FUDMA Journal of Sciences (FJS)
ISSN online: 2616-1370
ISSN print: 2645 - 2944
Vol. 8 No. 2, April, 2024, pp 241 - 246
DOI: https://doi.org/10.33003/fjs-2024-0802-2308

A CRITICAL EVALUATION OF SECURITY APPROACHES FOR DETECTION AND PREVENTION OF SQL INJECTION ATTACKS IN WEB-BASED APPLICATIONS

Yusuf Bukar Maina

Department of Computer Science Yobe State University, P.M.B 1144 Damaturu

*Corresponding authors' email: yusufbukarmaina1@gmail.com Phone: +2347036359042

ABSTRACT

SQL Injection Attack (SQLIAs) is a web application attack that has been known for almost two decades, and that has been among the biggest cyber threats especially because most of the world's population interacts with web apps in one way or the other. Over the years many methods have been developed to identify and deter SQLIAs, thereby reducing the risk on web applications. Four various methods used to identify and stop SQLIAs are reviewed, compared and critically evaluated in this paper, these include tokenization and lexicon detection process, combined static and dynamic method, novel, and search-based methods. This work further reveals the gap in current knowledge, specifically, increased efficiency can be achieved by integrating two of the most effective approaches. Furthermore, a real-world application of these methods is presented and finally, recommendations are made for further study.

Keywords: SQL Injection, Web Application, XSS Vulnerability, Web Security

INTRODUCTION

Internet and web application are fast becoming a household technology. So, protecting the privacy of user's information in web application have become more and more challenging. Consequently, web apps have been, and are still open to malicious attacks perpetrated by hackers. The popularity of web applications in social networking, financial transactions and health problems is growing very quickly, thus, website security has now become a main concern. There is a need to proactively detect and prevent SQLIAs. It is worth noting that, the security loopholes are mainly app layer vulnerabilities, such as XSS and SQL injection by Santhosh, et al (2018).

Several researchers have been working on a broad variety of SQLIA prevention areas, such as data encryption algorithm, PHP escape function, pattern matching algorithms, and set randomization. For instance, research conducted by Temeiza, et al. (2017), SHA-1 hashing algorithm was used to avoid batch query SQL injections. Ghafarian (2017) proposed a hybrid algorithm which attempts SQLIA detection by considering three phases of the web application life cycle; common gateway interface (CGI), implementation and database design. The algorithm operates by extracting query attribute values from stored inputs, and prior to that, the values have been hashed using the SHA-1 algorithm. Summarily, before the execution of all inputs, they will first be hashed and compared to a stored database of hashed data.

Zar Chi Su & Myo (2020) also proposed a technique of lexicon and tokenization for detecting SQLIA. In this strategy, wrong inputs are not processed explicitly. This is because each input is tokenized and compared to a predefined lexicon. Thus, malicious and normal queries are easily identified.

Voitovych, et al. (2016) and Liu, et al. (2019) approached the problem by developing a software tool that incorporates a server-side language (PHP), formal language theory (regular expressions) and client-side language i.e. JavaScript.

Rana, et al. (2017) proposed a method based on the hierarchical analyzer model. The framework receives a request from the client and assesses it with the help of a knowledge base, thereby categorizing the request as one that is for pages with no loopholes (P') and that for pages with weaknesses (P). Gu, et al. (2020) developed a framework that monitors the network traffic (to-and-fro), by leveraging the computational power of a GPU, they conduct check a huge regular expressions' dictionary. They claimed to be able to give prompt warnings thereby preventing SQLIAs.

The aim of this paper is to review, critically evaluate and analyze and also present different methods that have been proposed by researchers, in attempt to detect as well as prevent SQLIAs. Towards this end, this paper details, methods and techniques used, conclusions and claims reached as well as results presented by state-of-the-art literature. Subsequently, recommendation is made at the end of the paper in order to guide future work as well as the development of the improved solution.

