Protection Web Applications using Real-Time Technique to Detect Structured Query Language Injection Attacks

Nabeel Salih Ali
Information Technology Research and Development Center
University of Kufa
Al-Najaf, Iraq

Abd Samad Shibghatullah
Faculty of Information and Communication Technology
University Technical Malaysia Melaka
Melaka, Malaysia

ABSTRACT
At present, Web applications have been used for most of our life activities increasingly, and they affected by Structured Query Language Injection Attacks (SQLIAs). This attack is a method that attackers employ to impose the database in most of the web applications, by manipulate SQL queries, which sent to the Relational Database Management System (RDBMS). Hence, change the behavior of the applications. In This paper, developing Web Application SQLI Protector (WASP) tool in real-time web application to detect SQL injection attacks in stored procedures. Then, evaluated and analyze the developed tool respect to efficiency and effectiveness in practices. The propose technique uses real-time based on positive tainting, accurate and efficiency taint propagation, and syntax aware evaluation of the query strings at the application level to detect illegal queries before they reach at the database by using Microsoft ASP.NET. The developed tool effective due to it capable of detect and stop all SQLI attacks in real-time environment and did not generate any false negative, a few-false positive values in the results and impose minimal deploy requirements.

General Terms
Security, Approaches, Web attacks, WWW, Web of Things.

Keywords
Web applications, SQL Injection, Detection, WASP, Techniques

1. INTRODUCTION
Nowadays, the Internet is becoming a wide-spread information infrastructure. Whereas, web applications have become the most adequate way to offer access to online services via the Internet [1]. Web applications are becoming a widely pervasive in all kinds of Commercial, corporate businesses, institutions, organizations, numerous financial transactions and other Internet-based services [2]. They often stored valuable and confidential information, which making them a good target for penetration threats that may be achieved by database injection [3]. If an unauthorized user can gain the sensitive information by send malicious code and caused downtime and damages for the services [4]. 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 [5]. However, it is importance to provide the protection of the web applications from the targeted SQLIAs [6]. Structured Query Language Injection attack (SQLIA) is most common and damaging for online services via web applications. This attack takes the benefit of trust existing between the users and server as well as, take the feature of absence of the input/output validation on the server to reject malicious codes [7]. Furthermore, SQLI is a technique that threats a database layer of the web application due to occurring security vulnerability of an application. It is the kind of attack that takes the advantages of incorrect coding via an application [8]. Indeed, SQLIA categorized as one of the top 10 web application vulnerabilities in 2013 experienced by Web applications according to Open Web Application Security Project (OWASP) [9]. SQLIAs is prevalent and dominant classes of serious web application attacks. Hence, could give crafter easily illegal access to the underlying database in web application, Thereby, gain full control of the system caused loss amounts to millions of dollars for corporations. Stored procedures are sub-routines or collections of SQL queries or statements stored within the database that resides in the RDBMS [10]. Stored Procedures (SP) are an important part of modern-day web applications and stored on the server side that they can be available to all clients [11]. Moreover, stored procedures SQL injection attack is one of the serious attacks that posed database threats in the underlying database that underlie web applications. Whereas, the attack can be crafted to execute stored procedures that provided by a particular database, encompasses procedures that deal with the operating system [12].

This research focuses on the analysis and resolution of the problem SQL injection attacks to detect these attacks in real environment via the concept of dynamic positive tainting and syntax-aware evaluation of web application to develop a real-time technique. The main contributions of this research study are: develop WASP tool that has been proposed by Halfond, 2008, to detect the attacks of SQL injection in real-time web applications, evaluate the results accuracy of the propose technique based on the standard performance metrics (false negative, and false positive) as well as, perform the evaluation of the technique effectiveness in practices based on effectiveness metrics. The rest of this paper is organized as follows: In section2, introduces the problem statement of the research. Section3 reviews and discuss the works that have been done to address a particular attacks. Provide a clearly description for the propose design in section4. Implementation and testing of the propose tool will be presented and discussed in section5. Section6 presents the empirical evaluation and analysis for the propose tool. And finally, conclude and outlines the future scope will be provided in section7.

