Journal of Theoretical and Applied Information Technology

20th June 2015. Vol. 76 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REVIEW OF THE DEFENSIVE APPROACHES FOR STRUCTURED QUERY LANGUAGE INJECTION ATTACKS AND THEIR COUNTERMEASURES

[1] NABEEL SALIH ALI*, [2] ABDUL SAMAD SHIBGHATULLAH, [3] MUNQATH H. AL ATTAR

[1,2]Faculty of Information and Communication Technology, University Technical Malaysia Melaka, Malaysia, Melaka, [3]Information Technology Research and Development Center, University of KUFA.

[1]Nabeel@uokufa.edu.iq, [2]Samad@utem.edu.my, [3]Munqith.alattar@uokufa.ed.iq

ABSTRACT

Recently, Web applications have been used for most of the activities in animation. These applications are affected by the structured query language injection (SQLI). In this paper, four major objectives can be organized to direct the work study are:

- Conduct a detailed review of various SQLI attacks and investigation of previous approaches that detected and prevented these attacks in Web applications.
- Compare the performance metrics of the different techniques to evaluate the precision of the results and the cost of the time required to identify the efficiency of the techniques.
- Evaluate the effectiveness of the techniques in practices based on the effectiveness metrics.
- Define the efficiency and effectiveness direction of defensive approaches.

The main contributions of this work are:

- Summary and analysis of a critical review (strengths and weaknesses) of the defensive approaches that have been implemented.
- Comparison of the result accuracy of the different approaches through an evaluation using the standard performance metrics.
- Evaluation of the effectiveness of the techniques in practice.
- Identification and focus on the critical and important lines or defensive techniques that need comprehensive studies by future researchers through which the advantages of high efficiency and effectiveness can be obtained.

Keywords: *SQL, Injection, Detection, Prevention, Approaches. Techniques.*

1. INTRODUCTION

The recent increase in Web applications for online services via Web pages had led to the increase in the number of customers for their public access [1]. Therefore, Companies and Organizations have been constantly striving to enhance their communication capabilities by providing secure application levels to achieve the functionality that will allow them to build and maintain relationships with their stockholders [2]. Consequently, this creates a situation where an attacker can use the structured query language injection (SQLI) vulnerabilities to control and corrupt the target.

Web applications play a significant part in our life and in any country's evolution. Web applications typically interact with backend database to retrieve and present persistent data to the end user. Therefore, a loophole in an application's secure design may allow illegal access into the backend database via crafted injection and malicious update [3] . Thus, security has become one of the main challenges in the recent years because most of the Web applications have suffered from vulnerabilities that have made them attractive targets of security attacks [4]. Hence, this flaw can used by terrorists to collect private data and obtain illegal access to

* Corresponding Author H.F: +9647803896893 University of Kufa

the target [5]. SQL injection attack (SQLIA) is one of the most prevalent and dominant classes of serious Web application attacks [6]. SQLIA is a provides attackers opportunities to gain direct access to the database and extract sensitive information from the backend database [1].
SQL injection for online application is the legal access to the database. Unauthorized access to the current data by a crafted user causes threat to their Confidentiality, integrity, and authority. As a result, the system may bear significant loss in giving correct services to its users or face complete destruction [7]. SQLIA is categorized as one of the top 10 Web application vulnerabilities in 2013 experienced by Web applications according to Open Web Application Security Project (OWASP) [8].

SQLI refers to a class of code-injection attacks in which the data provided by the crafted user are included in the SQL query in such a way that part of the user's input is treated as an SQL code. The trick used is the injection this query or command as an input, possibly via the Web pages. SQLI is a one of the more general classes of vulnerabilities that could occur when a programming or scripting language is embedded inside another [9]. The attack occurs when data provided by user is not properly validated and when the data are included directly. The attack is a mechanism or technique that exploits a security weakness occurring in the database layer of an application [10]. The loophole is present when user input is not strongly typed and thereby unexpectedly executed or when user input is in correctly filtered for string literal escape characters embedded in the SQL statements [7].

In this paper, we present a detailed review and comparative analysis of the common (previous and existing) defensive approaches or techniques against SQLIAs, to give a unified view of proposed approaches as future reference for conduct of a comprehensive study, and highlight the need for further studies on the efficient and effective techniques or approaches.

