Asian Journal of Information Technology 16 (6): 451-457, 2017
ISSN: 1682-3915

A Flexible and Extendable Data Mining Based Generic Framework for Preventing SQL Injection Attacks

[1]J. Pradeep Kumar, [2]A. Udaya Kumar and [3]T. Ravi
[1]Department of CSE, Aditya College of Engineering, Madanapalle, Andhra Pradesh, India
[2]Hindustan Institute of Technology and Science, Chennai, India
[3]Srinivasa Institute of Engineering and Technology, Chennai, India

Abstract: As the contemporary applications are database-driven, SQL Injection Attacks (SQLIAs) have been capable of causing potential risk to businesses across the globe. Most of the existing solutions focused on SQL and its structure at application level which is doomed to fail when stored procedures are targeted. In this study, we propose a framework for detecting SQLIAs at database level. We exploit kernel level functions and data mining techniques such as classification to have basis for detection of such attacks. The framework provides placeholders to have flexible mechanisms that help in using different approaches in future. Thus, the framework provides pluggable mechanisms, so as to support future techniques as well at database level. We implemented the functionality of the framework using PostgreSQL. The kernel functions of the RDBMS are exploited in order to have integrated functionality to detect SQLIAs. The empirical results revealed that the proposed framework is able to provide 99% probability of protecting applications from SQLIAs. The framework also achieve 100% true positives in detecting SQLIAs.

Key words: Database, data mining, SQL injection attack, classification, applications, pluggable mechanisms

INTRODUCTION

SQL Injection is a mechanism in which SQL commands are inserted into input fields of a web based application. Such commands are appended to the underlying SQL query being used when web form is submitted. This is an attack which is aimed at breaking security of a system and gain access to its sensitive data. An attacker can inject parts of commands into an SQL command of a web application either through input fields or cookies. SQL injection comes under code-injection attacks (Halfond and Orso, 2005a, b) with SQL injection attacks are made on Oracle database server without direct connection to the database by injecting SQL, exploiting buffer overflows and by redirecting using UTL_HTTP package when output is not returned. With direct connection to database, attacks can be made by injecting SQL into pre-defined and user-defined stored procedures, exploiting buffer overflow in user-defined and pre-defined procedures and printing output on attacker's session (Fayo, 2005). Since, user authentication or login is one of the main security measures, this process is exploited by attackers. Here, they try SQL injection attacks. Once authentication is successful, the malicious users can gain access to sensitive information which will be used for monetary gains (Chapela, 2005). For instance consider the following code snippet that demonstrates SQL injection attack to gain access to database of SQL server (Algorithm 1).

Algotithm 1; A common vulnerable login query:

SELECT*FROM USERS WHERE userid = 'anand' and password =' 12345'
The above query appears as follows with SQL Server syntax
var query = "SELECT*FROM USERS WHERE userid ="'+formuser + "' and password = "'+formpassword +"'"
Here formuser and formpassword are names of text fields that become request parameters when HTML form is submitted. The injection attack on the login web page is done by injecting some command through input field
The following are the means to make SQL injection attack
Formuser =' or 1 = 1 -- (observe that -- is the comment as per SQL Server)
Form password = anything
Thus, the final SQL query appears as follows
SELECT*FROM USERS WHERE userid = '' or 1 = 1 -- and password =' anything';
In similar fashion, the same query for MY SQL appears as follows
SELECT*FROM USERS WHERE userid = '' or 1 = 1 # and password =' anything'

As seen in the final SELECT query the text preceded by is considered a comment which has no effect in thequery. Thus, the potential AND logical command is eliminated from the original query. Moreover or 1 = 1 is appended to the predicate and userid is given nothing.