2. PROBLEM STATEMENT
SQLIAs are a threat to the security and privacy of both the clients and the applications. Web applications are frequently vulnerable to SQLI attacks due to poor in design, configuration faults, or weakness written code of the web applications [4]. Hence, these flaws can used by terrorists to collect private data and obtain illegal access to the target [13].

Furthermore, stored procedures are a significant section of modern era web applications [11]. Stored procedures SQL injection attack is one of the serious attacks that posed database threats in the underlying database that underlie web applications. Whereas, the attack can be crafted to execute stored procedures. Typically, system stored procedures are written in SQL, and it stored on the server side. Hence, they are available to all clients. So, when stored procedure is modified, then all clients can get the new version automatically [14].

This research work focuses on detection in SQLI attacks includes stored procedures attacks in real time environment. After carefully analyzing the type of the SQLI attacks in such a scenario, and read previously proposed approaches and techniques. noticed in a WASP tool that proposed by Halfond et al. 2008, to prevent SQLIA's, and shown the great level of effectiveness over the existing methods during its evaluation using data sets at the simulation level as well as, those are not yet tested in the real time environment. Moreover, WASP developed by Medhane.2013 to detect SQLIAs exclude stored procedures. Therefore, propose to develop a WASP tool in order to detect stored procedures SQLIAs under real time settings.

3. PREVIOUS WORKS

Web applications are becoming a widely pervasive in all kinds of Internet-based services in the recent years [2]. Thus, security is becoming one of the major concerns for applications. Hence, it is importance to provide the protection of the web applications from the targeted SQLIAs [6]. Authors have been proposed a wide range of the techniques to address the problem of SQLIA. These techniques for detecting and preventing SQLIA range from filtering, information- flow analysis, penetration testing, development best practices and defensive coding to a fully automated framework. Some of the proposed techniques used to solve SQLIAs required 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 [12].

Wei et al.2005 [15], presented a stored procedures detection tool that used a static analysis of the stored procedure source as a one-time offline procedure via the form of a SQL-graph. But, Limited in terms of developing a complete implementation of the proposed architecture to extend the prototype. SAFELI tool proposed by Lu et al. 2007[16], which capable of discovering very delicate vulnerabilities by taking advantage of source code information. The drawbacks of this tool illustrate via the tool automatically enumerated SQL WHERE clauses by exploring algorithms and does not complete the implementation. As well as, Halfond et al, 2008[17], they conducted WASP tool which efficient and effective in stopping more than 12,000 attacks without generating any false positives. Whilst, the limitations can be found via implemented the approach for the binary applications and deployed web applications. On the other hand, SQL-IDS tool presented by Kemalis and Tzouramanis, 2008[18]. The query-specific detection approached efficient because it stopped the attacks without producing false positives or false negatives. But, limited to identify and check sources and sinks are subject to input validation, and flow-sensitive. In 2009, Kie et al. [19], proposed ARDILLA tool that stronger in Detected real attacks in SQLI and XSS. Whilst, ARDILLA can be created only for PHP script as inputs at a time and cannot simulate sessions are the drawbacks for ARDILLA. In addition to, R-WASP tool that conducted by Medhane, 2013[20]. The R-WASP capable to stop all attacks effectively. But it required more practices in order to mitigate stored procedures attacks efficiently.

Most of the detection tools such as: stored procedures [15], SAFELI [16], WASP [17], SQL-IDS [18], ARDILLA [19], were detected stored procedures SQLIAs. Whilst, only R-WASP [20] the proposed approach that did not detect stored procedures. On the other hand, only two detection tools, namely, SQL-IDS [18], and Real Time-Web Application SQL Injection Detector and Preventer (R-WASP) [20] are detected SQLIAs in real-time environment.

Furthermore, stored procedures [15], WASP [17], were based on combined static and dynamic analysis. The combination, which is considered highly proficient against SQLIAs, but very complicated. Whereas, SAFELI [16], and SQL-IDS [18] are two of the proposed approaches based on static analysis that analyses the code for vulnerability without actually executing the code. But, only ARDILLA [19] based on systematically dynamic taint analysis. The destination from this research paper, to develop a WASP tool in real-time settings in order to detect the attacks of SQLIAs includes stored procedures via the concept of dynamic positive tainting and syntax-aware evaluation of web application.