2. PREVIOUS WORKS

Research authors proposed a wide range of techniques to address the SQLI problem. These techniques for detecting and preventing SQLIA range from filtering, information- flow analysis, penetration testing, development best practices and defensive coding to fully automated framework. Some techniques used to solve SQLIAs require security awareness of the user, which cannot be guaranteed. Moreover, some of the existing solutions are unacceptably slow and can be bypassed. Some are too restrictive, resulting in loss of functionality [7].
On the other hand, many authors that presented review or comparative analysis articles to investigate and evaluated SQLI detection and prevention tools or techniques as well as compared these techniques or tools in terms of the ability to address and stop SQLIAs. Relevant reviews or evaluations articles have been presented, such as the evaluation presented by Halfond et al. 2006. They presented a survey and comparison of proposed techniques for detecting and preventing SQLIAs. Identified the various types of SQLIAs and evaluated the techniques in terms of their ability to detect and/or prevent such attacks. As well as, studied the different mechanisms and identified which techniques could handle which mechanisms and summarized the deployment requirements of each technique, but did not focus on the evaluating the techniques precision and effectiveness in practices [11]. Rahul et al. 2012, presented a survey on different classes of SQLIA and some of the important approaches for detection of SQLIA but did not evaluate these approaches in terms of the ability to address the SQLIAs [12]. A survey on SQL Injection Attacks, detection and prevention techniques presented by Kumar and Pateriya, 2012 that presented different types of SQL injection attacks and their prevention techniques and conducted a comparative analysis of different types of detection and prevention techniques of SQL Injection attacks with respect to automation, code suggestions and generates a report, but did not an assessment the techniques based on the common evaluation parameters such as performance, efficiency and effectiveness [13]. Sankaran et al. 2014, presented the various attack methods, their classification using which the system administrators and programmers can understand about SQLIA and secure the web application [14].

Our work focuses on the approaches that have been employed by other researchers from 2005until 2014 to solve the SQLIA and highlights their strengths and weaknesses (critical review) to give a unified view of proposed approaches as future reference for conduct of a comprehensive study. Furthermore, our work focuses on the evaluation of the precision and effectiveness of defensive approaches via comparing the performance of the different techniques based on standard performance metrics (false positive, false negative and protocol overhead) and results accuracy to define the efficiency of the techniques, as well as evaluate these techniques in terms of the ability to address and stop the attacks of SQLI by comparing the techniques respect to which technique can detect and prevent the attacks and can stop all types of the attacks. Finally, identification of the area of the defensive approaches that need validation through more studies in the future.

3. SEARCH METHODS

The Method used to collect or search the data for this study was based on multiple sources. The first attempt was aimed at finding all the synonyms for the SQLIAs and using these synonyms as the search criteria. These criteria were used to find any conceivable material available.
The selected main keywords related to the study scope were SQLIA and detection and prevention SQLIAs. Searches on reliable databases, such as the Web of Science (ISI), ScienceDirect (Scopus), and IEEE Xplore (IEEE) from 2005 to 2014, were undertaken. After applying the filters, the searching engines derived 24 studies from the Web of Science (ISI), 118 studies from ScienceDirect (Scopus), and 12 studies from IEEE Xplore (IEEE). Subsequently, we selected 24 common studies as the basis for the conduct of our review and evaluation.

4. DEFENSIVE APPROACHES FOR (SQLI) ATTACKS

In general, there are several kinds of Web attacks such as: SQLI, cross- side scripting (XSS), remote command execution (RCE) and path traversal attacks. There are many ways to prevent SQLIAs and protect a Web application, such as defensive coding, information flow analysis, content filtering, and penetration testing. The prevention of SQLI concerns with correctness of input value which is supplied by the client or user at the coding level. There are two major concerns. One is the crucial need for a mechanism to detect and exactly identify SQLIAs. The other is necessary of knowledge of SQLI vulnerabilities (SQLIVs) to secure a Web application. Research authors have proposed methods and techniques to address SQLIAs. These include static analysis, dynamic analysis, combined static and dynamic analysis, Web framework, defensive programming and machine learning techniques. Some techniques could not address all SQLIA types whereas some have special deployment requirements. Further, some have not been implemented yet.

5. COMPARISON OF DEFENSIVE APPROACHES BASED ON CRITICAL REVIEW

The detailed critical review conceptually provides insights into the common (previous and existing) works.
These works, which have been conducted from 2005 until 2014, were aimed to determine the defense against the SQLIAs and highlight their weaknesses. The review intends to address the lack of knowledge, conduct a comparative analysis between previous works, and highlight the need for further studies on the efficient and effective techniques or defensive approaches.
In this section, we present a detailed critical review of the previous defensive approaches that have been implemented and highlight their strengths and weaknesses.

Table 1 Critical Review of the Defensive Techniques

Author and year	***Strength***	***Weakness***
Wei et al. 2005[15]	Uses static analysis of the stored procedure source as a one-time offline procedure via the form of a SQL-graph	Limited in terms of developing a complete implementation of the proposed architecture to extend the prototype
Buehrer et al. 2005 [16]	Efficient and effective.	Additional overheard computation and listing of input (black or white)
Halfond & Orso 2005 [17]	Efficient and effective. The techniques stopped all of SQL injection attacks without generating any false positives.	Requires the modification of the Web application's source code.

