Study of SQL Injection Attacks and Countermeasures

Sayyed Mohammad Sadegh Sajjadi and Bahare Tajalli Pour

***Abstract*—SQL injection is an attack technique that exploits a security vulnerability occurring in the database layer of an application and a service. This is most often found within web pages with dynamic content. This paper provides taxonomy on SQL injection prevention and detection approaches. Furthermore, for each type of vulnerability, we provide descriptions of how attacks of that type could take advantage of that vulnerability and perform attack. We also present and analysis some of existing detection and prevention techniques against SQL injection attacks. Finally, we compare different type of approaches and techniques and provide a list of their deployment requirements.**

***Index Terms*—SQL injection attack, SQL queries, web application, DBMS, taxonomy, web application security.**

I. Introduction

Nowadays by rapid development of Internet, online services use web applications to present their services and use the web paradigm are becoming an interesting strategy for application software companies. It allows the design of pervasive applications which can be potentially used by thousands of customers from simple web clients. World Wide Web a great growth, but attacks on web increased simultaneously. Therefore, effective security mechanisms on web applications and addressing them seem to be very important.

Code injection is a type of utilization caused by processing invalid user inputs. The concept of injection attacks is to inject (or insert) malicious code into a program so as to change structure of SQL query. Such an attack may be performed by adding strings of malicious characters into data values in the form or argument values in the URL. Injection attacks generally take advantages of improper validation over input/output data. SQL Injection Attack or SQLIA is a type of code injection attacks which consist of injection of malicious SQL commands by means of input data from the client to the application that are later passed to the instance of the database for execution and aim to affect the execution of predefined SQL commands. There are a number of ways a programmer/system administrator can prevent or counter attacks made on their systems. In these ways a programmer or system administrator uses different techniques in development cycle of application which contains uses parameterized queries, least privilege, different account, customized error message and etc. Although these techniques remain the best way to prevent SQL injection vulnerabilities, but their application is problematic in practice. These techniques are prone to human errors and are not as rigorously and completely applied as automated techniques. Whereas most developers do make an effort to code safely, it is extremely difficult to apply defensive coding practices rigorously and correctly to all sources of input. Therefore researchers suggested a range of techniques and approaches to assist developers and compensate for shortcoming in the application of defensive coding. In these techniques utilize static, dynamic or hybrid analysis for detecting SQLIA. Some methods use machine learning techniques for improvement and training their legitimate query lists. Furthermore, also there are other ways for countering to SQLIA which we described in the rest of this paper.

The rest of this paper is organized as follows. We begin by motivating vulnerability concepts and introducing SQL injection attacks in Section II. Section III present taxonomy of different type of SQL injection prevention and detection approaches. In Section IV, we compare and evaluate different SQLIA countering techniques and characterize their deployment requirements. Finally, a brief conclusion of this paper is provided in Section V.

II. Understanding Vulnerability and SQL Injection

Vulnerability is a weakness in the application which can be a design flaw or an implementation bug. An attacker can use such vulnerabilities, harm to the stakeholders of an application. SQL Injection Attack, Cross-Site Scripting (XSS), Cross- Site Request Forgery (CSRF), Broken Authentication and Session Management are some of the application layer vulnerabilities targeting most of the current web application [1]. According to reports that are provided by OWASP [2] and WHID [3], among all these attacks SQLIA and XSS are very common. SQLIA is considered a severe of attack affecting confidentiality, integrity and availability of information. SQL injection vulnerability is a type of attacks adds Structured Query Language code to a web form input box to gain access or make changes to data. By using this vulnerability an attacker could send his commands directly to web application's underlying database and destroy functionality or confidentiality. SQLIA can be classified into five basic classes based on vulnerabilities in web applications. This classification is illustrated in Table I.

III. Review and Analysis Different Types of SQLIA Countermeasures

Several papers in literatures have proposed ways to prevent SQLIA in the application or database tier. We provide a

Manuscript received January 22, 2013; revised April 15, 2013.

S. Sajjadi is with the Department of Electrical, Computer and IT Engineering, Islamic Azad University, Qazvin Branch, Qazvin, Iran (e-mail: s.sajjadi@qiau.ac.ir).