4. PROPOSE TECHNIQUE DEVELOPMENT (MATERIALS AND METHODS)

In this section, present and describe the proposing system or technique design, design requirements, and methods in order to, develop technique or Real-Time Web Application SQLI Protector (RT-WASP) tool to detect and stop SQLIAs include stored procedures attack.

4.1 Propose System or Technique Design (Modeling)

After identified, the various obstacles and factors for the studies and works (approaches or techniques) that have been done in related works that mentioned in Section3 that would be encountered will lead to be build more successful technique in many aspects of proposed develop technique to address the problem of SQLIAs. To evaluate the performance of the existing methods or techniques that have been proposed, select two of the existent techniques that closed to my technique and the existing work that tried to solve more than one type of SQL injection attacks. These are some limitations in their proposed systems as shown in Table 1.

Table 1. Performance of The Existing Techniques

Author, Year	Tool	Stored procedures SQLIAs	Rest of SQLIAs	Real-Time
Halfond et al, 2008	WASP	✓	✓	✘
Medhane ,2013	R-WASP	✘	✓	✓
Propose Technique	RT-WASP	✓	✓	✓

According to Halfond et al. 2008, proposed an approach in order to mitigate and encounter from the attacks of SQL injection web applications that depending on positive tainting via WASP tool at runtime. On the other hand, Medhane, 2013,

presented a technique to detect SQL injection attacks in real time web application. Therefore, the propose method to detect the danger of SQLIAs is to develop WASP tool in real-time environment.

4.2 Propose Design Requirements

In this section, describes clearly all the requirements that necessary for the system development to detect SQLIAs. It deals with explain in details about Software and Hardware requirements.

4.2.1 User Interface

User Interface (UI) that required for the interaction between the user and the system.

4.2.2 Hardware Requirements

Processor: - P-IV– 0.5 GHz to 3.0 GHz

RAM: - 1GB

Disk: - 20 GB

Monitor: - SVGA

Mouse: - Two or three bottom Mouse

Keyboard: - Standard Windows Keyboard.

4.2.3 Software Requirements

Operating System: - Windows 7/XP

Development End: - ASP.NET (with C#

Programming language)

Web Technology: - HTML, ASP.NET, and CSS

IDE: - Microsoft ASP.NET

Database Server: - SQL Server 2008 R2

4.3 Propose Method to Develop Technique

The first goal of the planned system was to extend. Furthermore, to develop an extremely automated technique against SQL injection attacks that is ready to stop access, and report injection attacks before queries reach the database and performing any damage for sensitive information. Figure1 provides an overview of the propose technique. The propose tool would automatically transform the web application into web application that is semantically equivalent to protect from SQL injection attacks. Likewise, require to evaluate the propose tool under the real time environment. The proposed technique or tool is called a Real Time Web Application SQLI Protector (RT-WASP).

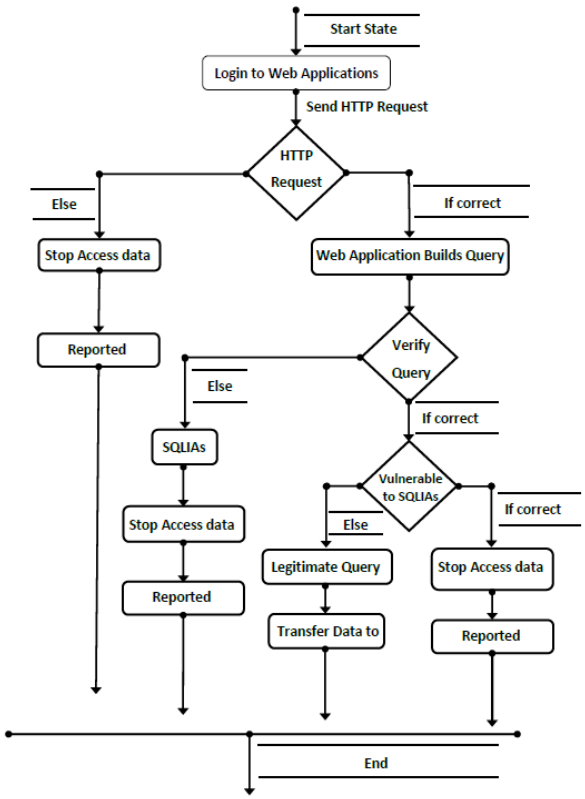

Fig 1: Propose RT-WASP Tool

The RT-WASP tool aims to check if the web pages or websites and web applications have any in danger to the SQLIAs via scanning the web page by URL based on SQLI error queries and SQLI keywords to find out if a site is vulnerable to SQLI. Furthermore, the site can be injected by using the SQLI keywords.

