E-ISSN: 2706-8927
P-ISSN: 2706-8919
IJAAS 2020; 2(1): 06-10
Received: 05-11-2019
Accepted: 07-12-2019

Zargay Habibi
Assistant Professor, Database and Information Systems Department Nangarhar University, Jalalabad, Afghanistan

Corresponding Author:
Zargay Habibi
Assistant Professor, Database and Information Systems Department Nangarhar University, Jalalabad, Afghanistan

A review on the comparable analyses of SQL injection: attacks, vulnerabilities, and their detection techniques

Zargay Habibi

Abstract
Background: The concept of SQL injection is indeed a technique that exploits a security vulnerability that occurs in the application layer of a database. The attack takes benefits of weak input validation in code and website administration. It allows attackers to get illegal access to the back-and part of the database to modify the intended application which generate SQL queries.
Method: This paper does a review on comparable analysis to find out more on different types of attacks, vulnerabilities and their detection techniques.
Conclusion: According to the reviews which we have done researchers have offered several solutions to address SQL injection challenges. In addition to this, many of them have limitations and mostly cannot address all type of injection problems. More to this, several new types of injection attacks have arisen in the last several years. To better avoid such attacks, specifying and knowing existing techniques and methods are very important. In this research paper we discuss and review all types of SQL injection attack as well as different techniques and tools which can detect or avoid these attacks.

Keywords: SQL injection attacks, web application, prevention, detection

1. Introduction
Internet is a huge data and information infrastructure. Unaware of the security and privacy, the internet is becoming a repository of information. Information and data is the most important business asset in today's environment and achieving an appropriate level of Information Security. Applications are vulnerable to a variety of new security threats. One of the most threads to web application is SQL injection attack. According to Open Web Application Security Project (OWASP) [1] top 10 threats for web application security in 2013, injection attacks stands first. For example, financial fraud, online banking, theft of private data, shopping and cyber terrorism. Web applications that are susceptible to SQL injection may allow an attacker to gain complete access to their essential databases. To implement security guidelines inside or outside the database the access of the sensitive databases should be monitored. Detection or prevention of SQL injection attacks is a topic of active research in the industry and academia [5].
SQL injection attacks can be done in different ways like modifying the data, query manipulation, data extraction etc. Through the execution of altered SQL query, the attackers can get unauthorized access to the application, get important and sensitive identity information. Web Application Firewalls (WAF), such as Apache Mod Security, offer some level of protection. However, they require lots of complex configuration and high processing time in order to detect malicious traffic. As SQL injection vectors can be formed in numerous ways, attacks can be specifically devised from cleverly crafted injection vectors to bypass detection techniques [11-13]
In later years, many literatures have been conducted to research the detection and defense of SQL injection attack, but regretfully some of them have the low efficiency and high rate of false alarm. Traditional loophole attack detection and prevention techniques include non-executive stack technology, nonexecutive heap and data technology, address space randomization techniques, misuse detection, abnormal detection and so on. To achieve those purposes, automatic tools and security system have been implemented, but none of them are complete or accurate enough to guarantee an absolute level of security on web application [4].

2. SQL Syntax Analysis
Structure Query language system has complete and perfect grammar and lexical process, which constitute the grammar rules to follow on BNF rule [6]. SQL syntax mode can be defined as follows: SQL Expression = ({sqlexp0072}, + I {sqlexpr},)" I). Sqlexpr is the syntax block of SQL and this form of a formal language expression can be described by a finite state automaton of formal language as is showed in Figure 1. The above syntax form can be expanded as follows: SQL Expression = sqlexpr _keyword I, L sqlexpr _keyword,)" I, sqlexpr_keyword = expr I sqlexpr_keyword I "," expr. In this expression, expr is the basic statement block of SQL, and sqlexpr_keyword is the keywords of the SQL statement.