Corresponding Author: J. Pradeep Kumar, Department of CSE, Aditya College of Engineering, Madanapalle, Andhra Pradesh, India

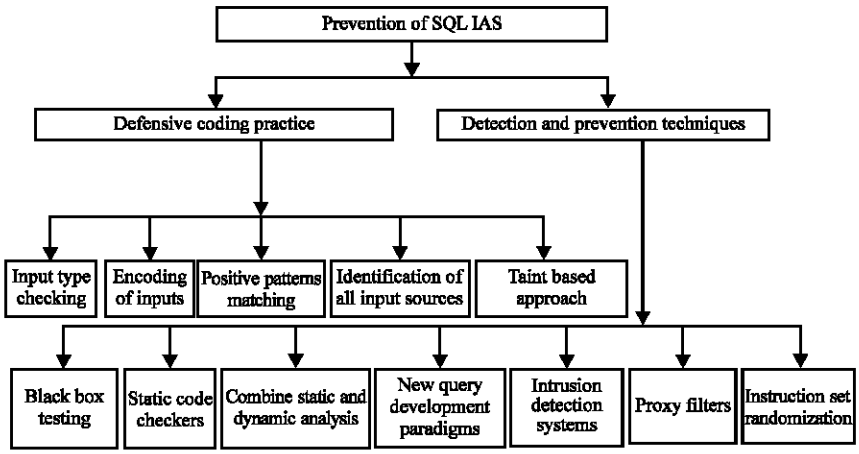

Fig. 1: Prevention measures for SQL injection attacks

The appended predicate or 1 = = 1 results in true always. For this reason, the SELECT query returns results that are internally used for authentication. The authentication is always succeeded (Chapela, 2005). Thus, SQL injection attacks successfully made in the process of authentication of web applications.

Halfond provides a survey of different types of SQL injection attacks and the counter measures available. Here, we mention the categories and counter measures. The types include tautologies that bypass authentication, illegal queries to find structure of database schema, union queries for by passing authentication, piggy-backed queries for manipulating data, stored procedures for privilege escalation, inference attacks to know database schema and alternate encodings to evade detection. The counter measures available include defensive coding practices and detection and prevention techniques. The summary of these counter measures is presented in Fig. 1.

As can be shown in Fig. 1, there are many approaches to prevent SQL injection. However, the focus in this study is to propose and implement a generic framework that can help in preventing SQL injection attacks using data mining techniques. Since, the database level protection is considered the right place for comprehensive solution to such attacks, we used data mining approach in this study.

The contributions in this study include the study into SQL injection attacks and counter measures besides the implementation of a generic framework that can be used to prevent SQL injection attacks with defence provided at database level.

Literature review: Web applications are generally vulnerable to SQL injection attack. This study reviews literature such attacks and prevention measures. Doug and Evans (2004) proposed an automated mechanism that could prevent two kinds of vulnerabilities such as cross-site scripting and SQL injection attacks. They replaced PHP interpreter with an extended one for precisely tracking taintedness and malicious content. This solution is nothing to do with the designer skills in developing a web site. Since, the modified PHP interpreter takes care of every web site that has been deployed into web server which used the extended interpreter. Halfond and Orso (2005a, b) presented a model based approach that could detect and neutralize SQL injection attacks. Its solution has both static and dynamic parts. In the static part it builds a model which will be used in the dynamic part at runtime to compare the static and dynamic modes to identify any suspicious injection of SQL commands. Their solution is based on an intuition that says that web applications have SQL queries that can be distinguished from malicious ones. Similar kind of research was carried out by Halfond and Orso (2005a, b). According to Litchfield (2005), there are three classes of SQL injection attacks. They are named as in band, out of band and inference attacks. When data is extracted from the same channel of client and server, it is known as in band. When different communication channel is employed, it is known as out of band attack while inference attack is the attack where no data transfer takes place but the attacker can infer the value of data.

Some developers obfuscate their code in order to prevent analysis of code to detect malicious content. Such content can be interpreted by using abstract interpretation based on the analysis proposed by Gerhard Goos. They combined the abstract stack graph and value set analysis to create an analyzer that was able to analyze obfuscated code containing malicious instructions. Stored procedures are actually vulnerable to SQL injection attacks. Wei *et al*. (2006) proposed a technique to prevent such attacks. Generally most of the solutions that prevent SQL injection attacks work at

application layer. However, the solution works at database layer where stored procedures reside. Their technique combines static analysis of application code and validation at run time to get rid of such attacks. The solution builds a graph containing SQL query behaviour and that can be used at runtime to validate and prevent SQL injection attacks. McClure and Kruger (2005) presented and employed SQLDOM as solution to SQL injection attacks. SQLDOM is a set of strongly typed classes that are generated automatically based on the database schema. The SQLDOM can help generate SQL commands in a web application instead of string manipulation. It solves all Call Level Interface (CLI) vulnerabilities.

