERENCES

[1] R. Ezumalai and G. Aghila. Combinatorial Approach for Preventing SQL Injection Attacks. IACC, 2009.

[2] MeiJunjin. An approach for SQL Injection vulnerability detection. IEEE,2009.

[3] Marco Cova, Davide Balzarotti. Swaddler: An Approach for the Anomaly-based Detection of State Violations in Web Applications. In Proceedings of the 10th Interational Symposium on Recent Advances in Intrusion Detection (RAID), (Queensland, Australia), September 5-7, 2007, pp. 63-86.

[4] D. Stuttard and M. Pinto, “The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws”, Wiley Publishing Inc., 2007

[5] Symantec. “Symantec Report on the Underground Economy”, 2008.

[6] F. Valeur, D. Mutz, and G. Vigna. A Leaing-Based Approach to the Detection of SQL Attacks. In Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), Vienna, Austria, July 2005.

[7] M. Howard and D. LeBlanc, “Writing secure code”, Microsoft Press, 2003

[8] R. McClure and I. Kruger, “Sql dom: Compile time checking of dynamic sql statements,” in Proceedings of the 27th International Conference on Software Engineering (ICSE 05), St. Louis, Missouri, USA, May 2005.

[9] S. W. Boyd and A. D. Keromytis, “Sqlrand: Preventing sql injection attacks,” in Proceedings of the 2nd Applied Cryptography and Network Security (ACNS) Conference, June 2004, pp. 292–302.

[10] F. Valeur, D. Mutz, and G. Vigna, “A learning-based approach to the detection of sql attacks,” in Proceedings of the Conference on Detection of Intrusions and Malware

[11] and Vulnerability Assessment (DIMVA), Vienna, Austria, July 2005.

[12] D. Scott and R. Sharp, “Abstracting application-level web security,” in Proceedings of the 11th International Conference on the World Wide Web (WWW 2002), 2002, pp. 396–407.

[13] Y. Huang, S. Huang, T. Lin, and C. Tsai, “Web applica-tion security assessment by fault injection and behavior monitoring,” in Proceedings of the 11th International World Wide Web Conference (WWW 03), May 2003.

[14] H. Shahriar and M. Zulkernine, “Music: Mutation-based sql injection vulnerability checking,” in The Eighth Inter-national Conference on Quality Software, 2008, pp. 77–86.

[15] Y. Kosuga, K. Kernel, M. Hanaoka, M. Hishiyama, and Y. Takahama, "Sania: Syntactic and semantic analysis for automated testing against sql injection," in Twenty- Third Annual Computer Security Applications Conference, 2007, pp. 107–117.

[16] Konstantinos Kemalis and Theodoros Tzouramanis. SQL-IDS: A Specifcation-based Approach for SQL Injection Detection Symposium on Applied Computing. 2008, pp: 2153-2158 , Fortaleza, Ceara, Brazil. New York, NY, USA: ACM.

[17] A. S. Christensen, A. M0ller, and M. I. Schwartzbach. Precise Analysis of String Expressions. In Proc. 10th International Static Analysis Symposium, SAS '03, volume 2694 of LNCS, pp 1-18. Springer-Verlag, June 2003.

[18] G. T. Buehrer, B. W. Weide, and P. A. G. Sivilotti. Using Parse Tree Validation to Prevent SQL Injection Attacks. In International Workshop on Software Engineering and Middleware (SEM), 2005.

[19] P.Grazie., PhD SQLPrevent thesis. University of British Columbia (UBC) Vancouver, Canada.2008.

[20] C. Gould, Z. Su, and P. Devanbu. JDBC Checker:A Static Analysis Tool for SQL/JDBC Applications. In Proceedings of the 26th International Conference on Software Engineering (ICSE 04) Formal Demos, pp 697–698, 2004.

[21] C. Gould, Z. Su, and P. Devanbu. Static Checking of Dynamically Generated Queries in Database Applications. In Proceedings of the 26th International Conference on Software Engineering (ICSE 04).

[22] Shaukat Ali,Azhar Rauf and Huma Javed, An Authentication Mechanism Against SQL Injection.In European Journal of Scientific Research,pp 604-611,2009

Short Biodata of Authors:

Nikhil Patearia presently pursuing M.Tech in the department of Computer Science & Engineering at Samrat Ashok Technological Institute, Vidisha, M.P., India. The degree of B.E. secured in Computer Science & Engineering at Truba Institute of Engineering & Information Technology Bhopal, in 2009. Research Interest includes Network Security, Data Mining and Artificial Intelligence.
Mobile: +91-8989443679, E-mail: nikhil_sati29@rediffmail.com

Mr.Romil Rawat presently pursuing M.Tech in the department of Information technology & Science at Samrat Ashok Technological Institute, Vidisha, M.P., India. The degree of Research Interest includes Network Security, Data Mining and Artificial Intelligence.
Mobile: +91-9907708093, E-mail: rawat.romil@gmail.com

Mr.Sumit Dhariwal presently pursuing M.Tech in the department of Information technology & Science at Samrat Ashok Technological Institute, Vidisha, M.P., India. The degree of Research Interest includes Network Security, Data Mining and Artificial Intelligence.
Mobile: +91-9752070470, E-mail: sumitdhariwal22@rediffmail.com