5. IMPLEMENTATION AND TESTING

In this section, describes the methods, and implementation of the proposed technique or RT-WASP tool, and presents the testing results of the RT-WASP tool with set of websites or web applications online.

5.1 Implementation

The RT-WASP aims to check if the web pages or websites and web applications have any vulnerable to the SQLIAs via scanning the web page by URL based on SQLI error queries and SQLI keywords to find out if the site in danger to SQLI as well as, the site can be injected by using the SQLI keywords.

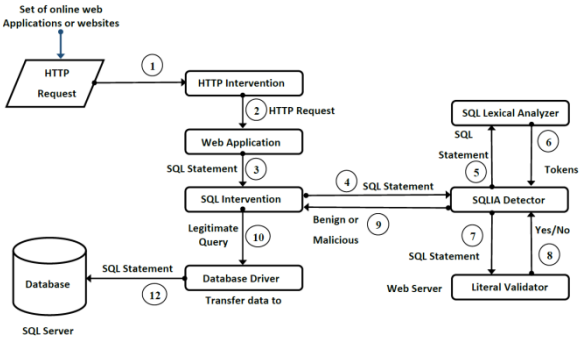

Fig 2: RT-WASP Implementation

Figure 2, shown the detection technique that used for server-side script manipulation via sending queries to the database source RDBMS. The injection process can be done by manipulating client-side data that sent are encompassed changing the SQL values and concatenations of the SQL statement to the web server embedded in HTTP requests. The scanning process or scanning steps to implement RT-WASP tool to detect the attacks of SQLI are included: first step to check HTTP request that sent from set of online websites or web applications by checking the page extension for each URL page, next step will be conducted to check whether, SQL statement or query has any vulnerable to the injection attack that can be done by SQLIA detector that encompasses two mechanisms of checking to perform the detection of SQLIA for each query. These mechanisms are: checking the injection point if any that achieved by SQL lexical analyzer or SQL parser to check the syntax aware evaluation of the query string before sent to DBMS. The next checking will perform checking probability of the injection by literal validator that perform to check whether, the web applications or web sites are possible to inject data keywords. After SQLIA detector decide whether, the query is benign or malicious, legitimate query will transfer data to DBMS. Finally, the tool reported the come out from the scanning process as a report presents all the information regarding to the web page such as: page URL, injection point, injection probability, and the time taken to complete the scan process.

5.2 User Interface (Application)

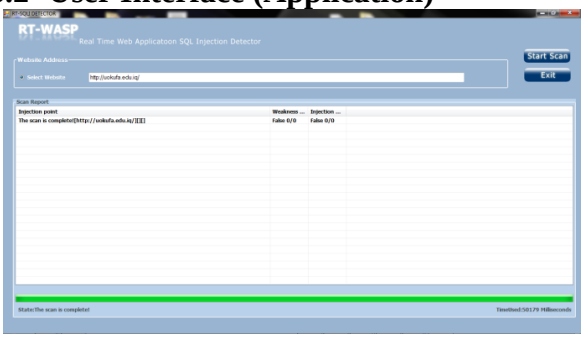

Fig 3: RT-WASP User Interface (UI)

The user interface of the RT-WASP tool contains one User Interface (UI) page with several parts can explain respectively: In the top of the UI has the name of the tool and close, and minimize buttons. "Website address" is the next part in UI that is a text box to insert or put URL of a page to do scanning or checking to the particular page when put URL in "select website". There are two buttons in the top right of the tool UI that is: "Start Scan" to implement the process of scanning and "Exit" to close the UI of the tool. Last part regarding to scan report that present in a grid view all information of the scanned page results. These results included: injection point if any and weakness in sensitive information as well, the probability injection of the page. In addition to, displayed the completing state of the scanning process and the time taken in the scan process. From the "Scan Report" part, can get two values that are "True" and "False" that are denoted to weakness in sensitive information as a result from the checking of the injection point if any, as well as, the probability of the injection that referred to the vulnerability point of the particular page in the web site and the possibility of the injection. As we shown have in Figure 3.