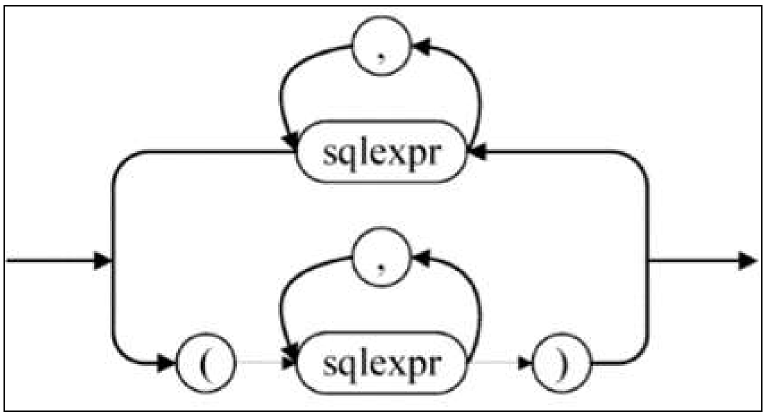

Fig 1: SQL syntax rules

As per the figure 1 SQL syntax is defined in recursive nested form, and syntax can be expanded top-down. According to the syntax rule and logical structure of syntax formation, SQL syntax analysis process can be divided into lexical analyzer and syntax analyzer, and lexical analyzer is the basis of the syntax analyzer [7].

2.1 Lexical Analysis
Lexical analysis is the basis of syntax analysis. Its function is to parse the dynamic SQL progressive and continuous string into a number of Token tags, then it transfer the Token tag to the syntax analysis module for specific syntax analysis. The process is shown in figure 2.

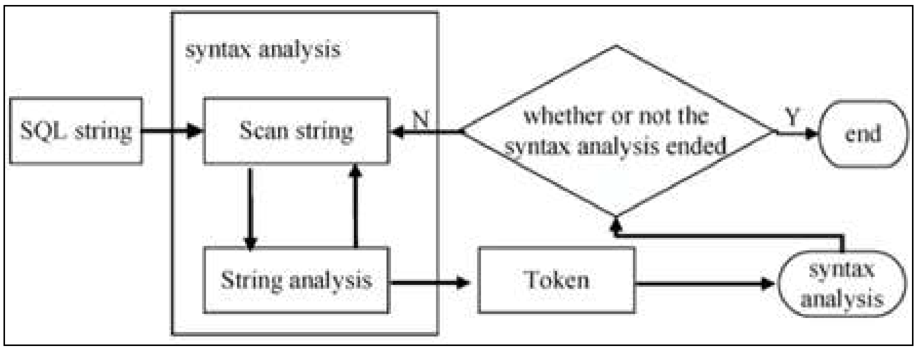

Fig 2: Token flow graph

There are four categories of word formations of SQL statement in database systems: SQL reserved word, User defined tags and data constants, special characters and system tag of database. Such as system tables, system function and storage process of system [7, 8].

1. **SQL reserved word:** SQL reserved word is a kind of keywords which has a special function and represents the type of SQL statement. Such as select, insert, Update, etc. In the syntax analysis phase, these keywords have the mark function and represent the special function of the analysis statement.

2. **User-defined tags and data constants:** In the SQL language, the user-defined tags is a string of characters which begin with letters and numbers, including table name, function name, stored procedure name and view names. Constant data into numeric constants and string constants, only represent the data values. Data Constants are divided into string constants and numeric constants and they just represent a data value.

3. **Special characters:** Special characters contain math and logic operators (such as +, -, *, I), relational operator (such as >, >=, <, <=, =), separator (such as", “,), brackets (such as (,), {,}). These characters have a special meaning, it will establish the Token attribute for special characters in the process of syntax analysis. According to the Token attribute to determine the meaning of the corresponding block.

4. **Database system Tag:** The database system provides users with the completion of a specific function stored procedure or system function. However, there are differences in the types of stored procedures and functions in different databases. The reason is the product's function and syntax definition habits during the development of the different data. So we need to

consider the system tags and symbols in different database when the Token is defined and designed.

3. Background on SQL injection

SQL injections is one of the most web attack mechanism used by different hackers to hack data from various firms and companies. If it happens against the information systems of a university, the private information [2] of the faculties may be leaked out which could threaten their reputation or may be a case of defamation. These attacks not only make the attacker to affect the security and steal the whole content of the database but also, to make optional changes to both the database schema and the contents of the database.