B. Tajalli Pour is with Department of Computer Engineering, Islamic Azad University, Tehran North Branch, Tehran, Iran (e-mail: bahar_tj@yahoo.com).

comprehensive survey of SQLIA detection and prevention techniques. SQLIA countermeasures techniques are divided into three main approaches: static, dynamic and hybrid approaches [4]. Static approaches are desirable during development and testing phase of applications. Developers should follow some techniques for SQL injection prevention. Static approaches counteract the possibility of SQLIA at compile time. Whereas, dynamic approaches are useful for analysis of dynamic SQL query, generated by web application. This approach performs countering the possibility of SQLIA at runtime. Both approaches may need analysis or modification of application's source code.

In hybrid approaches exploit a combination of static and dynamic approaches. These approaches attempt to utilize advantages of both approaches for preventing and detecting SQLIA. In the rest of this paper, some of the new and popular existing static, dynamic and hybrid techniques are presented.

A. Static Approach

1) An algorithm of prepared statement replacement for removing SQLIVs

Thomas *et al.* [5] proposed a prepared statement replacement (PSR) algorithm and corresponding automation for removing SQLIA vulnerabilities from vulnerable SQL statements by replacing them with secure prepared statements. This method analysis source code containing SQLIVs and generates a specific recommended code structure containing prepared statements. An SQLIV exists when an SQL statement does not keep statement structure and input separate.

TABLE I: SQLIA CLAASIFICATION BASED ON VULNERABILITY.

Vulnerability	A brief explanation
Bypassing Web Application Authentication	This is the most common usage adopted by the attackers to bypass authentication pages, used in web applications. In this category of attack, an attacker exploits an input field that is used in a query's 'where' condition part.
Getting Knowledge of Database Fingerprinting	This attack is considered as pre-attack preparation by an attacker. This category of attack is performed by entering some inputs by which it generates an illegal or the logically incorrect queries. The error messages reveal the names of the tables and the columns that cause error. The attacker also comes to know about the application database used in the backend server.
Injection with UNION query	In such an attack, an attacker extracts data from a table which is different from the one that was intended in the web application by the developer. An attacker exploits a vulnerable parameter to change the data set returned for a given query.
Damaging with additional injected query	This category of attack is generally very harmful. An attacker enters input such that an additional injected query is generated along with the original query.
Remote execution of stored procedures	This category of attack is conducted by executing the procedures, stored previously by the web application developer.

PSR-algorithm collects information from application's source code which possible including SQLIVs. Then generates secure prepared statement code that maintains functional integrity. Another algorithm which called Prepared Statement Replacement Generator (PSR-Generator) is created for automates the generation of the prepared statement-based code in Java, which results from the PSR-Algorithm. PSR-Algorithm is useful for developers which have source code contains SQLIVs and need to be removed. Authors claim that their proposed method is remove SQLIVs with minimal manual intervention. Note that PSR-Algorithm is used to remove only SQLIV and does not have to be integrated into the runtime environment.

2) MUSIC: mutation-based SQL injection vulnerability checking

Shahriar and Zulkernine[6], proposed a MUtation-based SQL Injection vulnerabilities Checking (testing) tool (MUSIC) that automatically generates mutants for the applications written in Java Server Pages (JSP) and performs mutation analysis. Mutation is the act of intentionally modifying a program's code, then re-running a suite of valid unit tests against the mutated program. Mutation testing is a method of fault-based software testing, which involves modifying programs' source code or byte code in small ways. Mutation testing is done by selecting a set of mutation operators and then applying them to the source program one at a time for each applicable piece of the source code. The result of applying one mutation operator to the program is called a mutant. These mutants are killed by a test case if it is causes different end output or different intermediate state between the original program and a mutant. Otherwise the mutant is remaining alive. Additional test cases should be generated for killing live mutants. Authors proposed nine mutation operators to inject SQLIV in source code of application which four of them inject faults into generated SQL queries and remaining five of operators inject faults into the API method calls. However, MUSIC is very simple and effective way for testing SQL queries having simple form, but it cannot address the SQLIV of stored procedures.

3) Sania: syntactic and semantic analysis for automated testing against SQL injection

Sania [7], is a technique for detecting SQLIV in web applications in development and debugging phase which using the following procedures. 1) Sania intercepts the SQL queries between a web application and a database. Then, collects normal SQL queries between client and web applications and between the web application and database, and analysis the vulnerabilities. 2) It automatically generates elaborate attacks according to the syntax and semantics of the potentially vulnerable spots in the SQL queries. 3) After attacking with the generated code, it collects the SQL queries generated from the attack. 4) Sania compares the parse trees of the intended SQL query and those resulting after an attack to assess the safety of these spots. 5) Finally, it determines whether the attack succeeded or not. By analyzing the syntax in the parse tree of SQL queries, it is possible to generate precise pinpoint attack requests.