Diego et al. 2005 [18]	The approach checked the correct code written by the user at the compile level.	Possible undesirability of the technique to developers because it requires new programming paradigms.
Martin et al. 2005 [19]	Prevents SQL injection attacks by using static and dynamic checker	Difficulty in identifying all sources of user input is the main limitation of this approach.
Halfond et al. 2006 [20]	The technique is precise and efficient, and has minimal deployment requirements.	Did not use static analysis or the approach for binary applications
Pietraszek & Berghe 2006 [21]	Uses SQL token to detect and reject queries that have been constructed by illegitimate input.	Requires rewriting code
Kosuga et al. 2007 [22]	Performs comparison between the parse trees of SQL query and the results after an attack to assess the safety of these spots	Not efficient because it has a few false positives and 39 SQL injection vulnerabilities
Lu et al. 2007 [23]	SAFELI is capable of discovering very delicate vulnerabilities by taking advantage of source code information	Automatically enumerates SQL WHERE clauses by exploring algorithms and does not complete the implementation.
Halfond et al. 2008 [24]	Efficient and Effective in stopping more than 12,000 attacks without generating any false positives	Implements the approach for binary applications and deploys Web applications
Kemalis & Tzouramanis 2008 [25]	The query-specific detection approach is efficient because it stops attacks without producing false positives or false negatives.	Limited to identifying and checking sources and sinks are subject to input validation; flow-sensitive.
Ezumalai, 2009 [26]	Proposed approach reduces the time and space complexity.	Low execution overhead and requires no modification of the runtime system
Khoury et al. 2011 [27]	The approach cannot detect without exploiting the vulnerability.	The identified weaknesses of black-box scanners reside in many areas: crawling, input values, and attack code selection.
Yan et al. 2011 [28]	The system effectively guarantees the security of the database.	The proposed system could protect the common SQL attacks, but could not prohibit some rare attacks
Alserhani et al. 2011 [29]	Proposed model successfully generates security level in real time environment based on establishing a correlation between attack signatures.	Ignores the false positives detected; not concerned with huge data
Kim 2011 [30]	Can be used for detection program modularization, SQL query profiling, and SQL query listing during the implementation	Implementing in a machine learning algorithms is the lack in this method.
Shar & Tan 2012 [31]	Detects more than 80% of the vulnerabilities and provides an alternative and cheap way of mitigating common Web applications vulnerabilities	These models cannot enhance the prediction models with more precise dynamic analysis-based classification methods.
Balasundaram & Ramaraj 2012 [32]	Effective in stopping all the SQL injection attacks without generating any false positive	Protocol overhead with query based technique
Scholte et al. 2012 [33]	IPAAS protects real-world applications against SQL injection and XSS vulnerabilities.	Low false positive rate
Natarajan & Subramani 2012	Provides optimized runtime analysis and does not need further code modification	Does not consider the construction SQL parser and lacks dynamic checking complier

[34]		
Lee et al. 2012 [35]	Independent approach of the DBMS and does not need complex operations that are based on removal of the attribute values in SQL queries during the analysis	Does not apply this approach based on machine learning algorithms
Shar et al. 2013 [36]	The model results in 93% recall and 11% false alarm rate in predicting SQLI vulnerabilities.	Difficulty in measuring and locating vulnerable code at software component or file level
Gadgikar 2013 [37]	Negative tainting approach does not require any customized and complex runtime environments	Produces false positives at the initial stages; therefore, poor in terms of security system
Ashitah et al. 2014 [38]	The framework provides security level and prevents hackers from exploiting the databases by using SQL injection attacks.	Difficulty in detecting malicious code harder; only recognizes and detects listed character

Most of the defensive techniques, such as: AMNESIA [17], SecuriFly [19], combinational approach [26], combined static and dynamic analyses [30], [32], [35], and hybrid Program Analysis [36], were based on combined Static and Dynamic Analyses. The combination, which is considered highly proficient against SQLIAs, but very complicated, can compensate the limitations of each method. SAFELI [23], SQL-IDS [25] and Prediction static code attributes [31] are three of the proposed approaches based on static analysis that analyzes the code for vulnerability without actually executing the code. Only two, namely, parse tree validation [16] and Sania [22], are based on comparison at run time of the parse tree of the SQL statement before the inclusion of the user input with the result after the inclusion of the input.