Buehrer *et al.* (2005) proposed a solution based on the assumption that all SQL injection attacks alter the original query written by developer of interactive web applications in one way or other. Their solution is known as Parse Tree Validation (PTV). Parse tree is constructed before including the user inputs and after including user inputs. The comparison of the two provides evidence of any malicious code that has been inserted into genuine SQL command. A classification of SQL injection attacks and counter measures are found by William. Application layer intrusion detection (Rietta, 2006), SQL injection analysis in PHP (Merlo *et al.*, 2006, 2007), web vulnerability analysis (Kals *et al.*, 2006), SQL UnitGet for dynamic analysis (Shin *et al.*, 2006), SQL UnitGet for generating test cases (Shin *et al.*, 2006), static analysis framework (Fu *et al.*, 2007), automatic fix generation (Thomas and Williams, 2007; Dysart and Sherriff, 2008), dynamic candidate evaluations (Bandhakavi *et al.*, 2007), modelling SQL injection attacks (Kiezun *et al.*, 2009; Ali *et al.*, 2011), character distribution models (Kiani *et al.*, 2008), genetic algorithms (Shahriar and Zulkernine, 2008), static and dynamic analysis (Orso *et al.*, 2008), symbolic execution (Fu and Qian, 2008), location specific signatures (Mitropoulos and Spinellis, 2009), testing and comparing (Fonseca *et al.*, 2007), symptom correlation approach (Ficco *et al.*, 2009), static analysis and penetration testing (Antunes and Vieira, 2009a, b), detection of attacks in web services prepared statement based solution (Thomas *et al.*, 2009), database centric web services (Laranjeiro *et al.*, 2009), proxy-based solution (Liu *et al.*, 2009) are different kinds of solutions found in the literature.

There are many other approaches used for preventing SQL injection attacks as explored by Vieira *et al.* (2009), Antunes *et al.* (2009), Ciampa *et al.* (2010), Kindy and Pathan (2011), Wang *et al.* (2010), Tajpour *et al.* (2010a-c), Halder and Cortesi (2010), Ali *et al.* (2011), Khoury *et al.* (2011), Lee *et al.* (2011), Johari and Sharma (2012), Clarke (2012) and Das *et al.* (2010). Out of them Vieira *et al.* (2009) and Antunes *et al.* (2009) explore solutions to attacks in web services. A good survey of SQL injection attacks is found by Shrivastava and Tripathi (2012) and Johari and Sharma (2012). Augmented attack modelling (Wang *et al.*, 2010) obfuscation based analysis (Halder and Cortesi, 2010) and dynamic query matching are other solutions available.

MATERIALS AND METHODS

Overview of the proposed architecture: The proposed framework is flexible and extendable as it provides placeholders for various methods used in the process. The placeholder (shown in red colour horizontal bar) indicates that the method is extendable and our prototype support plugging new method in future, so as to support multiple kinds of methods for a particular mechanism.

As can be shown in Fig. 2, it is evident that there are two distinct layers in the proposed solution. They are categorized into offline and online processing. Offline processing is the continuous database server side work irrespective of user queries. Even when no user query is being executed, the offline processing takes place. The offline processing includes extraction of database logs that hold previously executed queries, pre-processing, conversion and transform. Once database logs are extracted they are pre-processed in order to let them to be suitable for further processing. The pre-processed content is given to a conversion technique which considers syntactic and semantic features of queries and converts the content into an intermediary form that can be used conveniently. It is in our case in the form of multi-dimensional feature vectors that can be used to build a model by using a transformation technique. Once model is built, it is provided to online processing. The modelling is done by to the transformation function that makes use of classification techniques like SVM.