5.3 Testing

In order to, evaluate the efficiency and the effectiveness of the propose technique or (RT-WASP) tool using several different criteria. Firstly test RT-WASP tool with twenty cases (websites pages) in order to, evaluate the RT-WASP tool by test each website page via put the page URL in the "website address" text box and click on "start scan" button in the UI of the tool as shown in Figure 3, to list the real results that came out from the proposed attack detection tool. On the other hand, examines all the website pages by send error query to check the predicated or expected results from the attacks, and then, compared with the actual results that came out from the tool end that can be shown in scan report grid view in the UI for the RT-WASP tool.

From the purpose of the testing, divided the tool testing table into two groups: expected result and the actual result. Each group have two subgroups: weakness in sensitive information and injection probability. From the perspective of expected or predicated result, weakness in sensitive information in the web pages URL, and the injection probability of the web page URL can be examined by, send error query keywords. From the perspective of actual result, weakness in sensitive information in the web page URL can be checked by the injection point if any by RT-WASP tool. Furthermore, checking the probability of the injection also by using proposed tool, and listed the time cost of the technique testing. Used different types of markings to indicate the come out from the tool or the technique in the actual result. The symbol "T" denotes that the technique or tool has one of the weaknesses in the sensitive information, conversely. The symbol "F" denotes that the technique or tool does not have any weakness in the sensitive information and so on so forth. As we see in Table 2.

Table 2. Testing of RT-WASP Tool

Test Case No.	Expected Result		Actual Result		Protocol Overhead
	Injection point	Injection probability	Injection point	Injection probability	
TC1	T	T	T	T	25004 Ms
TC 2	T	T	T	T	11969 Ms
TC 3	T	T	T	T	7267 Ms
TC 4	T	T	T	T	25003 Ms
TC 5	T	T	T	T	25005 Ms
TC 6	T	T	T	T	25003 Ms
TC 7	T	T	T	T	25004 Ms
TC 8	F	F	F	F	49999 Ms
TC 9	T	F	T	F	3631 Ms
TC 10	T	T	T	T	50006 Ms
TC 11	F	F	T	T	25004 Ms
TC 12	F	F	F	F	13817 Ms
TC 13	T	T	T	T	5367 Ms
TC 14	F	F	F	F	24461 Ms
TC 15	F	F	F	F	25004 Ms

TC 16	T	T	T	T	3041 Ms
TC 17	F	F	F	F	4768 Ms
TC 18	T	T	T	T	25004 Ms
TC 19	T	F	T	F	10764 Ms
TC 20	F	F	T	F	25014 Ms

Several points can be addressed from the testing results of the proposed RT-WASP tool via real time settings. Most of the tested websites have weaknesses in their sensitive information and the probability of the injection in their results. Whereas, fifteen sites that have lack with their sensitive data or injection point can be found in TC1-TC7, TC9-TC11, TC13, TC16, and TC18-TC20. Whilst, twelve sites have weaknesses in their injection probability results such as, TC1-TC7, TC10, TC11, TC13, TC16, and TC18. On the contrary, a few tested sites did not have weaknesses in their sensitive data or injection points for instance, TC8, TC12, TC14, TC15, and TC17. Furthermore, eight sites that tested were did not have lacks in their injection probability results such as,TC8, TC9, TC12, TC14, TC15, TC17, TC19, TC20. Whilst, five only of the websites tested did not have any weakness in the sensitive information and the injection probability are, TC8, TC12, TC14, TC15, and TC17.

6. EVALUATION AND ANALYSIS

In this section, evaluate the efficiency and effectiveness of the technique or (RT-WASP) tool using several different criteria. First, consider the technique efficient by evaluating with performance metrics (false negative, and false positive). Then, evaluate the effectiveness of the technique by examine the proposed technique based on capable the technique detected the attacks and can check the probability of the SQLIA.

