Indian Journal of Science and Technology, *Vol 10(30), DOI: 10.17485/ijst/2017/v10i30/115513, August 2017*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Analysis of SQL Injection Detection and Prevention

Haripriya Rana and Shelly Sachdeva*

Department of Computer Science and Engineering, Jaypee Institute of Information Technology University, Sector-128, Noida – 201301, Uttar Pradesh, India; priyarana1606@gmail.com, shelly.sachdeva@jiit.ac.in

Abstract

Objectives: SQL Injection Attack (SQLIA) is a frequent and a severe security issue in the web applications. In SQLIA, hacker can obtain the benefit of poor input validation and weak coded web application. Due to the successful execution of a SQLIA, integrity and confidentiality of data are lost which results in the degrading organization's market value. This paper gives a valuable analysis of various types of SQLIAs, methods and mechanisms. It also explores various detection and prevention techniques. **Methods/Analysis:** A rigorous survey has been conducted and consequently, comparative analysis of various detection and prevention techniques is done with respect to various types of attacks. In current research various pattern matching algorithms for the detection and prevention of SQLIA are analyzed and few are tested. **Findings:** Comparative analysis of Boyer Moore pattern matching algorithm is done with Naive String pattern matching algorithm. The time and memory consumption taken by both the algorithms has been analyzed. The results show that Boyer Moore is more efficient to detect and prevent the SQLIAs as compared to Naive string. **Novelty/Improvement:** In future it aims to propose an algorithm which will enhance in terms of efficiency and resource usage. The approach needs to be implemented for every pattern matching algorithm to find the best solution regarding detection and prevention of SQLIA.

Keywords: Boyer Moore Algorithm, Detection, Prevention, SQLIA, SQL Injection

1. Introduction

The web applications are recommended by most of the organizations to have a worldwide access from anywhere. Web applications have many advantages, there are also many risks associated with them. There is a huge increase in the number of database attacks, which has made data security a big concern. Web applications have been targeted to access vital data of organizations. To accomplish this, hackers perform many types of attacks to breach the privacy. SQL Injection Attack (SQLIA)[1] is the major and common attacks performed by the attacker. It has turn out to be one of the serious threats. It has also been placed in top ten vulnerabilities of web applications[2]. Web applications run on different servers by using World Wide Web protocol. Poor input validation property of web applications makes it easy for hackers to attack. A malicious SQL query is inserted by the attacker into the web application appending it to the input parameter. Due to lack of strong input validation, SQL injection gets easily appended to the web application and it is executed on the database. Hence, malicious query is executed.

The current study summarizes various types of attacks, methods and mechanisms, and detection and prevention techniques. Figure 1 presents the flow chart of current research. It started with critical analysis of various types of SQLIAs. To detect and prevent SQLIA, various techniques are studied and analyzed. Consequently, a comparative analysis is done among various types of attacks, and techniques used to mitigate these attacks. It

*Author for correspondence

has been found that there is a need of an efficient algorithm to detect and prevent SQLIA. The paper analyzes various pattern matching algorithms for the detection and prevention of SQLIA. The Boyer Moore pattern matching algorithm and Naive String pattern matching algorithm are tested regarding time and memory consumption.

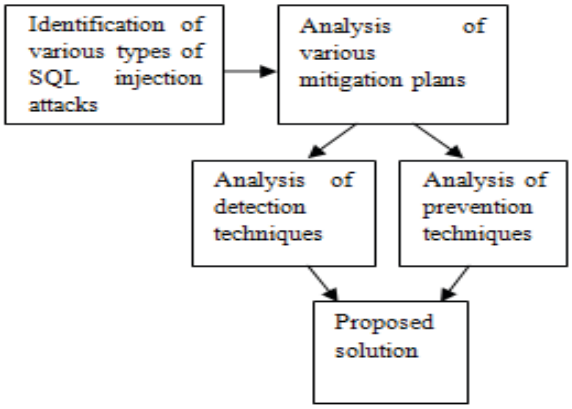

Figure 1. Flow chart of current research.