The online processing starts only when user provides an SQL query either directly or through an application. Then the query is parsed and matched with the query representations in the model. Then, the suspected query is given to detection method that thoroughly investigates to know whether it is an injection attack. Once it is confirmed that there is injection attack, this query is provided to the offline processing where model gets updated in order to have knowhow on such queries to handle in future with ease. Once the attack is detected it is prevented by ignoring the query or ensuring the ACID features of the current transaction.

RESULTS AND DISCUSSION

We used PostgreSQL which is an open source RDBMS. As it allows users to customize, the kernel level interfaces are exploited and the proposed system is

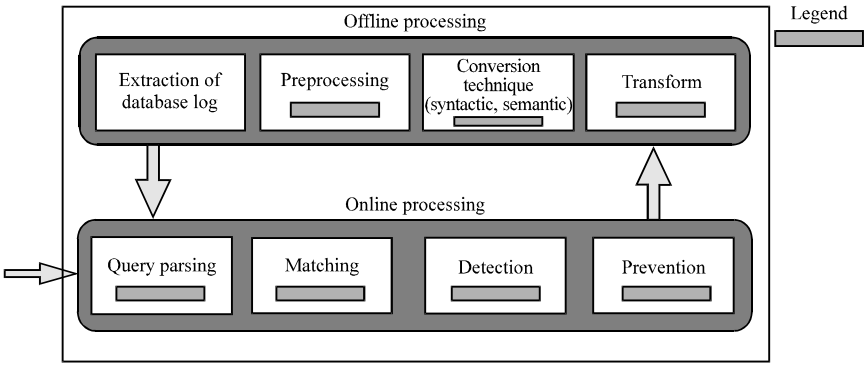

Fig. 2: Overview of the proposed architecture

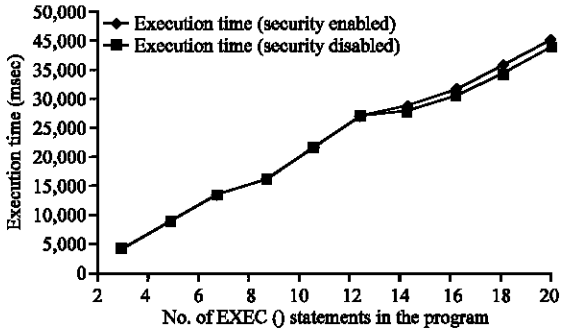

Fig. 3: Performance comparison

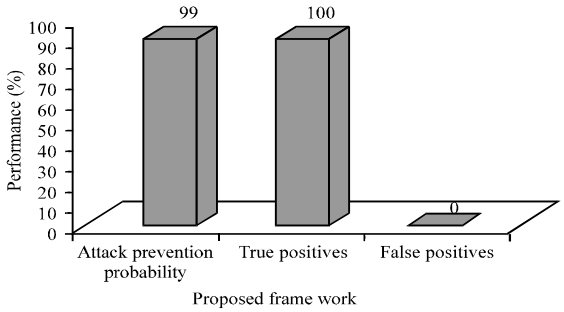

Fig. 4: Performance of the proposed framework

implemented. The whole framework has nothing to do with client applications. Instead it is at server side and its offline and online processing methods are executed in the database server itself. We implemented the whole functionality using PL/SQL package with all underlying functions. These functions are executed in such a way the improved versions can be plugged in future with ease. The standard interfaces we used can help in switching to different future techniques.

Figure 3 shows that, it is evident that there is no much difference in execution time of the proposed security when compared with queries without security. The reason behind this is that the offline processing makes a model that can be used to implement security without taking additional time. Therefore, there is negligible difference between execution time between security enabled and security disabled settings.

Figure 4 shows that, the proposed solution achieves 100% true positives and 99% probability of detection probability.