6. EVALUATION OF THE DEFENSIVE APPROACHES

In this section, we evaluate the efficiency and effectiveness of the techniques using several different criteria. We first consider which technique efficient by evaluating with performance metrics (false negative, false positive and protocol overhead). We then evaluate the effectiveness of each technique by comparing the techniques based on which technique can detect and prevent the attacks and can stop all types of the attacks. Finally, we identify the techniques that have efficient results and effectiveness based on the empirical evaluation in practices. We examine the author's description of the technique and its current implementation of the efficiency and effectiveness metrics.

6.1 Evaluation Based on Performance Metrics

To evaluate the efficiency of the defensive approaches, three standard performance metrics can be used to compare the previous defensive approaches against SQLIAs. These metrics are:

- False Negative: How many SQLIAs can go undetected by these approaches?
- False Positive: How many legitimate SQL queries are assessed as SQLIA and blocked?
- Protocol Overhead: What is the runtime cost of using the defensive approaches?

Both the false negative and false positive metrics are very important in measuring the effectiveness of security mitigation approaches.

We evaluate each of the proposed approaches to assess whether these are efficient via comparison of their performances techniques based on the standard performance metrics (false negative, false positive and protocol overhead) and on their empirical evaluations in practices. We consider which techniques have result accuracy based on the false negative and false positive values in their empirical evaluation in practices. Subsequently, we compare the techniques with the protocol overhead metric to evaluate the cost of the time process toward obtaining a unified view of their efficiency.

For the purpose of comparison, we divide the techniques into three groups: false negative, false positive and protocol overhead. False negative techniques have many false negative values in their results, false positive techniques have a few false positive values in their results, and overhead protocol techniques have the runtime costs.

Table 2 summarizes the performance metrics evaluations of the precision in practices. We use two different types of markings to indicate how a technique performance. The symbol "✓" denotes that the technique has one of the metrics of that type. Conversely, the symbol "×" denotes that a technique does not have any metrics of performance.

From the accuracy standpoint, we assess each of the techniques with respect to their precision (accuracy) results based on the performance metrics (false negative and false positive) shown in Table 2. Each of the techniques has different result. To evaluate the precision result for each technique, we evaluate each technique with respect to the following criteria. (1) Does the technique have any false negative in the implementation? (2) Does the technique have any false positive in the result and throughput? (3) Does the technique have Protocol overhead in the process and implementation? The answers to these questions are summarized in Table2.

Table 1 Comparison Defensive Approaches for SQLI attacks based on Performance Metrics

Scheme	False Negative	False Positive	Protocol Overhead
Stored Procedures [15]	×	✓	✓
Parse tree Validation [16]	×	×	✓
AMNESIA [17]	×	×	✓
SQL-DOM [18]	×	×	✓
SECURIFLY [19]	×	×	✓
Positive Tainting [20]	×	×	✓
CSSE [21]	×	✓	✓
Sania [22]	×	✓	×
SAFELI [23]	×	×	×
WASP [24]	×	×	×
SQL-IDS [25]	×	×	×
Combinational Approach [26]	×	×	✓
Black -Box Testing [27]	×	✓	×
Database Protection System [28]	×	✓	×
Alert Correlation System (MARS) [29]	×	✓	×
Static and Dynamic Analysis [30]	×	×	×
Prediction Static Code Attributes [31]	×	✓	✓
Static and Dynamic Analysis [32]	×	×	✓
IPAAS [33]	×	✓	×
SQL-IF [34]	×	×	×
Hybrid Program Analysis [36]	×	✓	×
Static and Dynamic Analysis [35]	×	×	×
Negative Tainting Approach [37]	×	✓	✓
Combined Query Tokenization and Adaptive Method [38]	×	×	×

Journal of Theoretical and Applied Information Technology
20th June 2015. Vol. 76 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Most of the proposed techniques have a few false positive in the results, which are: [15], [21], [22], [27], [28], [29], [31], [33], [36] and [37]. On the contrary, none of the techniques have any false negative results. Most of the techniques have remarkable accuracy, which means that these do not have any false negative and false positive values in the implementation. These are: [16], [17], [18], [19], [20], [23], [24], [25], [26], [30], [32], [34], [35], and [38]. Half of the defensive approaches have protocol overhead process, namely, [15], [16], [17], [18] [19], [20], [21], [26], [31], [32] and [37] that affect the technique time cost. And the techniques that did not have any protocol overhead process, namely, [22], [23], [24], [25], [27], [28], [29], [30], [33], [34], [35], [36], and [38]. As a results from our evaluation of the efficiency measurement (result precision) based on the standard performance metrics (false positive, false negative and protocol overhead) illustrate that SAFELI [23], WASP [24], SQL-IDS [25], Static and Dynamic Analysis [30], SQL-IF [34], Static and Dynamic Analysis [35], and Combines Query Tokenization and Adaptive Method [38] can be efficient techniques.
However, the effectiveness of the techniques needs to be measured via comparison based on other criteria prior to the conduct of the technique effectiveness evaluations.