B. Dynamic Approach

We discuss three different popular dynamic approaches for countering to SQLIA.

1) AIIDA-SQL [8]

This method suggests a hybrid approach based on Adaptive Intelligent Intrusion Detection (AIIDA-SQL) for detection of SQLIA. AIIDA-SQL combines the advantages of Case-Based Reasoning (CBR) systems, such as learning and adaptation, with the predictive capabilities of a combination of Artificial Neural Network (ANN) and Support Vector Machine (SVM). Through these mechanisms, their take advantages of both strategies in order to trustworthy classifying the SQL queries. In final manner, in order to classify SQL queries as distrustful, utilized a virtualization mechanism, which combines clustering techniques and unsupervised neural models to reduce dimensionality of the data.

2) *A query tokenization based method*

In [9], a technique based query tokenization is proposed for detect and prevent SQL injection attacks. This method checks user inputs whether their cause changes query's intended result. At the next step, this method tokenizing original query and malicious injected query, separately. After tokenizing, two arrays are created by all tokens. Finally, the lengths of obtained arrays are compared. If their length be different, an injection attack is detected.

3) *A learning based approach*

Bertino *et al.* [10] have proposed a framework based on anomaly detection techniques to detect malicious behavior of database application programs. The approach is as follows. At first step, a fingerprint of an application program based on SQL queries is created. Then, take advantages of association rule mining techniques to extract useful rules from these fingerprints. These rules depict normal behavior of the database application. Finally, dynamic queries check against these rules to detect injection attacks that not conform to these rules.

C. *Hybrid Approach*

1) *A method based on removing SQL query attribute values*

Lee *et al.* [11] proposed a simple and efficient method for detecting SQLIA. Their method utilizes static and dynamic phases for finding vulnerabilities in web application. This method removes the attribute values of SQL queries at runtime (dynamic method) and compares them with the SQL queries analyzed in advance (static method). It detects attacks by comparing the structure and the grammar of the queries. If a dynamically generated query has a different structure or uses a different grammar from that of a static query, it is detected. Authors shown that their proposed method has time complexity O (1) and can implement on any type of DBMS.

2) *Obfuscation-based Analysis of SQL Injection Attacks*

Halder and Cortesi [12] proposes the obfuscation and deobfuscation based technique to detect the presence of possible SQLIA in a query before submitting it to a DBMS. In the static phase, the queries in the application are replaced by queries in obfuscated form. Now the Obfuscated code is a source code that has been made difficult for human. In obfuscation approach the possible attack injection are verified at atomic formula level and only those atomic formulas which are tagged as vulnerable, also this approach avoids the root cause of SQL injection attacks in dynamic query generation. Authors show that their proposed algorithm could detect SQLIA with negligible runtime overhead and do not dependent on specific application.

IV. Evaluation

In this section, SQL injection countermeasure techniques presented in Section III would be compared. In fact, SQLIA countermeasures dependent on that prevent or detect SQL injection attacks, can divide two main approaches: detection techniques and prevention techniques. In prevention techniques, prevent to build injection SQL statements through user input analysis whereas in detection-focused techniques detect attacks in runtime and prevent to gain access to back-end database. To determine the effort and additional elements required to use the technique, we examined the author's description of the technique and its current implementation. Table II summarize the result of comparison of additional factors.

TABLE II: Analysis of Additional Elements of Each Detection and Prevention Method

Detection/ prevention method	**Modify code base**	**Detection**	**prevention**	**Additional elements**
PSR-Algorithm [5]	Yes	N/A	Code suggestion	Developer learning,
MUSIC[6]	Yes	N/A	Auto	N/A
Sania[7]	No	N/A	Auto	Proxy filter, Attack code generator
AIIDA-SQL [8]	No	Auto	Auto	CBR engine, Developer learning
Query tokenization [9]	No	Auto	Auto	Query parser method
Learning[10]	No	Auto	N/A	Developer leaning, Fingerprinting database application, Anomaly detection model training sets
Removing SQL query attribute[11]	Yes	Auto	Auto	Developer leaning
Obfuscation-based analysis[12]	No	Auto	Auto	N/A