3.1 What is SQL injection?

Most web applications today use a multi-tier design, usually with three tiers: a) a presentation tier (front end). This is the topmost level of the application. This tier displays information related to such services as browsing merchandise, purchasing, and shopping cart contents. It communicates with other tiers by outputting results to the browser/client tier and all other tiers in the network. b) Application tier (Middle tier). This tier implements the software functionality by performing detailed processing, and c) the data tier (Backend). This tier consists of database servers, keeps data structured and answers to request from the application tiers.

Three-tier is a client server architecture in which the user interface, functional process logic, data storage and access are developed and maintained as independent modules, most often on separate platforms. SQL injection is a type of attack which the attacker adds Structured Query Language code to input box of a web form to gain access or make changes to data. SQL injection vulnerability allows an attacker to flow commands directly to web applications underlying database and destroy functionality or confidentiality.

3.2 Types of vulnerabilities

In this section, we present the most common security vulnerabilities found in web programming languages [3, 4] (see Table 1).

Table 1: Types of Vulnerabilities

Vulnerability types	Description
Type I	Input validation is an attempt to verify or filter any input for malicious behavior. Insufficient input validation will allow code to be executed without proper verification of its intention. Attacker taking advantages of insufficient input validation can utilize malicious code to conduct attacks.
Type II	Lack of clear distinction between data types accepted as input in the programming language used for the web application development.
Type III	Delay of operations analysis till the runtime phase where the current variables are considered rather than source code expressions.

The SQL injection attacks can be performed using various techniques. Some of them are specified as follows: Tautologies: The main goal of tautology-based attack is to inject code in conditional statements so that they are always evaluated as true. Using tautologies, the attacker wishes to either bypass user authentication or insert inject-able parameters or extract data from the database. A typical SQL tautology has the form, where the comparison expression uses one or more relational operators to compare operands and generate an always true condition. Bypassing authentication page and fetching data is the most common example of this kind of attack. In this type of injection, the attacker exploits an inject-able field contained in the where clause of query. He transforms this conditional query into a tautology and hence causes all the rows in the database table targeted by the query to be returned. For example, select * from user where id='2' or '2=2'-'AND password='1435'; "or 2=2" is the most commonly known tautology. Logically incorrect query attacks: The main objective of the Illegal/Logically Incorrect Queries based SQL Attacks is to get the information about the back end database of the Web Application. When a query is rejected, an error message is returned from the database including useful debugging information. This error messages help attacker to find vulnerable parameters in the application and consequently database of the application. In fact attacker injects junk input or SQL tokens in query to produce syntax error, type mismatches, or logical error by purpose. In this example attacker makes a type mismatch error by injecting the following text into the input field:

1. Original URL:http://www.toolsmarket-al.com/veglat/?id_nav=2234
2. SQL Injection: http://www.toolsmarket-al/veglat/?id_nav=2234'
3. Error message showed: SELECT name FROM Employee WHERE id=2234\'.

From the message error we can find out name of table and fields: name; Employee; id. By the gained information attacker can organize more strict attacks. The Illegal/Logically Incorrect Queries based SQL attack is considered as the basis step for all the other techniques.

Union Query: The main goal of the Union Query is to trick the database to return the results from a table different to the one intended. By this technique, attackers join injected query to the safe query by the word UNION and then can get data about other tables from the application. This technique is mainly used to bypass authentication and extract data. For example the query executed from the server is the following: Select Name, Phone from Users where Id=$id. By injecting the following Id value: $id =1 union all select credit Card Number, 1 from Credit sys Table. We will have the following query: select Name, Phone from Users where Id=1 union all select credit card Number, 1 FROM Credit sys Table. This will join the result of the original query with all the credit card users.

4. Comparative Analyses

In this section, the SQL injection detection and prevention techniques presented in section III would be compared. It is noticeable that this comparison is based on the articles not empirically experience.

4.1. Comparison of SQL Injection Detection Techniques With Respect to Attack Types

Detection techniques are techniques that detect attacks mostly at runtime. Table 1 shows a chart of the schemes and their detection capabilities against various SQL injections attacks and summarize the results of this comparison. The symbol √ is used for techniques that can successfully detect all attacks of that type. The symbol × is used for techniques that is not able detect all attacks of that type. The symbol □ refers to techniques that detect the attack type only partially because of natural limitations of underlying approach.