6.2 Evaluation Based on Effectiveness

Each of the techniques have different characteristics in relation to the effectiveness metrics.
To determine the effectiveness metrics required in a technique, we evaluate each technique with respect to the following criteria.
(1) Can the technique detect and prevent SQLIAs?
(2) Can the technique stop all types of the SQLIAs?
The answers to these questions are summarized in Tables 3 and 4.
To compare, we divide the approaches or techniques into two groups, namely, detection and prevention approaches (Table3). Detection techniques detect attacks mostly at runtime. Prevention techniques statically identify the vulnerabilities in the code to stop the attacks.
Table 3 summarizes the results in our evaluation. We use two different types of markings to indicate how a technique performs with respect to a given detection and prevention approach. We use the symbol "✓" to denote that a technique can detect or prevent SQLIAs. Conversely, we use the symbol "✕" to denote that a technique cannot detect or prevent SQLIAs.

Table 2 comparison between Defensive Approaches based on detection and prevention for SQLIA

Scheme	Detection	Prevention	Automated/Code Suggestion
Stored Procedures [15]	✓	✓	Automated
Parse tree Validation [16]	✓	✓	N/A
AMNESIA [17]	✓	✓	Automated
SQL-DOM [18]	✓	✓	Automated/code suggestion
SECURIFLY [19]	✓	✓	Automated
Positive Tainting [20]	✓	✓	Automated
CSSE [21]	✓	✓	Automated
Sania [22]	✓	✓	Automated
SAFELI [23]	✓	✕	Automated
WASP [24]	✓	✓	Automated
SQL-IDS [25]	✓	✕	Automated
Combinational Approach [26]	✓	✓	Automated

Black -Box Testing [27]	✓	✕	Automated
Database Protection System [28]	✓	✓	N/A
Alert Correlation System (MARS) [29]	✓	✕	N/A
Static and Dynamic Analysis [30]	✓	✓	Automated
Prediction Static Code Attributes [31]	✓	✕	N/A
Static and Dynamic Analysis [32]	✓	✓	Automated
IPAAS [33]	✓	✓	Automated
SQL-IF [34]	✓	✓	Automated
Hybrid Program Analysis [36]	✓	✕	N/A
Static and Dynamic Analysis [35]	✓	✓	Automated
Negative Tainting Approach [37]	✓	✓	Automated
Combined Query Tokenization and Adaptive Method [38]	✓	✓	N/A

From the perspective of detection and prevention SQLIAs. Almost all of the techniques or approaches can effectively detect and prevent of the SQLIAs are [15], [16], [17], [18], [19], [20], [21], [22], [24], [26], [28], [30], [32], [33], [34], [35], [37], and [38], except for a few, which are SAFELI [23], SQL-IDS [25], Black-Box Testing [27], Alert Correlation System (MARS) [29], Prediction System [31] and Hybrid program analysis [36]. We evaluate each proposed approaches or techniques and assess it is capable of addressing the different kinds of SQLIAs. In general, the different types of SQLIAs are not performed separately, many are used together or sequentially based on the intention of a specific attacker. SQLIA types are Tautology, Illegal/Logically Incorrect Queries, Union Query, Piggy–Backed Queries, Stored Procedures, Inference and Alternate Encodings.

Our assessment of the techniques is optimistic compared to their performances in practices.

Table 4 summarizes the results of our comparison. We use three different types of markings to indicate how a technique performs with respect to the attack type. We use the symbol "●" to denote that a technique can successfully stop all attacks of a particular type, the symbol "✕" to denote that a technique is not able to stop attacks of a particular type, and the symbol "-"to denote that a technique addresses the attack type partially.

Table 3 Comparison of defensive approaches based on stop to attack types

Scheme	Taut	Illegal/ Incorrect	Piggy-Back	Union	Stored Proc.	Infer.	Alt. Encodings.
Stored Procedures [15]	●	●	●	●	●	●	●
Parse tree Validation [16]	●	●	●	●	●	●	●
AMNESIA [17]	●	●	●	●	×	●	●
SQL-DOM [18]	●	●	●	●	×	●	●
SECURIFLY [19]	-	-	-	-	-	-	-
Positive Tainting [20]	●	●	●	●	●	●	●
CSSE [21]	●	●	●	●	×	●	×
Sania [22]	×	●	●	●	●	●	●
SAFELI [23]	●	●	●	●	●	●	●
WASP [24]	●	●	●	●	●	●	●
SQL-IDS [25]	●	●	●	●	●	●	●
Combinational Approach [26]	●	●	●	●	●	●	●
Black -Box Testing [27]	●	●	●	●	×	●	●
Database Protection System [28]	●	●	●	●	●	●	●
Alert Correlation System (MARS) [29]	●	●	●	●	●	●	●
Static and Dynamic Analysis [30]	●	●	●	●	●	●	●
Prediction Static Code Attributes [31]	●	●	●	●	●	●	●
Static and Dynamic Analysis [32]	●	●	●	●	●	●	●
IPAAS [33]	●	●	●	●	●	●	●
SQL-IF [34]	●	●	●	×	×	●	●
Hybrid Program Analysis [36]	●	●	●	●	●	●	●
Static and Dynamic Analysis [35]	●	●	●	●	●	●	●
Negative Tainting Approach [37]	●	●	●	●	×	●	●
Combined Query Tokenization and Adaptive Method [38]	●	●	●	●	●	●	●