Section 2 details the categories and types of SQLIAs. Section 3 talks about various detection and prevention techniques. A comparative analysis is presented between detection and prevention approaches, and various types of SQLIAs in section 4. Various pattern matching algorithms for the detection and prevention of SQLIA are explored and few are tested. Finally, section 5 presents conclusions.

2. SQL Injection Attack (SQLIA)

In 1998, SQL injection was coined. Its first usage was recorded in 2000. SQL injection arises when the developer makes use of dynamic queries which are concatenated with the malicious variables by the hacker. In SQLIA[3], hacker can take the advantage of poor input validation and weak coded web applications. Due to the successful execution of SQLIA, integrity and confidentiality of data are lost which results in the loss of market value of an organization.

2.1 Reasons behind SQLIA

Nowadays, web applications are targeted the most due to its vulnerability to the attacks. There are so many reasons behind SQLIAs[4]. Due to the poor input validation property of web applications, flexibility property of SQL language and weakness in software of the web script, it gets easy for the attacker to perform SQLIA. Various causes of SQL injection are invalidated input, generous privileges, uncontrollable variable size, error message, client side only control, stored procedure, into out file support, and sub-select (inserting SQL query in WHERE clause of other SQL query). The major reason of SQLIA is the lack of strong input validation.

2.2 SQLIA Process

SQL injection is a kind of attack where attacker inserts malicious SQL query into the web application by adding it to the input parameters. During SQLIA, the malicious SQL query is inserted into the web application and it concatenates itself to the legitimate query. When this query is sent to the database management system for execution, the malicious query will be executed, and SQLIA will be performed. As a result, it can extract, modify and execute sensitive data in the database.

2.3 Categories of SQLIA

Categories of SQLIA describe various methods an attacker uses to execute the attack. SQLIA is divided into four categories, namely, SQL Manipulation, Code Injection, Function Call Injection, and Buffer Overflow. SQL manipulation is the procedure of using build in function to alter the SQL statements. Code injection is when attacker inserts new vulnerable SQL statements. Function call injection is the method of inserting an SQL function call into vulnerable SQL statements which can cause data manipulation. Buffer overflow is caused when a server is unpatched, and then by using function calls statement.

2.4 Mechanisms of SQLIA

SQLIA works on various procedures and different goals. Mechanisms of SQLIA are divided into two types, namely injection mechanism and attack intent. Injection mechanism is the procedure which is used to inject malicious code into the applications. Attackers use different types of variables and sources to inject the malicious code. This procedure further includes many categories which are used to perform the attack. First type of injection mechanism is injection through user input where user inputs the malicious data into the web application through 'http', 'get' or 'post' method. Second method is injection through cookies, which includes build-

ing SQL queries by using cookies content. Third method is injection through server variable where http, network headers and environmental variables are used to perform the attack. Attack intent tells about the type and the goal of various attacks. It has further many types. Identifying inject able parameter is a mechanism which discovers about the vulnerable parameter in the database. Performing database fingerprinting discovers the type and version of database which the attacker is going to attack.

Determining the database schema is a procedure where attacker discovers the database schema. This type of mechanism shuts down the database to deny the user. Extracting data mechanism is the method where data values are extracted which is sensitive and highly desirable. Performing denial of service mechanism shuts down the database to deny the user. Detection evading is to evade from security mechanisms. Remote command execution includes execution of remote commands and stored procedure in the database.

2.5 Types of SQLIAs

Various types of SQLIAs are Tautologies, Piggy backed queries attack, Union attack, Illegal incorrect queries, Inference based attack, Alternate encodings and Stored procedures[5]. Table 1 summarizes various types of SQLIAs, type of categories (SQL Manipulation, Code Injection, Function Call Injection, and Buffer Overflow), description and their corresponding examples.

3. Detection and Prevention Techniques

This section presents various techniques to detect and prevent SQLIA. The techniques which are analyzed in current research are AMNESIA, SQLCHECK Approach, CANDID, Auto-mated Approach, Tautology Checker, SQLrand, SQLDOM, CSSE and WebSSari.

