A Component Based SQL Injection Vulnerability Detection Tool

Muhammad Saidu Aliero, Imran Ghani, Murad Khan, Mustapha Atiku and Mannir Bello

Faculty of Computing
Universiti Tecknologi Malaysia
81310 Skudai, Johor Bahru Malaysia
msaidua2000@gmail.com

***Abstract*- SQL injection attack (SQLIA) is one of the most severe attacks that can be used against web database driving applications. Attackers' use SQLIA to get unauthorized access to and perform unauthorized data modification. To mitigate the devastating problem of SQLIA, different researchers proposed variety of web penetration testing tools that automation of SQLI vulnerability assessment that result in SQLIA. Recent study shows that there is need for adaptation of object orienting approach in development of application program in order to reduce the cost of integration and maintenance, as well as improve the efficiency of application programs. Most of the proposed SQLIV (SQL injection vulnerability) detection tools by academic researchers seem to focus on improving efficiency or effectiveness of SQLIV detection tool thereby paying less attention to advantage of adopting reusable component. Therefore, this paper propose component based (CBC) SQLIV detection tool that has the potential to enable developer to reuse component where necessary and allow integration and maintenance fast and in less cost. The proposed tool was tested on three different vulnerable web applications after which its effectiveness was compared against seven(7) different SQLIV detection tool accordingly, the result of evaluation proves that the tool has all the potential to detect SQLIV vulnerabilities on different scenarios that other of scanners ware unable to detect.**

Keywords- SQLIV, SQL injection, Web-based, Vulnerability, Component based.

I. INTRODUCTION

Technology and networks enable organizations to adopt Web based applications as framework on whichthey conduct their day to day activities. For instance, E-commerce, health care, transportation, social activities are now readily available on Web-based database driving applications.Various study shows that the security of Web applications is, in general, quite poor and demand to use these applications is very high[1,2,3]. These applications process data and store the result in back-end database server where the organization's related data are stored. Depending on the specific purpose of application, most applications can be invoked by anyone worldwide to draw the attention of attackers who wish to take advantage of these vulnerabilities. One of the techniques to exploit these applications (Web-based driving database applications) is called SQLIA (SQL injection attack). SQLIA is a situation whereby an attacker modifies programmer intended queries in order to have access to restricted data or perform unauthorized data manipulations [4, 5, 6, and 7]. SQLIA comes in variety of types depending on what attacker wants to accomplish, but the main cause of SQLIA is result of improper validation of input by user which programmer should take care of while developing application [8]. Different researches proposed different techniques of development of the tool that automate testing of SQLIV in Web application. However these techniques not only have limitation on triggering some type of SQLIV but also seem focus on developing tool in traditional way without considering object oriented approach.

To tackle this we proposed component based SQLIVdetection tool with enhanced features that enable the reuse of these components, provide easier integration of new tools, and provide flexibility for improvements and most importantly carryout penetration testing in more effective and efficient way.

I. RELATED WORK

978-1-4673-8227-4/15/$31.00 ©2015 IEEE

Web based penetration testing tools can be classified into three categories; academic, open source tool and commercial [9,10]. Academic tools represent those tools proposed by individuals in a field of research such as SQIVS [11], Enhanced MySQLinjector [12], secubat[13]wave[14], Amnisia[15]etc. Majority of academic tools are language specific, their developments are ongoing process and public access to these tools is unavailable.However, method used in development of such tools are publically available to shade light to individuals or academic researchers who want to improve existing tools or proposed new methods with enhanced features. Unlike academic tools, open source tools such as Vega, Zap, Wa3p, Wapit, Nikito etc [16] are available for public use inform of source code application under copyright for free of charge. However, architecture, algorithm or development approach are not available to public. Individuals or researchers are permittedto study and improve open source tool with consent of the owner. Beside, Open source and academic Web penetration testing tool, there are also commercial tools such as AppScan, Acunetix,Bugblast, Netsparker etc.These tools are totally different from academic and open source tools in the sense that users can only utilize the full functionalities of these tools by purchase, also architecture, algorithms or method used by development of these toolsis not available to public and no vendor allows improvements of their tool[16,17,18]. The advantage of commercial tool over other tools is that; they provide user with extensive help and functionalities that are not available in academic and commercial tools [19].

Basically, there are two common approaches use in development of Web penetration testing tool whether its academic tools, open source or commercial or combination of both. These approaches can be classified into two, static and dynamic approach. Tools that implement dynamic approach are usually known by attacks tools because they analyze server response to find flaws, vulnerabilities through attacking target application.They do not need target source code to perform security analysis. On the other hand, toolsthat implement static approach need to analyze the source code of target application and identified flaws or vulnerabilities via control flow of information, taint analysis, modeling checking or using their combination [20,21,22].