Almost all of the techniques [15], [16], [19], [20], [22], [24], [25], [26], [28], [29], [30], [31], [32], [33], [36], and [38] effectively handle all the SQLI attack types. Some techniques are only partially effective and cannot handle all the attack types for SQLI [17], [18], [21], [23], [27], [34], [35], and [37].

Stored procedures caused problems for most techniques, such as AMNESIA [17], SQL-DOM [18], CSSE [21], black-box Testing [27], SQL-IF [34] and Negative Tainting [37]. With stored procedures, the code that generates the query is stored and executed on the database.

From Tables 3 and 4, we can conclude that the effective techniques that capable of addressing the problems of the SQLIAs and the ability to stop the attack based on the results from our evaluation of the effectiveness metrics are: Stored Procedures [15], Parse Tree validation [16], SECURIFLY [19], Positive Tainting [20], Sania [22], WASP [24], Combinational Approach [26], database Protection System [28], Combined Static and Dynamic Analysis [30], Static and Dynamic Analysis [32], IPAAS [33] and Combined Query Tokenization and Adaptive Method [38].

6.3 Comparison Based on Efficiency and Effectiveness Results

We evaluate each proposed technique to assess whether it is efficient and its effective based on the number of criteria as summarized in Tables 2, 3 and 4. The evaluation of the efficiency is based on the false negative, false positive and protocol overhead values in the results, whereas the evaluation metrics for the effectiveness of the technique are the capabilities to stop all types of the attacks and can detect and prevent the problems of SQLIAs.

Based on the evaluation results of the techniques or approaches that evaluated in section 6.1 and 6.2 to identify the efficient and effective technique, techniques SAFELI [23], WASP [24], SQL-IDS [25], Static and Dynamic Analysis [30], SQL-IF [34], Static and Dynamic Analysis [35], and Combines Query Tokenization and Adaptive Method [38] are efficient. Half of the techniques, namely, Stored Procedures [15], Parse Tree validation [16], SECURIFLY [19], Positive Tainting [20], Sania [22], WASP [24], Combinational Approach [26], database Protection System [28], Combined Static and Dynamic Analysis [30], Static and Dynamic Analysis [32], IPAAS [33] and Combined Query Tokenization and Adaptive Method [38] show possible effectiveness. The results from the comparison between the two metrics (efficiency and effectiveness) illustrate that dynamic tainting (WASP) [24], combined static and dynamic analysis [30] and combined query tokenization and adaptive method [38] can be effective and efficient. On the contrary, techniques Stored Procedures [15], Parse Tree validation [16], SECURIFLY [19], Positive Tainting [20], Sania [22], SAFELI [23], SQL-IDS [25], Combinational Approach [26], database Protection System [28], Static and Dynamic Analysis [32], IPAAS [33], SQL-IF [34], and Static and Dynamic Analysis [35] cannot be efficient and effective.

7. Conclusion

At present, SQLIAs are the most prevalent and dominant class of serious security issue caused by Web application vulnerabilities. SQLIAs are a threat to the security and privacy of both the clients and applications. In this paper, we presented a detailed review and comparison of the most popular existing SQLIAs approaches that have been proposed to solve the attacks. We presented a comprehensive critical review and highlighted the strengths and weaknesses of each approach that has been employed to provide a unified view of the proposed approaches, which could serve as reference for the conduct a comprehensive study in the future. We conducted a comparative analysis of the defensive approaches based on the evaluations respect to efficiency and effectiveness of each approach or technique. In our evaluation, we evaluated comparing the techniques based on the standard performance metrics (false negative, false positive and protocol overhead) to identify the efficiency of the techniques. Subsequently, we evaluated the techniques by comparing them based on the effectiveness metrics, in terms of the capability to stop all types of the SQLIAs, as well as, detect and prevent the attack. We also defined the techniques that are efficient and effective based on the previous comparison.