Table 1. Types of SQLIA

SNo	Type of Attack	Category	Working Method	Example
1	Tautologies	SQL manipulation	In tautology, malicious code is injected into the conditional statements so that it always evaluates to true.	select * from users where name= 'ram' OR '1' = '1'
2	Piggy backed queries attack	Code injection	In piggy backed queries attack, attacker injects an independent query to the legitimate query and it gets executed after the already run first query.	select * from users where username = "; INSERT INTO users VALUES('anything',1536')--
3	Union query	SQL manipulation Code injection	Union query basically joins two independent queries and attacker uses UNION to join these queries and extract the data from other table.	select * from users where username = "UNION SELECT SUM (USERNAME) from users BY user id having 1 = '1' and password = 'anything'.
4	Illegal incorrect queries	SQL manipulation	In Illegal incorrect queries attacker collects the information about the schema of database by injecting the wrong query into the web application due to which some error is generated and this error contains important information about the database.	select * from users where username = Having 1 = '1' and password = 'anything'
5	Inference based attack	Code injection Buffer overflow	In this type of attack useful information is inferred by asking the server some true false questions.	http://victim/listproducts. asp?cat=books' or '1' = '1'. select * from products where category = 'books' OR '1' = '1';
6	Alternate encodings	SQL manipulation	It aims to defend against the secure defensive coding and automated preventive mechanisms.	
7	Stored procedures	Code injection	Attacker executes built-in procedure using malicious SQL statements.	

AMNESIA technique detects SQL injection in the web application[6]. It is an automated mechanism in which the invalid queries are recognized before executing on the targeted database. This mechanism is based on the static approach and run time monitoring.

SQLCHECK Approach[7] presented a technique which used grammar called "SQLCHECK" to test, validate and sanitize query. In this approach, an algorithm is implemented with the help of SQLCHECK tool, and developer applies a check on input queries. Secret key is used during parsing to restrict a user input. Only legal query is examined during runtime. Queries rely on model specified by the developer of the SQLCHECK.

CANDID[8,9] presented a technique based on automatic change in the code and used a 'PREPARE' statement. Candidate evaluation approach is for the automatic prevention of SQLIA. It automatically extracts the structure of intended queries from the legal users for any input. The format of the legal query is checked against the invalid or malicious query. It dynamically constructs the structure of the intended query when any issue is found.

Manual Approach[10] took the help of analysis and runtime testing to detect the input vulnerabilities and presented a technique which is implemented in SQL injection Gen. This technique is used to prevent the SQLI input manipulation flaws.

Automated Approach[10] highlight the use of automated approaches. There are two main schemes. First one is static analysis to find bugs approach. It detects bugs in SQLIA and gives warning when finding any malicious query or malicious variable. Second one is web vulnerability scanning in which software agent scans the web application and detects the vulnerability.

Tautology checker. In[11] presented a technique using static analysis and automated reasoning in order to guarantee that the input does not contain a tautology. This approach uses static analysis framework to detect the SQLIAs. There are mainly three steps which detect the SQLIA (abstract model, syntactic structure and security checking).

SQLrand[12] presented the concept of Instruction Set Randomization (ISR). ISR is an approach which protects the systems against the code injection attacks. Randomization algorithm is used in this approach and uses a key to randomize the instructions. This key can't be predicted by the attacker. SQLrand is a system for preventing SQLIAs against web servers.

SQL DOM[13] introduced Sqldomgen which generates dynamic link library which consists of strongly typed classes. These classes are known as SQLDOM. Using these classes dynamic SQL statements are constructed without changes in the string. These SQL statements are checked at the compilation time due to which there is no data mismatch. Hence, reliability is increased.

CSSE[14] proposed an automatic procedure which works with PHP language and stops SQLIAs by using a channel that consists of an assignment of metadata to user input and data preserving string operation and context sensitive string evaluation.

WEBSSARI. WebSSARI stands for Web application Security by Static Analysis and Runtime Inspection. An automated tool is used in this approach which works on the basis of certified input. This input is certified by passing through the filter. It uses static analysis. In event of exposure to threat it gives an alert[15].

4. Comparative Analysis

A comparative analysis is done between detection and prevention approaches and various types of SQLIAs.