This paper shall be divided into sections as follows: Section 2; evaluation of current methods, section 3; comparison of current SQL injection methods, section 4; recommendation and section 5; conclusion.

Evaluation of the Current Method of SQL Injection Detection and Protection

This section presents an evaluation of current detection and prevention methods.

Tokenization and Lexicon Detection Method

Following the section that presents an evaluation of tokenization and lexicon detection which detect SQLIA by tokenization characteristic.

Zar Chi Su & Myo (2020) proposed a strategy based on the sanitization of the query statement, consisting of two steps: the first step is the tokenization of the input query by the recipient. The tokenization method is achieved by finding white space, a double dash (-), a sharp sign (#) and all strings before each symbol is a token. In the second stage, each string token with the contents of a predefined lexicon is described. The contents of the lexicon are primarily words (commands) and some logical operators that are reserved. The outcome of some query injected, and possible attack being detected are shown in the table 1 below.

Table 1: Outcomes of Query Statements (Zar Chi Su & Myo, 2020)

	Prediction - normal	Prediction - injection	Total
Actual - normal	3	0	3
Actual - injection	0	7	7
Total	3	7	10

Table 2: Experiment Outcomes (Zar Chi Su & Myo, 2020)

SQLIA Techniques	Proposed approach's outcomes
Tautologies	Successful prevention
Malformed queries	Successful prevention
Union queries	Successful prevention
Piggy-back queries	Successful prevention
Inference	Successful prevention
Stored procedure	Successful prevention

The method was tested in a controlled experiment, in an isolated environment with 10 samples of SQL injection query phrases.

There are three usual statements and seven injection statements. Results are listed in Table 1 above. The method used shows an overall accuracy of false positive of 0 and true negative of 7.

According to the researchers, the proposed method of prevention and detection of SQL injection is efficient and successful for prevention of various malicious SQL injections which include tautologies, malformed queries, union queries, piggy-back queries, inference and stored procedure.

Critically evaluating the proposed, it has been observed that the method wasn't thoroughly tested, specifically, only six types of SQL injection attacks were evaluated. Summarily, the number of sample queries used were also inadequate.

To make a bold statement about the efficacy of the proposed technique, there is a need for more thorough evaluation.

The obvious strength of the proposed is based on the results presented, the results have shown that

the method is effective in protecting some types of SQLIA against the code with the reserved words-based lexicon. However, the sample size could be expanded.

Combined Static and Dynamic Method

Ghafarian (2017) proposed a solution to prevention and exploration of SQLIA which consists of three stages.

The first process proposed that all database tables should be augmented to contain a record include only symbols such as the dollar sign seen in Table 3 & 4.

The second step suggests an algorithm whose job is to dynamically process and track the execution of all incoming queries.

Any user query can go through this algorithm before execution is granted or denied, and the third step uses the algorithm to execute a string- matching procedure between the obtained SQL queries and the previously planned SQL queries.

Table 3: With augmentation (Ghafarian, 2017)

ProductID	ProductName	Constructor
101	%$%	%$%
102	Pen	Stabilo
103	Pencil	Steadtler
104	Eraser	Artline

Table 4: The comparison of the proposed method with different methods (Ghafarian, 2017)

Feature	SQLIF	LINQ	PSR	Proposed
Dynamic	yes	yes	No	yes
Tautology	yes	yes	No	yes
Illegal/Query	yes	No	No	yes
Union	yes	No	No	yes
SP	yes	yes	No	yes
N. limit input query	yes	yes	No	yes
Plat.indepedent	yes	yes	No	yes

Ghafarian (2017) concluded that the existing methods of identification and prevention of SQLIA was contrasted with the proposed method. The suggested solution is more effective since it can accommodate all forms of queries and the algorithm is platform-independent.

Interestingly, the method was tested with many samples of SQLIA types which is quite good. But the drawback is that there is no evidence showing a sample of queries. and system used in the experiment. And this experiment could not be considered as repeatable.

Therefore, more is required in the experiment before the validity of the result and claims made by the researcher can be verified.