6.1 Evaluation Based on Performance Metrics

To evaluate the efficiency of the detection technique or tool, two standard performance metrics can be used to examine RT-WASP detection technique against SQLIAs. These metrics are: False Negative: How many SQLIAs can go undetected by this technique? False-Positive: How many legitimate SQL queries are assessed as SQLIA and blocked?. Both the false negative and false positive metrics are very important in measuring the effectiveness of the security mitigation for the detection technique. Evaluate proposed technique to assess whether, the technique or tool is efficient via comparing of the performance technique based on the standard performance metrics (false negative, and false positive) and on the empirical evaluations in practices between the predicated or expected and actual results. From the expected result, the author examined each website page for weakness in sensitive information (injection point) and injection probability by send error query keywords. On the other hand, from the actual result standpoint, the result will be gotten from testing the proposed tool to the same purpose.

For the comparison point, divides the comparison table into two groups: expected result and actual result. Expected result has two subgroups: false negative, and false positive. Table 3 summarizes the performance metrics evaluation of the precision in practices. Moreover, use two different types of markings to indicate the technique performance. The symbol "✓" denotes that the web application or website has one of the metrics of that type of result. Conversely, the symbol "✗" denotes that a web application or website does not have any metrics of performance.

From the accuracy standpoint, Intent to assess each of the websites with respect to their precision (accuracy) results based on the performance metrics (false negative and false positive) for the predicated and actual results. Each of the web application or website had different results, to evaluate the precision of the results for each website based on predicated and actual results. Then, evaluates each web application or websites with respect to the criteria.(1) Does the website have any false negative in the testing result when comparing with their predicated or actual results? (2) Does the technique have any false positive in the result and throughput when comparing with the predicated results? The answers for these questions are summarized in Table 3.

Table 3. Comparison of The Results Based on Performance Metrics

Test case(TC) No.	Expected Results		Actual Results	
	False Negative	***False positive***	***False Negative***	***False positive***
TC1	✗	✗	✗	✗
TC2	✗	✗	✗	✗
TC3	✗	✗	✗	✗
TC4	✗	✗	✗	✗
TC5	✗	✗	✗	✗
TC6	✗	✗	✗	✗
TC7	✗	✗	✗	✗
TC8	✗	✗	✗	✗
TC9	✗	✗	✗	✗
TC10	✗	✗	✗	✗
TC11	✗	✓	✗	✓
TC12	✗	✗	✗	✗
TC13	✗	✗	✗	✗
TC14	✗	✗	✗	✗
TC15	✗	✗	✗	✗
TC16	✗	✗	✗	✗
TC17	✗	✗	✗	✗
TC18	✗	✗	✗	✗

TC19	✗	✗	✗	✗
TC20	✗	✓	✗	✓

The technique or RT-WASP tool has only two false-positive values in the results, which are: TC11 and TC20. On the contrary, the technique did not have any false-negative values in the results. The technique can detect all the SQLIAs in the websites. The technique has all most of the websites that are remarkable accuracy, which means that these do not have any false negative and false positive values in the implementation and testing.

The evaluation results for the efficiency measurement (result precision) based on the comparison results between the technique results and the practical results for the standard performance metrics (false negative and false positive) can illustrate the technique have a few false positive, none of the false negative values, and can be protected all the websites or web applications against the SQLIAs in the implementation. However, the effectiveness of the technique 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 Metrics

The technique or RT-WASP tool has different characteristics in relation to the effectiveness metrics. To determine the effectiveness metrics that required in technique. First, evaluate the technique with respect to the following criteria, (1) can the technique detect the SQLIA? (2) Can the technique examine the probability of the SQLIA if any?. The answers to these questions are summarized in Table 2, and Table 3.

From the detection standpoint, the technique or tool could effectively detect all the websites or web applications of the SQLIA. As shown in Table 3. Furthermore, the technique could effectively have examined or checked all the web applications or websites in terms of the probability of the SQL injection. As had shown in Table 3.

Finally, can conclude that the technique is capable to detect the attacks of the SQLI in all websites or web applications effectively. Additionally, the technique is capable of examined or checked the probability of the injection in all websites effectively.