Table 2 shows the comparative analysis between various techniques (AMNESIA, SQLCHECK, CANDID, Automated approach, Tautology checker, SQLrand, SQLDOM, CSSE, WebSSARI) and various types of attacks (tautology, logically incorrect queries, union query, stored procedure, piggy backed queries, inference attack, alternate encodings). '√' sign denotes if technique is able to detect or prevent the attack. '×' sign denotes if technique is not able to detect or prevent the attack.

From the comparative analysis it has been found that WebSSari technique is able to prevent all types of SQLIAs. WebSSARI technique is thus considered to be the best technique among all the approaches. CANDID and tautology checker only prevent one type of attack (i.e. tautology) as shown in table. Puneet analyzes the detection and prevention using classical methods and modern approaches[16]. A recent study also compares various detection and prevention techniques for SQLIAs[17]. It concludes that SQL IDS, SQL Checker and SQL Prevent can detect all type of SQLIAs (such as tautology, piggy backed, illegal/incorrect, union, alternate encoding, timing attack, blind attack, and stored procedure).

Table 2. Comparative analysis among detection, prevention techniques and types of attacks

Types of Attacks → / Detection and Prevention Techniques ↓	Tautology	Logically incorrect queries	Union query	Stored procedure	Piggy backed queries	Inference attack	Alternate encodings
AMNESIA	√	√	√	×	√	√	√
SQLCHECK	√	√	√	×	√	√	√
CANDID	√	×	×	×	×	×	×
Automated approach	√	√	√	×	√	√	×
Tautology checker	√	×	×	×	×	×	×
SQLrand	√	×	√	×	√	√	×
SQLDOM	√	√	√	×	√	√	√
CSSE	√	√	√		×	√	√
WebSSARI	√	√	√	√	√	√	√

√ This sign denotes if technique is able to detect or prevent the attack.
× This sign denotes if technique is not able to detect or prevent the attack.

SQLIA detection can be done by checking anomalous SQL query structure using string matching, pattern matching and query processing.

4.1 Pattern Matching Algorithms for Detection and Prevention

In current research various pattern matching algorithms for the detection and prevention of SQLIA are explored.

In pattern matching algorithm[18], text T is sequence of characters.

Σ is set of alphabets used in a string.

Pattern P is a string which is of length m. Prefix of P is substring of p[0.....k] and the suffix of P is substring of type P[K.....M-1].

The problem of pattern matching is to determine the substring of T equal to P. Various algorithms use different methods to find the exact pattern in the given text string. A recent study compares four pattern matching algorithms namely Brute-force, RabinKarp[19], Boyer-Moore and Knuth-Morris-Pratt for intrusion detection and prevention systems[20]. From practical viewpoint, the result shows that Knuth-Morris-Pratt and Boyer-Moore algorithms are efficient. Boyer Moore is the one of the efficient algorithm among existing pattern matching algorithms[21].

Naive String matching algorithm uses constant space and searches the pattern character by character. This algorithm compares every character in T with first character in P and then with the second character, and so on, until the exact matching takes place. We choose to test Boyer Moore and Naive string matching algorithms to detect and prevent the SQLIA efficiently. Prabakar et.al have used Aho-Corasick pattern matching algorithm for SQLIA detection and prevention[22]. It runs in linear time for multiple pattern matching strings, but requires large memory to store the finite state machine while practical implementation. SQLIA can also be prevented using hashing and encryption[23].

4.2 Architecture used for Detecting SQLIA

Figure 2 shows the architecture (adapted from[21]) to detect and prevent SQLIA. The proposed scheme comprises of two phases. First phase is static phase, and the second phase is dynamic phase. The static phase checks user generated query with static pattern list, whereas the dynamic phase analyzes query manually according to anomaly score value and then correspondingly updates static pattern list if required.

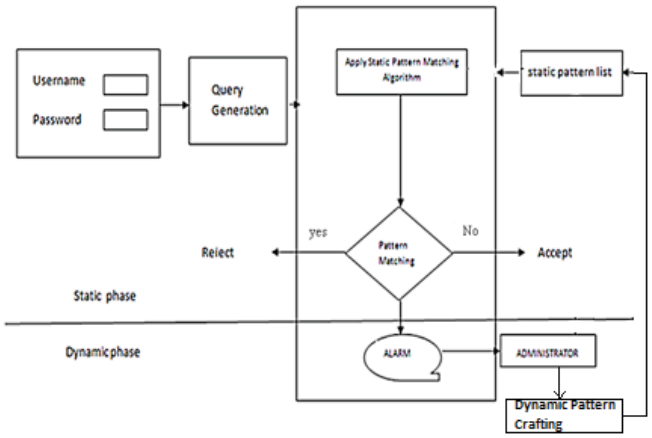

Figure 2. Architecture used for detecting SQLIA (adapted from[21]).