We found several general trends in the results during the evaluation. Half of the proposed approaches have a few false positive or protocol overheads that refer to the runtime cost. Whilst, none of them have false negative. Many of the techniques have problems handling all types of attacks, such as stored procedures. Whilst, most of all the techniques can detect and prevent of the SQLIAs. In conclusion, only a few techniques have good efficiency and effectiveness results that are: dynamic tainting (WASP) [24], combined static and dynamic analysis [30] and combined query tokenization and adaptive method [38].

Future study should focus on the comprehensive evaluation of the approaches to determine requirements and common development errors, comparison of the performance of the different approaches when these are subjected to legitimate input, and defining the approach environment (e.g. runtime, real time and online environment), and to real-world attacks.

REFRENCES:

[1] K. Amirtahmasebi, S. R. Jalalinia, and S. Khadem, “A Survey of SQL Injection Defense Mechanisms,”, International Conference for Internet Technology and Secured Transactions,(ICITST 2009). (ICITST 2009)2009.

[2] W. G. J. Halfond, S. R. Choudhary, and A. Orso, “Improving penetration testing through static and dynamic analysis,” *Softw. Testing, Verif. Reliab.*, vol. 21, no. 3, pp. 195–214, Sep. 2011.

[3] M.Prabakar, M.KarthiKeyan, and K. Marimuthu.2013, “AN EFFICIENT TECHNIQUE FOR PREVENTING SQL INJECTION ATTACK USING PATTERN,” *2013 IEEE Int. Conf. Emerg. Trends Comput. Commun. Nanotechnol. (ICECCN 2013) AN*, vol. 978–1–4673, no. Iceccn, pp. 503–506, 2013.

[4] E. Athanasopoulos, A. Krithinakis, and E. P. Markatos, “An Architecture for Enforcing JavaScript Randomization in Web2 . 0 Applications,” *Springer-Verlag Berlin Heidelb. 2011*, vol. M. Burmest, no. ISC 2010, LNCS 6531, pp. 203–209, 2011, pp. 203–209, 2011.

[5] T. Abaas, A. S. Shibghatullah, R. Yusof, and A. Alaameri, “Importance and Significance of Information Sharing in,” *Int. Symp. Res. Innov. Sustain. 2014*, vol. 2014, no. October, pp. 1719–1725, 2014.

[6] E. Janot and P. Zavarsky, “Preventing SQL Injections in Online Applications : Study , Recommendations and Java Solution Prototype Based on the SQL DOM.”,Open Web Apllication Security Project(OWASP),2008,Gehent,Belegium.

[7] M. H. A. S. P. Medhane, “Efficient Solution for SQL Injection Attack Detection and Prevention,” no. 1, pp. 395–398, 2013.

[8] OWASP Foundation.Top Ten Risks,2013.http://www.owasp.org/index.php/Top_10_2013_Top_10.

[9] S. Srivastava, R. Ranjan, and K. Tripathi, “Attacks Due to SQL Injection & Their Prevention Method for Web-Application,” vol. 3, no. 2, pp. 3615–3618, 2012.

[10] I. Balasundaram and E. Ramaraj, “An Efficient Technique for Detection and Prevention of SQL Injection Attack using ASCII Based String Matching,” *Procedia Eng.*, vol. 30, no. 2011, pp. 183–190, Jan. 2012.

[11] W.G. Halfond, J. Viegas, and A. Orso, “A Classification of SQL- Injection Attacks and Countermeasures,” Proc. IEEE Int’l Symp. Secure Software Eng., Mar. 2006.

[12] R. Shrivastava, J. Bhattacharyji, and R. Soni, “SQL INJECTION ATTACKS IN DATABASE USING WEB SERVICE : DETECTION AND PREVENTION – REVIEW,” vol. 6, pp. 162–165, 2012.

[13] P. Kumar and R. K. Pateriya, “A Survey on SQL Injection Attacks , Detection and Prevention Techniques,”, Third International Conference Computing Communication & Networking Technologies (ICCCNT 2012) , July, 2012.

[14] S. Sankaran, S. Sitharthan, and M. Ramkumar, “Review on SQL Injection Attacks : Detection Techniques and Protection Mechanisms,” vol. 5, no. 3, pp. 4019–4022, 2014.

[15] K. Wei and M. Muthuprasanna, “Preventing SQL injection attacks in stored procedures,” *Aust. Softw. Eng. Conf.*, p. 8 pp.–198, 2006.

[16] G. T. Buehrer, B. W. Weide, and P. A. G. Sivilotti, "Using Parse Tree Validation to Prevent SQL Injection Attacks," no. September, pp. 106–113, 2005.

[17] W. G. J. Halfond and A. Orso, "AMNESIA : Analysis and Monitoring for NEutralizing SQL-Injection Attacks.", international Conference on Automated software engineering, 2005, Pages 174-183.