7. RESULTS AND DISCUSSION

From the perspective of RT-WASP tool evaluation results. Based on the evaluation results of the technique or tool that evaluated in evaluation section (section6) in order to, identify the efficiency and effectiveness of the proposed technique, the proposed technique can be efficient and effective respectively. For the efficient standpoint, because the proposed technique has few false positive and did not have any false negative values in their testing results, and incurs a negligible performance overhead. Whilst, from the effective standpoint, because the technique can detect the attacks and checked whether, the websites or web applications are vulnerable to the injection attacks.

8. CONCLUSION AND FUTURE WORK

Recently, web applications have been used for most of our activities in our life. Web applications are affected by the attacks of SQL injection. Stored procedures SQL injection attack is one of the serious attacks that posed database threats in the underlying database that underlie web applications. Whereas, the attack can be crafted to execute stored procedures that provided by a particular database, encompasses procedures that deal with the operating system.

In This paper, developed WASP tool in order to, build a suitable real-time web application (RT-WASP) tool to detect SQL injection attacks in stored procedures. Then, evaluated and analyzed the developed tool respect to, efficiency and effectiveness of the technique in practices.

In the evaluation, Firstly, evaluated the proposed technique respect to the efficiency based on false negative, and false positive values in the results, whereas, the evaluation metrics for the effectiveness of the proposed technique is the capabilities to detect the serious of the attacks and can examined or checked the injection probability or checked the websites vulnerabilities. From the proposed technique evaluation end, the technique can be efficient and effective respectively.

As a part of the future study, intend to focus on two immediate goals for future scope. The first goal is, to further improve the performance of RT-WASP tool or technique. To this end, plan to extend proposed tool (RT-EASP) to encompass both of SQLI and XSS attacks in the web applications. The second goal is, to ensure the effectiveness of our technique. To this end, plan to develop proposed tool to perform the detection and prevention of SQLI and XSS stored procedures.

9. REFERENCES

[1] 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.

[2] M.Prabakar, M.KarthiKeyan, and K. Marimuthu, "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.

[3] 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.

[4] D. A. Kindy and A. K. Pathan, "A Detailed Survey on Various Aspects of SQL Injection in Web Applications : Vulnerabilities, Innovative Attacks, and Remedies," pp. 1–13, 2012.

[5] N. S. Ali, A. S. Shibghatullah, and M. H. A. L. Attar, "Review Of The Defensive Approaches For Structured Query Language Injection," vol. 76, no. 2, 2015.

[6] 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.

[7] A. K. Baranwal, "Approaches to detect SQL injection and XSS in web applications," EECE 571B, TERM Surv. Pap. April 2012, no. April, 2012.

[8] S. Srivastava, "A Survey On : Attacks due to SQL injection and their prevention method for web application," vol. 3, no. 1, pp. 3225–3228, 2012.

[9] OWASP Foundation. Top Ten Risks, 2013. [Online]. From:http://www.owasp.org/index.php/Top_10_2013_Top_10.[Accessed on 24 November 2013].

[10] P. Kumar and R. K. Pateriya, "A Survey on SQL Injection Attacks, Detection and Prevention Techniques," no. July, 2012.

[11] D. R. Rani, B. S. Kumar, L. T. R. Rao, V. T. S. Jagadish, andM. Pradeep, "Web Security by Preventing SQL Injection Using Encryption in Stored Procedures," vol. 3, no. 2, pp. 3689–3692, 2012.

[12] W. G. J. Halfond, J. Viegas, and A. Orso, "A Classification of SQL Injection Attacks and Countermeasures," 2006.

[13] 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.

[14] J. Clarke and R. M. Alvarez, SQL Injection Attacks and Defense. 2009.

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

[16] 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.

[17] 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.

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

[19] A. Kie, P. J. Guo, and M. D. Ernst, "Automatic Creation of SQL Injection and Cross-Site Scripting Attacks," pp. 199–209, 2009.

[20] M. H. A. S. P. Medhane, "R-WASP : Real Time-Web Application SQL Injection Detector and Preventer," no. 5, pp. 327–330, 2013.