Table 2 illustrates the addressing percentage of SQL injection attacks among SQL injection detection techniques. The percentage of techniques that detect tautology is calculated by this formula (1):

$$\frac{\text{Number of techniques detects tautology}}{\text{Total number of techniques}} * 100 = 10/14 * 100 = 72$$

Table 2: Comparison of SQL injection detection techniques with respect to attack types

Attack Typs	Techniques that can detect all attacks of that type(√)	Techniques that can detect the attacks only partially (□)	Techniques that is not able to detect attacks of that type(×)
Tautologies	73%	13%	15%
Piggy-backed	63%	13%	21%
Illegal incorrect	63%	13%	23%
Union	63%	13%	22%
Stored Procedure	28%	13%	57%
Inference	73%	13%	15%
Alternate Encodings	55%	13%	29%

5. Design and implementation of second-order SQL injection prevention system

In this order of SQL injection, the attack payload stored in database is non-credible data which malicious user input, while the SQL fragment in Web applications is credible, but the existing DBMS cannot distinguish the trusted and untrusted part of a SQL statement. The second-order SQL injection defense mechanism proposed in this paper randomizes the SQL keywords in trusted constant string in Web application, which will create dynamically new sets of SQL keywords, and can distinguish from standard SQL keywords in attack payload which attacker inject and stored in database or file system. We can find the SQL injection attack by detecting whether the SQL statement includes the standard keywords. [9]. In the second-order SQL injection example above, the raw, non-randomized statement contained in the Web application is: $query=“update Users SET username=‘$username’, password=‘$password’, email=‘$ emal’ where usemame=‘$username’;”

The detection method of second-order SQL injection random 3 keywords in the above SQL statement: update, SET, FROM, for example if the current random number is 775, the SQL statement after randomization is: $query=“update 775 Users set 775 username=‘$ username’, password=‘$ password’, email=‘$emal’ where 775 usemame=‘$username’锊” Then attack payload stored in the email field of database is extract to construct the SQL statement. The full SQL statement is: update 785 Users set 775 username=‘XXX’, password = ‘XXX’, email =‘123’ where l=(updatexml (1 钏 concat (0x5e24, (select password from admin limit 1) 钏 0x5e24),1)); where775 username=‘XXX’ The update statement contains the standard keywords “select” and “where”, these standard keywords come from the malicious user, the second-order SQL injection defense mechanism proposed in this paper will detect the attack. The second-order SQL injection defense system is composed of randomization module, attack detection module and de-randomization module. Randomization modules append a random integer to SQL keywords in Web application, which creates dynamically SQL keyword sets, the attacker don’t know the new SQL keyword sets, so standard SQL which attacker injects can be detected, but at the same time, DBMS can’t recognize the new keywords, too, a solution to solve the problem is to modify the database interpreter, but this method is very complex. We adopt the method of building a proxy server between Web server and database server, as shown in Fig. 3, the proxy server has two main functions, one is to detect SQL injection attacks, the other is to de-randomize the harmless instructions to standard SQL statement and forward it to DBMS [10].