CONCLUSION

In this study, we focused on SQL injection attacks and prevention measures. We proposed a generic framework that is extendable and flexible. The framework is based on the data mining techniques to detect and prevent SQL injection attacks. The existing solutions that operate at application level fail to address the issues when attackers target stored procedures that reside at server side. The framework overcomes this problem by analyzing the logs and building a model that can be used to detect SQLIAs. The framework has two layers known as offline and online processing. Offline processing makes a model that will be used to detect SQLIAs. The model is given to online processing layer where actual query is received from application and detection and prevention mechanisms are applied as discussed in this study. We did experiments with PostgreSQL and the results reveal that our solution achieves 99% probability in detection and 100% true positives.

REFERENCES

Ali, A.B.M., A.Y.I. Shakhatreh, M.S. Abdullah and J. Alostad, 2011. SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks. Procedia Comput. Sci., 3: 453-458.

Antunes, N. and M. Vieira, 2009b. Comparing the effectiveness of penetration testing and static code analysis on the detection of SQL injection vulnerabilities in web services. Proceedings of the 15th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'09), November 16-18, 2009, IEEE, Shanghai, China, ISBN:978-0-7695-3849-5, pp: 301-306.

Antunes, N. and M. Vieira, 2009a. Detecting SQL injection vulnerabilities in web services. Proceedings of the 4th Latin-American Symposium on Dependable Computing, September 1-4, 2009, Joao Pessoa, pp: 17-24.

Antunes, N., N. Laranjeiro, M. Vieira and H. Madeira, 2009. Effective detection of SQL/xpath injection vulnerabilities in web services. Proceedings of the IEEE International Conference on Services Computing, September 21-25, 2009, Bangalore, pp: 260-267.

Bandhakavi, S., P. Bisht, P. Madhusudan and V.N. Venkatakrishnan, 2007. CANDID: Preventing SQL injection attacks using dynamic candidate evaluations. Proceedings of the 14th ACM International Conference on Computer and Communications Security, October 28, 2007, ACM, New York, USA., ISBN:978-1-59593-703-2, pp: 12-24.

Buehrer, G., B.W. Weide and P.A.G. Sivilotti, 2005. Using parse tree validation to prevent SQL injection attacks. Proceedings of the 5th International Workshop on Software Engineering and Middleware, September 5-6, 2005, Lisbon, Portugal, pp: 106-113.

Chapela, V., 2005. Advanced SQL injection. MCS Thesis, OWASP, Maryland, USA.

Ciampa, A., C.A. Visaggio and D.M. Penta, 2010. A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications. Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, May 02, 2010, ACM, New York, USA., ISBN:978-1-60558-965-7, pp: 43-49.

Clarke, J., 2012. SQL Injection Attacks and Defense. Elsevier, Amsterdam, Netherlands, ISBN: 978-1-59749-963-7, Pages: 547.

Das, D., U. Sharma and D.K. Bhattacharyya, 2010. An approach to detection of SQL injection vulnerabilities based on dynamic query matching. Intl. J. Comput. Appl., 1: 39-45.

Doug, A.N.T.S.G. and G.D. Evans, 2004. Automatically hardening web applications using precise tainting. Master Thesis, University of Virginia, Charlottesville, Virginia.

Dysart, F. and M. Sherriff, 2008. Automated fix generator for SQL injection attacks. Proceedings of the 19th International Symposium on Software Reliability Engineering, November 10-14, 2008, IEEE, Seattle, Washington, ISBN:978-0-7695-3405-3, pp: 311-312.

Fayo, E.M., 2005. Advanced SQL injection in Oracle databases. Mandalay Bay, Nevada, USA.

Ficco, M., L. Coppolino and L. Romano, 2009. A weight-based symptom correlation approach to SQL injection attacks. Proceedings of the 4th Latin-American International Symposium on Dependable Computing (LADC'09), September 1-4, 2009, IEEE, Joao Pessoa, Brazil, ISBN:978-1-4244-4678-0, pp: 9-16.

Fonseca, J., M. Vieira and H. Madeira, 2007. Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks. Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing, December 17-19, 2007, Melbourne, Qld, pp: 365-372.

Fu, X. and K. Qian, 2008. SAFELI: SQL injection scanner using symbolic execution. Proceedings of the Workshop on Testing, Analysis and Verification of Web Services and Applications, July 20-24, 2008, Seattle, WA., USA., pp: 34-39.