Knuth-Morris-Pratt string matching

Oluwakemi, et al. (2020) proposed a novel method of prevention and detection of SQLIAs.

String Pattern of SQLIAs Formation

Each attack mode has some characters and keywords that are used by hackers to perpetrate their attacks, and these keywords and characters are used to produce malicious codes to perform different types of attacks. These are then stored in a database with a view to comparison to the users input. Sample of these are shown in tables 5 and 6.

Table 5: Keywords that are used to write SQL- injection code (Oluwakemi , et al., 2020)

S/N	Character	Description
1	'	Character string indicator
2	-- or #	Single line comment
3	/.../	Multiple line comment
4	%	Wildcard attribute indicator
5	;	Query terminator
6	+ or	String concatenate
7	=	Assignment operator
8	>, <, <=, >=, <> or !=	Comparison operators

Table 6: Different types of injection code with a standard pattern (Oluwakemi, et al., 2020)

S/N	Injection Type	Common Pattern	Example
	Boolean-based	'OR' > '	> =<| <=|= : ' -"#
	Union-based	'union select … from …;#'	'union select * from users;#' 'union select name from a;#'
	Error-Based	'…convert (|\avg(| round(…'	111' convert(int, abcd') A' avg(28%$#@")
	Batch query	'; drop | delete | insert | truncate | update | select…#'	aaa'; delete * from users; # ; drop table users;#
	Like-based	''OR … LIKE '…%';#''	\ ''OR username LIKE '%5%'#''
	\ XSS	\ <script> …</script>	<script>alert('Xss')</script>

Parse Tree Design of Various Attack

To reflect the syntactic structure of the different forms of SQL-Injection, a parse tree was used.

KMP Algorithm

A matching algorithm known as Knuth-Morris-Pratt (KMP) was used to compare the user input string with the stored patterns of SQL injection. PHP was use for scripting.

Filter Function

To avoid SQL injection, encoded injection and cross-site scripting (XSS) attacks, a filter function was formulated. These call external functions, each of which is written to detect a specific type of attack. If any of the stated three attacks is detected, the filter function blocks the user, resets the HTTP request, and shows the corresponding alert message.

Table 7: Comparison of the current and proposed method (Oluwakemi , et al., 2020)

Methodology	Ref.	Boolean-based SQLi	Union-based SQLi	Error-based SQLi	Batch query SQLi	Like-based SQLi	XSS Injection	Encoded Injection
Using pattern matching	[21]	✓	✓	✓	✓	✓	✓	×
	[22]	✓	✓	✓	✓	✓	✓	×
	[23]	✓	✓	✓	✓	✓	✓	×
	[24]	✓	✓	✓	✓	✓	✓	×
	[25]	✓	✓	✓	✓	✓	✓	×
	[26]	✓	✓	✓	✓	✓	✓	×
	[27]	✓	✓	✓	✓	✓	✓	×
Using data encryption	[28]	✓	✓	✓	✓	✓	×	✓
	[29]	✓	✓	✓	✓	✓	×	✓
	[30]	✓	✓	✓	✓	✓	×	×
ISR	[31]	✓	✓	✓	✓	✓	×	×
Proposed algorithm		✓	✓	✓	✓	✓	✓	✓

Oluwakemi, et al. (2020) conducted a controlled experiment in an isolated environment using Apache Server and Internet Information Server.

Experiments were conducted on computers and mobile phones with the most popular operating systems, Windows, Linux and Android. To extensively evaluate performance of the proposed algorithm, the hardware of the devices (i.e. microprocessor and RAM) were varied. The database used was a mysql database, and the attacks were tested on both local and remote servsers. Based on the results they presented as shown in Table 7, their technique was able to tackle encoded injections as well as Cross site scripting (XSS) effectively, this is in addition to the most common SQLIAs.