A list of known anomaly patterns is maintained in the static pattern list. In static phase, the user generated SQL queries are checked against malicious SQL statements by applying Static Pattern Matching Algorithm. Pattern matching finds the first occurrence of a given pattern in a sequence of a given text only if the pattern and text exactly match. Boyer Moore is considered as the most efficient string matching algorithm. It has been used to detect any type of attack in intrusion detection system. It increases the speed of intrusion detection system. It is a signature based detection algorithm and the system takes 30% to 60% processing time over this algorithm[24]. The performance depends upon choosing the accurate and better pattern matching algorithm. It works on two techniques. First technique is bad character heuristic. The pattern can be aligned to the alphabet if the alphabet does not match somewhere in the pattern. Second technique is good suffix heuristic which works as a pre-processing function. In this approach, if suffix matches the pattern, then the pattern can be moved until the next occurrence of the suffix in the pattern is aligned to alphabets same as the suffix.

The second phase is dynamic phase. In dynamic phase, if there is an occurrence of new anomaly, then an alarm is indicated and new anomaly pattern is produced. The administrator updates this new anomaly pattern to the static pattern list.

Following steps are generated during Static and Dynamic Phase:

4.2.1 Static Phase

Step 1: SQL query produced by the user is sent to the static pattern matching algorithms.

Step 2: In static pattern matching, Boyer Moore string is employed. Boyer Moore will scan the string character by character from the left to right. The current research also tested Naive string matching algorithm[25] for comparative analysis.

Step 3: Static pattern list maintains all the anomaly patterns. During pattern matching process, each pattern is compared within the stored anomaly pattern in the static pattern list.

Step 4: The SQL Query is affected with SQLIA if the pattern matches precisely with the stored anomaly pattern in the list.

4.2.2 Dynamic Phase

If the pattern does not match, then this scheme calculates anomaly score value of the user generated query for each pattern in the static pattern list. If this value is greater than the threshold value, an alarm is generated and the query is transferred to the administrator. The administrator analyses query manually to check whether it is affected by SQLIA or not. In case it affects, the pattern is generated and appended to the static pattern list as shown in Figure 2.

4.3 Testing

By using ASP.NET, tester and debugger are defined in different classes for validating the queries that cause vulnerabilities to occur. We considered different possible patterns which could be used by the attacker for SQLIA. These patterns are stored in MYSQL database by the developer. The detection of queries is done by comparing each word with the anomaly pattern stored in the database and if malicious variable, or query is found, an alarm is generated. This procedure employs Boyer Moore Algorithm. Figure 3 shows the implementation of Boyer Moore algorithm to detect and prevent SQLIA. Figure 4 displays the analysis result of the algorithm. Comparison analysis has been done to check the efficiency of the Boyer Moore Algorithm. To perform this analysis, we implemented Naive String matching algorithm to detect and prevent the SQLIA. The next step is to analyze the time and memory consumption taken by both the algorithms to detect same query during SQLIA. Figure 5 shows the

implementation of Naive string matching algorithm to detect and prevent the same query detected in Boyer Moore algorithm during SQLIA.

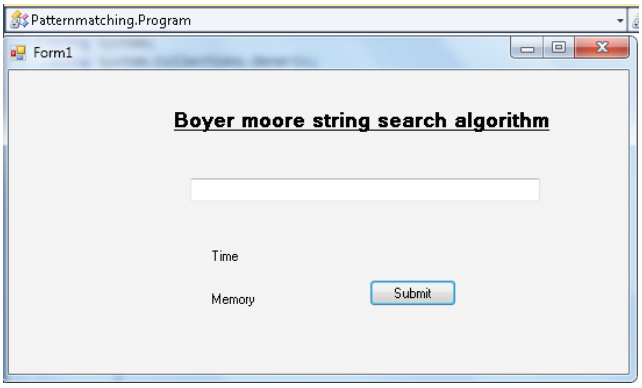

Figure 3. Home page for Boyer Moore implementation.