V. Conclusion

This paper provides taxonomy of methods for prevent and detect SQL injection attacks. We first define vulnerabilities in web application and how these vulnerabilities may cause SQL injection attacks. Then, we present a classification of SQLIA based on vulnerability. Afterwards, divide the SQL injection and prevention methods to three different categories: static, dynamic and hybrid approaches. These approaches different in the time which are counteracting to possibility of SQLIA. The paper discusses different SQL detection and prevention techniques for a given attack which recently been proposed. Furthermore, we evaluated these techniques, with respect to deployment requirements. For all the negative impact of SQL injection vulnerability, the countermeasures are surprisingly simple to enact. The first rule, which applies to all Web development, is to validate user-supplied data. SQL injection payloads require a limited set of characters to fully exploit vulnerability. Web sites should match the data received from a user against the type (for example, integer, string, date) and content (for example, e-mail address, first name, telephone number) expected. We believe that each detection or prevention technique cannot provide complete protection against SQLIA, but a combination of the presented mechanisms will cover a wide range of SQL injection attacks which will culminate in a more secure and reliable database which is protected against SQL Injection Attacks.

References

[1] M. Shema, *Seven Deadliest Web Application Attacks*, Elsevier Inc. , 2010, pp. 47-69.

[2] The Open Web Application Security Project, OWASP TOP 10 Projects. [Online]. Available: http://www.owasp.org/

[3] Web Hacking Incident Database Project. [Online]. Available: http://projects.webappsec.org/.

[4] W. G. Halfond, J. Viegas, and A. Orso , "A Classification of SQL Injection Attacks and Countermeasures," in *Proc. the International Symposium on Secure Software Engineering*, 2006.

[5] S. Thomas, L. Williams, and T. Xie, "On automated prepared statement generation to remove SQL injection vulnerabilities," *Information and Software Technology Journal*, vol. 51, 2009, pp. 589-598.

[6] H. Shahriar and M. Zulkernine, "MUSIC: Mutation-based SQL Injection Vulnerability Checking," in *Proc. The Eighth International Conference on Quality Software(QSIC 08)*, IEEE Press, 2008, pp. 77-86.

[7] Y. Kosuga, K. Kernel, M. Hanaoka, M. Hishiyama, and Y. Takahama, "Sania: syntactic and semantic analysis for automated testing against SQL injection," in *Proc. the Computer Security Applications Conference* , 2007, pp. 107–117.

[8] C. Pinzon, J. F. Paz, J. Bajo, A. Herrero, and E.Corchado, "AIIDA-SQL: An Adaptive Intelligent Intrusion Detector Agent for Detecting SQL Injection Attacks," in *Proc. 10th International Conference on Hybrid Intellignt Systems (HIS)*, IEEE Press, 2010, pp. 73-78.

[9] N. Lambert and K. S. Lin," Use of Query Tokenization to detect and prevent SQL Injection Attacks," in *Proc. 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT*), July 2010, pp. 438-440.

[10] E. Bertino, A. Kamra, and J. P. Early, "Profiling Database Applications to Detect SQL Injection Attacks," in *Proc. IEEE Internation Conference on Performance, Computing, and Communications (IPCCC)*, 2007, pp. 449-458.

[11] I. Lee, S. Jeong, S. Yeo, and J. Moon, "A novel method for SQL injection attack detection based on removing SQL query attribute values," *Mathematical and Computing Modeling Journal* , vol. 55, pp. 58-68, 2011.

[12] R. Halder and A. Cortesi, "Obfuscation-based Analysis of SQL Injection Attacks," in *Proc. IEEE Symp. On Computers and Communications (ISCC)*, 2010, pp. 931-938.

Sayyed Mohammad Sadegh Sajjadi received the B.S in computer software engineering from Islamic Azad University Shiraz Branch in 2007 and M.S. degree in computer software engineering from Islamic Azad University Qazvin Branch in 2012. His current research interest is designing and developing schemes and algorithms for countering to code injection attacks. His M.S's Thesis was a novel technique for detecting SQL injection attacks in web applications via machine learning methods. He is also very interested in the study of intrusion detection methods in web application, vulnerability testing of software, database and network security and cloud computing privacy and security.

Bahare Tajalli Pour received the B.S. in computer software engineering from Islamic Azad University Tehran North Branch in 2000 and M.S. degree in Information Security from Islamic Azad University Tehran North Branch in 2011. Her M.S.'s thesis was a method for prevention of malicious transactions in database management. She is very interested in the study of cryptography and foundation of computer Security, Network Security Application, etc.