The researchers argued that their proposed solution showed superior performers as compared to three algorithms that were proposed by Abikoye, et al. (2019), Benfano, et al. (2018) and Das, et al. (2019). This they were able to show in the result they presented (as shown in Table 7) their technique obviously tackled all five forms of SQL injection attacks (SQLIA) and additional addressed XSS and encoded injection attacks.

Based on the detail provided by the authors, the experiments are apparently repeatable. Furthermore, their work shows a degree of platform independence since varying hardware and operating systems were used throughout the experiments.

Impressively, an acceptable number of SQLIAs were used. The method was tested on several SQL statements including but not limited to Boolean based, union-based, and error-

based. Moreover, it has been shown to be effective against even other types of attack (XSS and encoded injection).
Summarily, the result they presented show that a broad variety of attacks taken from random samples and devices were evaluated, and their findings show sufficient evidence to support the statements made.

A Searched-Based Method Using Novel Fitness Function
Liu, et al. (2019) suggested a technique to create a thorough set of test cases to evaluate the weak points of the system under test (SUT), including the front-end and back-end, that mimic potential user inputs.
Liu, et al. (2019) conducted controlled experiments in an on 19 various types of SQL statements and attack modes, based on three real- world 158 Of web applications.
The method used two Targerted SQL statements (TOs) are randomly chosen and the number of generations is plotted in the trajectories. The diagram in Figure1(b) shows that the algorithm converges more effectively using the Similarity Matching Distance (SMD) in

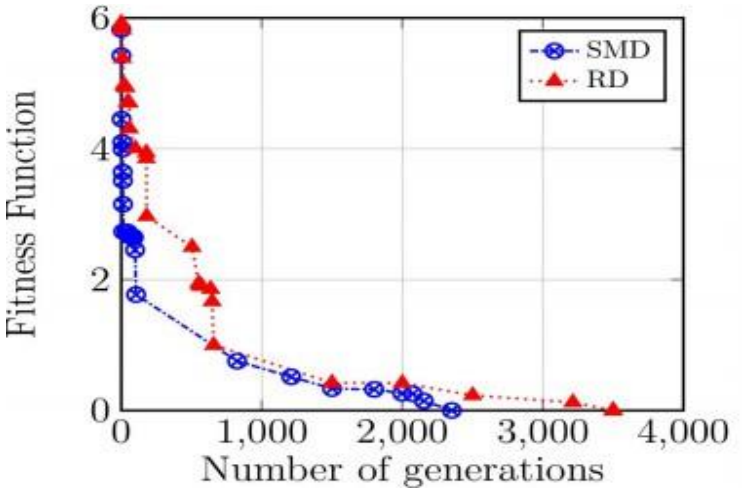

Figure 1(a) Showing Similarity Machine Distance (SMD).

The fact that the real-coded distance (RD) doesn't shift for a long period when it is close to 0. may explain this. Moreover, certain strings that are very close to each other can be distinguished by the SMD. Accelerating convergence is thus more efficient.

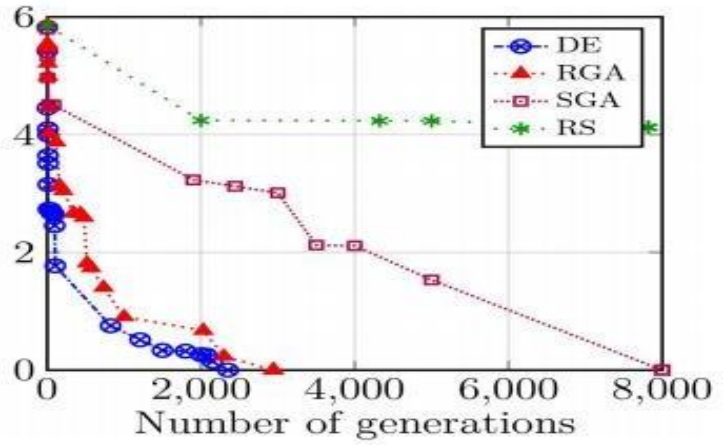

Figure 1(b) Showing Convergence rate with SMD and RD of a TO for DE (Liu, et al., 2019).