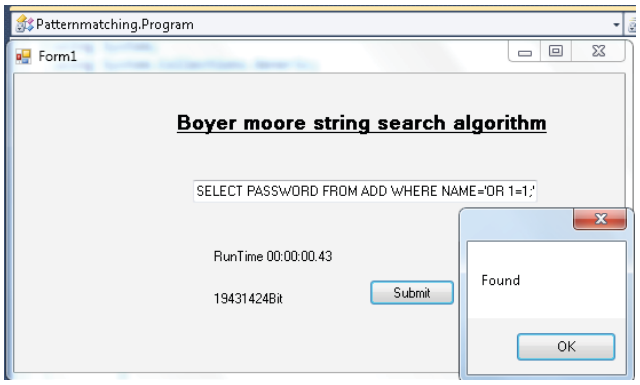

Figure 4. Results of Boyer Moore algorithm.

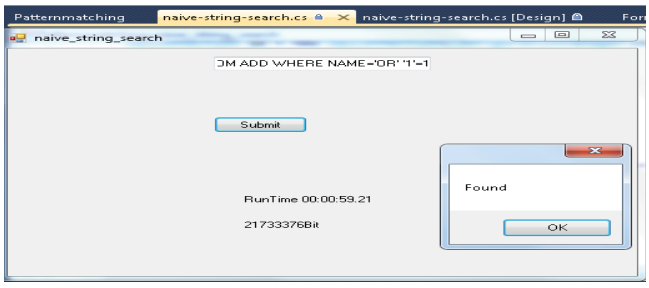

Figure 5. Results of naive string algorithm.

Figure 4 and Figure 5 show that time and memory consumption of Boyer Moore is less in comparison to Naive String matching to detect the same malicious query by using same approach. Hence, Boyer Moore is efficient approach as compared to Naive String matching algorithm.

5. Conclusions

SQL Injection is a severe security concern over web applications. In SQLIA, hackers take advantage of poor input validation and poor coded web application to attack the organization's systems. The current research provides a critical analysis of various types of SQLIA. To detect and prevent SQLIA, various techniques are studied and analyzed. On the basis of this research a comparative analysis is done between various types of attacks and techniques used to mitigate these attacks. With an aim to remove major types of SQLIA, various pattern matching algorithms (Brute-force, RabinKarp, Boyer-Moore, Knuth-Morris-Pratt and Aho-Corasick) are explored to detect and prevent these attacks. Boyer Moore Pattern Matching algorithm is investigated. This technique can work on many types of databases and it can be easily implemented on any platform. It works against different types of SQLIA, but puts a little overhead while dealing with large dataset. In future we aim to propose an algorithm which will enhance in terms of efficiency and resource usage. The approach needs to be implemented for every pattern matching algorithm to find the best solution regarding detection and prevention of SQLIA.

6. References

1. Balasundaram I, Ramaraj E. An approach to detect and prevent SQL injection attacks in database using web service. International Journal of Computer Science and Network Security (IJCSNS). 2011 Jan; 11(1):197–205.
2. Top 10 2010-Main, OWASP top 10 application security risks – 2010 [Internet]. 2010 [updated 2016 Dec 12; cited 2010 Apr 26]. Available from: Crossref.
3. Anley C. Advanced SQL injection In SQL server applications. Next Generation Security Software Ltd; 2002. p. 1–25.
4. McDonald S. SQL injection: modes of attack, defense, and why it matters. GovernmentSecurity.org; 2002 Apr. p. 1–32.
5. Medhane MHASP. Efficient solution for SQL injection attack detection and prevention. International Journal of Soft Computing and Engineering (IJSCE). 2013 Mar; 3(1):395–8.
6. William GJH, Orso A. Preventing SQL injection attacks using AMNESIA. In the Proceedings of the Association for Computing Machinery (ACM) 28th international Conference on Software Engineering, Shanghai, China; 2006 May 20–28. p. 795–8.
7. Su Z., Wassermann G. The essence of command injection attacks in web applications. In the Proceedings of the 33rd