Fu, X., X. Lu, P. Verger, B.S. Chen, K. Qian and L. Tao, 2007. A static analysis framework for detecting SQL injection vulnerabilities. Proceedings of the IEEE Annual International Computer Software and Application Conference, Volume 1, July 24-27, 2007, Beijing, pp: 87-96.

Halder, R. and A. Cortesi, 2010. Obfuscation-based analysis of SQL injection attacks. Proceedings of the IEEE Symposium on Computers and Communications, June 22-25, 2010, Riccione, Italy, pp: 931-938.

Halfond, W.G. and A. Orso, 2005b. Amnesia: Analysis and monitoring for neutralizing SQL-injection attacks. Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, November 7-11, 2005, Long Beach, CA., USA., pp: 174-183.

Halfond, W.G. and A. Orso, 2005a. Combining static analysis and runtime monitoring to counter SQL-injection attacks. Proceedings of the ACM International Conference on SIGSOFT Software Engineering Notes Vol. 30, May 17, 2005, ACM, New York, USA., pp: 1-7.

Johari, R. and P. Sharma, 2012. A survey on web application vulnerabilities (SQLIA, XSS) exploitation and security engine for SQL injection. Proceedings of the International Conference on Communication Systems and Network Technologies, May 11-13, 2012, Rajkot, pp: 453-458.

Kals, S., E. Kirda, C. Kruegel and N. Jovanovic, 2006. Secubat: A web vulnerability scanner. Proceedings of the 15th International Conference on World Wide Web, May 22-26, 2006, ACM, New York, USA., ISBN:1-59593-323-9, pp: 247-256.

Khoury, N., P. Zavarsky, D. Lindskog and R. Ruhl, 2011. An analysis of black-box web application security scanners against stored SQL injection. Proceedings of the 2011 IEEE 3rd Inernational Conference on Privacy, Security, Risk and Trust (PASSAT) and Social Computing (SocialCom), October 9-11, 2011, IEEE, Boston, Massachusetts, USA., ISBN:978-1-4577-1931-8, pp: 1095-1101.

Kiani, M., A. Clark and G. Mohay, 2008. Evaluation of anomaly based character distribution models in the detection of SQL injection attacks. Proceedings of the 3rd International Conference on Availability, Reliability and Security, March 4-7, 2008, Barcelona, pp: 47-55.

Kiezun, A., P.J. Guo, K. Jayaraman and M.D. Ernst, 2009. Automatic creation of SQL injection and cross-site scripting attacks. Proceedings of the 31st International Conference on Software Engineering, May 20-22, 2009, Vancouver, BC, Canada, pp: 199-209.

Kindy, D.A. and A.S.K. Pathan, 2011. A survey on SQL injection: Vulnerabilities, attacks and prevention techniques. Proceedings of the IEEE 15th International Symposium on Consumer Electronics, June 14-17, 2011, Singapore, pp: 468-471.

Laranjeiro, N., M. Vieira and H. Madeira, 2009. Protecting Database Centric Web Services against SQL/XPath Injection Attacks. In: Database and Expert Systems Applications, Kung, J. and W. Roland (Eds.). Springer, Berlin, Germany, ISBN:978-3-642-03572-2, pp: 271-278.

Lee, I., S.J.S. Yeoc and J. Moond, 2011. A novel method for SQL injection attack detection based on removing SQL query attribute. J. Math. Comput. Mod., 55: 58-68.

Litchfield, D., 2005. Data-mining with SQL injection and inference. Next Generation Security Software Ltd., Manchester, UK. https://www.exploit-db.com/docs/215.pdf.

Liu, A., Y. Yuan, D. Wijesekera and A. Stavrou, 2009. SQLProb: A proxy-based architecture towards preventing SQL injection attacks. Proceedings of the ACM Symposium on Applied Computing, March 8-12, 2009, Honolulu, HI., USA., pp: 2054-2061.

McClure, R.A. and I.H. Kruger, 2005. SQL DOM: Compile time checking of dynamic SQL statements. Proceedings of the 27th International Conference on Software Engineering (ICSE 2005), May 15-21, 2005, IEEE, California, USA., ISBN:1-59593-963-2, pp: 88-96.