In Figure 1(b) above, researchers concluded and clarified that differential evolution (DE) in the search process is constantly higher than other peer algorithms. Currently, DE's trajectory is the steepest in contrast to the others. This implies that DE's convergence speed is the lowest, costing only about a quarter of the calculation budgets that are given to the limit.
The method was tested on various type of SQL queries and 158 adequate web application.
The result show presents no bias as the method and sample of an experiment conducted on a variety of adequate sample. and the experiment could be repeated. Therefore, the finding shows that sufficient evidence made to support the claim is made.

Comparison of Current SQL Injection Methods
All the experiments/research works critically reviewed in this paper were based on identifying and preventing web apps from SQL injection attacks. It is worth nothing that all the methods discussed in section 2 have their advantages and disadvantages. Hence, the aim of this section is compared and contrast with a view making a recommendation for the most fit approach.

Table 8: Showing Limitation and Strength of Current SQL Injection Methods

Comparison of Current SQL Injection Methods	Limitation	Strength
Tokenization and Lexicon	Tautologies, malformed queries, and union query	finding white space, a double dash (-), a sharp sign (#)
Combined Static and Dynamic	Data entry restriction	Tautology, Union and Illegal Query
Searched Based using Novel Fitness Function	Semantic of SQL statement	granted or denied Plat. Independent and SP
Knuth-Morris-Pratt string matching (Novel Method)	Inference type of SQL injection	XSS, Boolean-Based and Like-Based

The tokenization and lexicon detection method had advantage of detecting and preventing various SQLIAs including but not limited to, tautologies, malformed queries, and union query. As evidenced by the result presented in the work of Zar Chi Su & Myo (2020), the proposed approach is capable of detecting five types of SQLIAs, it further showed that it had a false positive of 0 and true negative of 7. This will most probably help web developers to protect data against tampering. One thing that is lacking is sufficient testing, hence, the number/size of queries tested need to be expanded.

The combined static and dynamic method also have the capability of protecting and preventing tautology, as reported by Zar Chi Su & Myo (2020). Although not as effective as tokenization and lexicon method, it does however, have the advantage of being platform independent. Additionally, it has no data entry restriction. Here also, the research experiment needs to be improved because the results presented were based on limited samples.

A searched-based method using novel fitness function had the advantage of protecting and detecting SQL user inputs and validate the vulnerability in front-end and back-end. Even though has the limitation of semantic of SQL statement. But is better than the other method analyzed as it shows a lower advantage in improving the protection and detection of SQL injection attack.

Knuth-Morris-Pratt string matching (Novel Method) had an advantage of the detecting and preventing various types of attack including tautology, union query malformed query attack and many more as presented by Zar Chi Su & Myo (2020) and Ghafarian (2017) and Liu, et al. (2019). Even though not as tokenization and lexicon method. As this method doesn't have the chance to protect the inference type of SQL injection attack which is prevented the inference vulnerability.

Based on the critical comparison of the methods, it is apparent that no method can be considered as the crème de la crème since all four methods had their pros and cons. Thus, this suggests the need for further research.

RECOMMENDATION

SQL injection Attack can be countered in front- end and back-end of web applications. Thus, an ideal method is one that is robust to all forms of attacks. Specifically, an ideal solution should be able to protect against tautology, union query, malformed query and inference attacks as they are the attacks encountered in both front-end and back-end. It is therefore necessary to investigate and implement such an ideal solution. One way to achieve a robust solution will be to create a hybrid of the existing algorithms. Hence, we recommend the use of the solution proposed by Oluwakemi, et al. (2020) and that of Liu, et al. (2019), creating a hybrid of these two will bring on board their abilities to detect and prevent various SQLIAs vulnerabilities while each method mitigates the limitation of the other method. Furthermore, another hybrid solution to investigate is a combination of the lexicon and tokenization approach on one side, and the Knuth-Morris-Pratt's based solution. We believe this will help to create a robust SQLIA preventive solution. Besides, creating the hybrid techniques, it is also crucial to get much testing sample, since it has been observed that insufficient experimentation does not reveal the strengths and weaknesses of a solution.