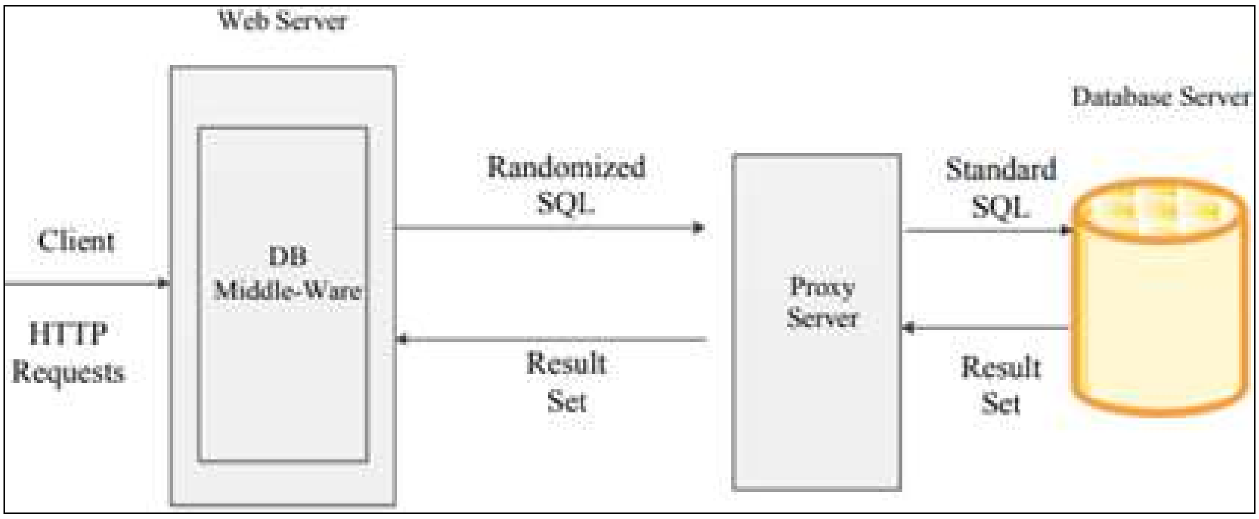

Fig 3: SQL injection detection system

6. Conclusions

In this review paper, we have presented a survey report on different types of SQL injection attacks, vulnerabilities, and detection and prevention techniques. We assessed SQL injection attacks among current SQL Injection detection and prevention techniques. To perform this evaluation, we first identified the various types of SQL injection attacks. Then SQL syntax analysis and SQL languages which constitutes the complete grammar rule as well as the lexical analysis which is the basis of syntax analysis and used to parse dynamic SQL progressive string into group of token tags. We also surveyed about four categories of word formations SQL statements in database systems, in section four we performed the comparison analysis and presented the results with respect to attack types [9].

Our future work will be performing systematic literature review where the shortages can be found through forming research questions, and to extend our research more on evaluation criteria.

7. References

1. The Open Web Application Security Project," OWASP TOP Project", https://www.owasp.org/SQL_Injection.
2. Bojken SH, Shqiponja A, Marin A, Aleksander XH, "Protection of Personal Data in Information Systems", International Journal of Computer Science, 2013, 10(2). ISSN (Online): 1694-0784.
3. Seixas N, Fonseca J, Vieira M, Madeira H. "Looking at Web Security Vulnerabilities from the Programming Language Perspective: A Field of Study", ISSRE 2009 pp.129-135.
4. Bojken S, Aleksander X. Comparative Analyses on SQL Injection: Vulnerabilities, Attacks and Techniques. International Journal of Computer Science. 2014; 11(4):28-31.
5. Dong Min. Research on The Attack Detection of SQL Injection Based on Dynamic Analysis. Beijing University of Technology, in Chinese, 2014.
6. Wang Hai-Yan, Yang He-Biao. ANTLR-based SQL Grammatical Analysis Strategy and Its Implementation. Computer Applications and Software, in Chinese, Nov. [II] LI Hai-Iong, ZHANG Wei-ming, 20I3; 30(ll):68-70.
7. Kuisheng W, Yan H. Detection Method of SQL injection Attack in Cloud Computing Environment. IEEE Computer Science & Engineering Conference. 2016; 10:1-5.
8. LI Hai-Iong, ZHANG Wei-Ming, LI Xi, XIAO Wei-Dong. Custom Standard SQL Grammar Analysis Model. Mini-Micro System, in Chinese. 2003; 24(II):1969-1972.
9. Chen P. A second-order SQL injection detection method. IEEE International Conference Software Engineering. 2017; 17(978):1792-1794.
10. Yan L, Li XH, Feng RT *et al.* Detection method of the second order SQL injection in Web applications J Lecture Notes in Computer Science. 2014; 8332:154-165.
11. Maor O, Shulman A. "SQL Injection Signatures Evasion (Whitepaper)," Imperva Inc, 04, 2004.
12. Dahse J. "Exploiting hard filtered SQL Injections (Whitepaper)," 03, 2010.
13. Luptk P. "Bypassing Web Application Firewalls," in Proceedings of 6th International Scientific Conference on Security and Protection of Information Czech Republic, 2011, 79-88.