Merlo, E., D. Letarte and G. Antoniol, 2006. Insider and ousider threat-sensitive SQL injection vulnerability analysis in PHP. Proceedings of the 13th International Working Conference on Reverse Engineering (WCRE'06), October 23-27, 2006, IEEE, Benevento, Italy, pp: 147-156.

Merlo, E., D. Letarte and G. Antoniol, 2007. Automated protection of PHP applications against SQL-injection attacks. Proceedings of the 11th European International Conference on Software Maintenance and Reengineering (CSMR'07), March 21-23, 2007, IEEE, Amsterdam, Netherlands, pp: 191-202.

Mitropoulos, D. and D. Spinellis, 2009. SDriver: Location-specific signatures prevent SQL injection attacks. Comput. Secur., 28: 121-129.

Orso, A., W. Lee and A. Shostack, 2008. Preventing SQL code injection by combining static and runtime analysis. Master Thesis, Defense Technical Information Center, Fort Belvoir, Virginia, USA.

Rietta, F. S., 2006. Application layer intrusion detection for SQL injection. Proceedings of the 44th Annual International Conference on the Southeast Regional, March 10-12, 2006, ACM, New York, USA., ISBN:1-59593-315-8, pp: 531-536.

Shahriar, H. and M. Zulkernine, 2008. MUSIC: Mutation-based SQL injection vulnerability checking. Proceedings of the 8th International Conference on Quality Software, August 12-13, 2008, Oxford, pp: 77-86.

Shin, Y., W. Laurie and X. Tao, 2006. SQLUnitGen: SQL injection testing using static and dynamic analysis. Master Thesis, Department of Computer Science, North Carolina State University, Raleigh, North Carolina.

Shrivastava, S. and R.R.K. Tripathi, 2012. Attacks due to SQL injection and their prevention method for web-application. Intl. J. Comput. Sci. Inf. Technol., 3: 3615-3618.

Tajpour, A., M. Massrum and M.Z. Heydari, 2010a. Comparison of SQL injection detection and prevention techniques. Proceeding of the 2nd International Conference Education Technology and Computer, June 22-24, 2010, Shanghai, pp: 174-179.

Tajpour, A., M.Z. Heydari, M. Masrom and S. Ibrahim, 2010c. SQL injection detection and prevention tools assessment. Proceedings of the 2010 3rd IEEE International Conference on Computer Science and the Information Technology (ICCSIT) Vol. 9, July 9-11, 2010, IEEE, Chengdu, China, ISBN:978-1-4244-5537-9, pp: 518-522.

Tajpour, A., Z. JorJor and M. Shooshtari, 2010b. Evaluation of SQL injection detection and prevention techniques. Proceeding of the 2nd International Conference Computational Intelligence, Communication Systems and Networks, July 28-30, 2010, Liverpool, pp: 216-221.

Thomas, S. and L. Williams, 2007. Using automated fix generation to secure SQL statements. Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems, May 20-26, 2007, IEEE, Washington, DC., USA., pp: 1-10.

Thomas, S., L. Williams and T. Xie, 2009. On automated prepared statement generation to remove SQL injection vulnerabilities. Inform. Software Technol., 51: 589-598.

Vieira, M., N. Antunes and H. Madeira, 2009. Using web security scanners to detect vulnerabilities in web services. Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks, June 29-July 2, 2009, Lisbon, Portugal, pp: 566-571.

Wang, J., R.C.W. Phan, J.N. Whitley and D.J. Parish, 2010. Augmented attack tree modeling of SQL injection attacks. Proceedings of the 2nd IEEE International Conference on Information Management and Engineering, April 16-18, 2010, Chengdu, pp: 182-186.

Wei, K., M. Muthuprasanna and S. Kothari, 2006. Preventing SQL injection attacks in stored procedures. Proceedings of the International Conference on Software Engineering Australian, April 18-21, 2006, IEEE, Sydney, New South Wales, Australia, pp: 1-8.