Association for Computing Machinery (ACM) SIGPLAN-SIGACT Symposium on Principles of Programming Languages, USA; 2006 Jan 11–13. p. 372–82. Crossref Crossref
8. Bisht P, Madhusudan P, Venkatakrishnan VN. CANDID: dynamic candidate evaluations for automatic prevention of SQL injection attack. Association for Computing Machinery (ACM) Transactions on Information and System Security (TISSEC). 2010 Feb; 13(2):1–39. Crossref.
9. Bandhakavi S, Bisht P, Madhusudan P, Venkatakrishnan VN. CANDID: Preventing SQL Injection Attacks using Dynamic Candidate Evaluations. In the Proceedings of the 14th Association for Computing Machinery (ACM) Conference on Computer and Communications Security, Virginia, USA; 2007 Oct 28–31. p. 12–24.
10. Junjin M. An approach for SQL injection vulnerability detection. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) 6th International Conference on Information Technology: New Generations; 2009 Apr 27–29. p. 1411–4. Crossref.
11. Wassermann G, Suo Z. An analysis framework for security in web applications. In the Proceedings of the FSE Workshop on Specification and Verification of Component-Based Systems (SAVCBS); 2004. p. 70–8.
12. Boyd SW, Keromytis AD. SQLrand: preventing SQL injection attacks. In the Proceedings of the 2nd International Conference on Applied Cryptography and Network Security, Lecture Notes, Springer. 2004 Jun; 3089:292–302. Crossref.
13. McClure RA, Kruger IH. SQL DOM: compile time checking of dynamic SQL statements. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) 27th International Conference on Software Engineering, Saint Louis, USA; 2005 May 15–21. p. 88–96.
14. Pietraszek T, Berghe CV. Defending against injection attacks through context-sensitive string evaluation. International Workshop on Recent Advances in Intrusion Detection, Lecture Notes, Springer. 2005; 3858:124–45.
15. Huang YW, Yu F, Hang C, Tsai C-H, Lee D-T, Kuo S-Y. Securing web application code by static analysis and runtime protection. In the Proceedings of the Association for Computing Machinery (ACM) 13th International World Wide Web Conference (WWW), New York, USA; 2004 May 17–20. p. 40–52.
16. Singh JP. Analysis of SQL injection detection techniques [Internet]. 2016 [updated 2016 Dec 15; cited 2016 May 9]. Available from: Crossref.
17. Dehariya H, Shukla PK, Ahirwar M. A survey on detection and prevention techniques of SQL injection attacks. International Journal of Computer Applications. 2016 Mar; 137(5):9–15. Crossref.
18. Corman TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. 3rd edition, The MIT Press; 2009. p. 1–9.
19. Karp MR, Rabin MO. Efficient randomized pattern-matching algorithms. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) IBM Journal of Research and Development. 1987 Mar; 31(2):249–60. Crossref.
20. Gupta V, Singh M, Bhalla VK. Pattern matching algorithms for intrusion detection and prevention system: a comparative analysis. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) International Conference on Advances in Computing, Communications and Informatics (ICACCI), New Dehli, India; 2014 Sep 24–27. p. 50–4.
21. Boyer RS, Moore JS. A fast string searching algorithm. In Communications of the Association for Computing Machinery (ACM). 1977 Oct; 20(10):762–72.
22. Prabakar MA, Karthikeyan M, Marimuthu K. An efficient technique for preventing SQL injection attack using pattern matching algorithm. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICE-CCN). 2013 Mar 25–26. p. 503–06. Crossref.
23. Dinu PS. Preventing SQL injection attacks using cryptography methods. International Journal of Scientific Research Engineering and Technology (IJSRET). 2015 May; 4(5):582–5.
24. Knuth DE, Morris JH, Pratt VR. Fast pattern matching in strings. Society for Industrial and Applied Mathematics (SIAM) Journal on Computing. 1977; 6(2):323–50. Crossref.
25. Marc GOU. Algorithms for string matching [Internet]. 2014 [updated 2016 Dec 12; cited 2014 Jul 30]. Available from: Crossref.