[18] S. Diego, G. Drive, L. Jolla, R. A. Mcclure, and I. H. Kruger, "SQL DOM : Compile Time Checking of Dynamic SQL Statements," pp. 88–96, 2005.

[19] M. Martin, B. Livshits, and M. S. Lam, "Finding application errors and security flaws using PQL," *ACM SIGPLAN Not.*, vol. 40, p. 365, 2005.

[20] W. G. J. Halfond, A. Orso, and P. Manolios, "Using Positive Tainting and Syntax-aware Evaluation to Counter SQL Injection Attacks," *Proc. 14th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, pp. 175–185, 2006.

[21] T. Pietraszek and C. Vanden Berghe, "Context-Sensitive String Evaluation," pp. 124–145, 2006.

[22] Y. Kosuga, K. Kono, M. Hanaoka, M. Hishiyama, and Y. Takahama, "Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection," *Twenty-Third Annu. Comput. Secur. Appl. Conf. (ACSAC 2007)*, pp. 107–117, Dec. 2007.

[23] X. Lu, B. Peltsverger, S. Chen, G. Southwestern, K. Qian, and S. Polytechnic, "A Static Analysis Framework For Detecting SQL Injection Vulnerabilities," pp. 1–8.

[24] W. G. J. Halfond, A. Orso, and I. C. Society, "WASP : Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation," vol. 34, no. 1, pp. 65–81, 2008.

[25] K. Kemalis and T. Tzouramanis, "SQL-IDS : A Specification-based Approach for SQL-Injection Detection," pp. 2153–2158, 2008.

[26] C. Science, "Combinatorial Approach for Preventing SQL Injection Attacks accaeds . In view , ritopreses am," no. March, pp. 6–7, 2009.

[27] N. Khoury, P. Zavarsky, D. Lindskog, and R. Ruhl, "An Analysis of Black-Box Web Application Security Scanners against Stored SQL Injection," *2011 IEEE Third Int'l Conf. Privacy, Secur. Risk Trust 2011 IEEE Third Int'l Conf. Soc. Comput.*, pp. 1095–1101, Oct. 2011.

[28] Y. Yan, S. Zhengyuan, and D. Zucheng, "The database protection system against SQL attacks," *2011 3rd Int. Conf. Comput. Res. Dev.*, pp. 99–102, Mar. 2011.

[29] F. Alserhani, M. Akhlaq, I. U. Awan, and A. J. Cullen, "Event-Based Alert Correlation System to Detect SQLI Activities," *2011 IEEE Int. Conf. Adv. Inf. Netw. Appl.*, pp. 175–182, Mar. 2011.

[30] J. Kim, "Injection Attack Detection Using the Removal of SQL Query Attribute Values," *2011 Int. Conf. Inf. Sci. Appl.*, pp. 1–7, Apr. 2011.

[31] L. K. Shar and H. B. K. Tan, "Predicting common web application vulnerabilities from input validation and sanitization code patterns," *Proc. 27th IEEE/ACM Int. Conf. Autom. Softw. Eng. - ASE 2012*, p. 310, 2012.

[32] I. Balasundaram and E. Ramaraj, "An efficient technique for detection and prevention of SQL injection attack using ASCII based string matching," *Procedia Eng.*, vol. 30, no. 2011, pp. 183–190, 2012.

[33] T. Scholte, W. Robertson, D. Balzarotti, and E. Kirda, "Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis," *2012 IEEE 36th Annu. Comput. Softw. Appl. Conf.*, pp. 233–243, Jul. 2012.

[34] K. Natarajan and S. Subramani, "Generation of Sql-injection Free Secure Algorithm to Detect and Prevent Sql-Injection Attacks," *Procedia Technol.*, vol. 4, pp. 790–796, 2012.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[35] I. Lee, S. Jeong, S. Yeo, and J. Moon, "A novel method for SQL injection attack detection based on removing SQL query attribute values," *Math. Comput. Model.*, vol. 55, no. 1–2, pp. 58–68, 2012.

[36] L. K. Shar, H. Beng Kuan Tan, and L. C. Briand, "Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis," *2013 35th Int. Conf. Softw. Eng.*, pp. 642–651, May 2013.

[37] A. S. Gadgikar, "Preventing SQL injection attacks using negative tainting approach," *2013 IEEE Int. Conf. Comput. Intell. Comput. Res.*, pp. 1–5, Dec. 2013.

[38] N. Ashitah, A. Othman, M. Science, U. Teknologi, S. Alam, F. Hani, M. Ali, M. Binti, and M. Noh, "Secured Web Application Using Combination of Query Tokenization and Adaptive Method in Preventing SQL Injection Attacks," no. l4CT, pp. 472–476, 2014.