CONCLUSION

SQL injection attack a popular and malicious means of tempering with data on web applications, was critically reviewed in this paper. Initially, a literature review was conducted, from the numerous approaches proposed in the literature, four of the most recent
techniques for identifying and avoiding this awful risk of SQLIAs, were chosen.

After thorough comparison of the selected solutions, it was observed that there is no clear winner, since they all had their strengths and weaknesses. Hence, a hybrid solution was recommended this is believed to have the ability to have reduced limitations and much strength since the combined methods complement each other. Combining static & dynamic approach and a searched-based approach does have some advantages, however it is believed that a hybrid formed from three methods tokenization and lexicon approach, searched-based and novel method will be a superior method.

Finally, while creating a robust hybrid solution is worth investigating, critical review conducted in this paper also revealed other gaps and suggestion to future researchers. These include the need to have much testing data, bigger and more thorough experiments, as well as well documented and reproduceable research papers.

REFERENCES

Abikoye, O., Dokoro, H., Abubakar, A., Oluwatobi, A., & Asani, E.O., 2019, 'Modified Advanced Encryption Standard Algorithm for Information Security', *Symmetry*, Vol. 11, pages 1-16

Benfano , S., Fergyanto E. , G., Hirzi & Frumentius, 2018, 'Prevention Structured Query Language Injection Using Regular Expression and Escape String', *Procedia Computer Science*, Vol. 135, pages 678-687

Das, D., Sharma, U. & Bhattacharyya, D. K., 2019, 'Defeating SQL injection attack in authentication security: an experimental study', *International Journal of Information Security*, 18(1), pp. 1-22

Ghafarian, D. A., 2017, 'A Hybrid Method for Detection and Prevention of SQL Injection Attacks', *Computing Conference, London*, pages 833-838

Gu, H., Liu, T., Zhang, J., Hu, M., Zhou, J., Wei, T., Chen., & M., 2020, 'DIAVA: A Traffic-Based Framework for Detection of SQL Injection Attacks and Vulnerability Analysis of Leaked Data', *IEEE Transactions on Reliability*, Volume 69, pages 1-15

J. Santhosh Kumar, B. & P. Anaswara, P., 2018, 'Vulnerability detection and prevention of SQL injection', *International Journal of Engineering & Technology*, Vol. 7, pages 16-18

Liu, M., Li, K. & Chen, T., 2019, 'Security Testing of Web Applications: A Search-Based Approach for Detecting SQL Injection Vulnerabilities', *GECCO 19: Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 417-418

Oluwakemi, C. A., Abdullahi, A., Ahmed, H. D., Oluwatobi, N. A. & Aderonke, A. K., 2020,' A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris- Pratt string match algorithm', *EURASIP Journal on Information Security*, Vol. 2020, pages 1-14

Rana, M. N., Rana, M. S., Rabnawaz , B. & Sidra, H., 2017, 'Detection and Prevention of SQL Injection Attack by Dynamic Analyzer and Testing Model', *International Journal of Advanced Computer Science and Applications*, Vol. 8, pages 209-214

Temeiza, Q., Mohammad, T. & J., I., 2017, 'A novel method for preventing SQL injection using SHA-1 algorithm and syntax-awareness', *Sudan Journal of Computing and Geoinformatics*, Vol. 1, pages 16-26

Voitovych, O., Yuvkovetskyi, O. & Kupershtein, L., 2016, 'SQL injection prevention system', *International Conference Radio Electronics & Info Communications (UkrMiCo), Kiev*, pages 1- 4

Zar Chi Su, S. H. & Myo, K., 2020, 'A Detection and Prevention Technique on SQL Injection Attacks', *IEEE Conference on Computer Applications (ICCA)*, pages 1-6