II. OUR APPROACH

Our SQLIV detection tool specifically implements dynamic approach and consists of four main components: Web crawling, attacking analysis and reporting (See Figure 1).

A. Component 1- Web crawling

In order to send an attacks to target application, it is required to identify all links and forms that are potential.Thismay suggest why it is important for penetration tester to make sure that all links and forms potential to SQLIV is identified either automatically or manually[11]

Once given URL (Seed URL) of Website is given to our SQLIV tool, it starts by going through seed URL, extracts all links and forms of application, and stores them in a database. It is very important to know that attacks can only be launched against previously identified links and forms that are potential to SQLIV. Study shows that coverage crawling activities is the one of the major challenges, of SQLIV tool that is implements dynamic approach especially in modern applications that require partial page refreshments or login authentication [23,24]. To avoid such challenge we designed our crawler to first go through seed URL and extract all links and forms and later we extract only forms and links that are potential to SQLIV, unlike crawler design approach used in [8] which contributes to low coverage existing SQLIV in tested application. This is simple becausethe proposed tool crawler was designed to look for the only page with injection parameters which makes tool to logout of the session and manual crawling need to be applied to extract unvisited links by tool.

B. Component 2- Attacking

The proposed tool consist of three sub-attacks components error based, blind and tautology SQIA component each with well designed database of different attack type.

1) Error based SQLIA:

Error-based SQLIA: is type of attack used to test SQLIV in Web application based on return SQL related errors from the application server as a result of violation of developer intended queries[25].Our SQLIV toolhas database of error related SQLIA of 65 different attack patterns. The purpose of this attack is to trick database server to return SQL query related error messages in response to attack request. Occasionally,different database server such as MySQL, Oracle, and SQL Server etc requires different attack patterns that trigger SQL query related errors in server response. Once link or form that is potential to SQLIA is parse the tool start by visiting the link page content, extract parameters list such as “Username”, “Password”, “Search” fields etc as well as Http “Get” and” Post “parameter and the third step in this component is to append attack to parameters list and send request to the database server. It is important to know that attacking activities in attack

component are going parallel to analysis activities.Analysis component also consists of three sub-analysis components each performing different analysis as we have three sub-attack components. Once a response is received by database server, analysis component takes place to check if page content consists SQL query related error. Our tool continues to inject error based attack until all attacksin database is exhausted or if SQL query related error was found the tool breaks the loop and takes the next link or form in question.

Sometimes due to developer configuration SQL related errors would not be returned to user request but application is still vulnerable.In this case,tool moves to next attack type which is blind SQL injection attack [25,26].

2) Blind SQLIA:

Blind SQLIA: is the situation where by tool tries to ask series of true and false questions to database and monitor the behavior response of each question [25]. The trick is that if two different request (true and false request) was found to same response tool it can be deduced that the target link is not vulnerable while if difference exist it can be deduced that the target link is indeed vulnerable to SQLIA. Therefore thetool uses blind SQLIA database that consist 35 valid/true request that works for most of the popular Relational database server constructed using "Order by", "equality""Union" and "If" operator. Once tool sends valid request to application, it stores the response of that request to database and continues by injecting invalid request and comparing the response of first request and previous request until all attacks in database are exhausted or if difference in response found the tool break the loop and parse next link in question.

One of the common challenges we noticed faced by many tools during penetration testing was bypassing the login authentications especially to those tool that apply one step crawling. Not only one step crawling contribute to failure to bypass login application but also include lack of different attack patterns needed to bypass login application as most of the time one attack pattern that bypass MySQL version 1 might fails to bypass MySQL version 2 or attack pattern that can bypass all MySQL server version might fail to bypass other relational database server such as Oracle, MySQL server, Sybase [27] etc. Another common issue is that understanding most common approaches application developers use to adopt while developing query to handle user authentication to applications. For example, application developer might decide to write query that connects valid user to application if and only if, one record is returned from database.However, a successful attack to bypass login authentication using SQLIA always return all records in database[28] and mostly connects last user of record in MySQL server but varies among relational databases.

During our penetration testing we observed that some tools inject successful attack that can bypass application but failed to connect or log into application as result of this rule applied by application developer.

3) Tautologies

Tautology SQIA: is the type of attacks that always shows true value when successfully executed by database server in its form it is constructed using keywords "OR 1=1", mainly used for authorized database extraction and bypassing authentication [25,28,29].

In this context therefore our SQLIV tool has database of tautology SQLIA type of 150 different patterns tested and worked on most commonly used relational database server. 40 of these attacks pattern contains keyword "limit by 1" which handles the rule enforced by application developer. The third attackcomponent used by our tool is tautology attack containing well constructed different attack pattern that enable tool to bypass login authentication of same database server with different versions as well as different database server.

There are four basic ways that automatic scanning tool can predict whether login is successful, these includes: HTTP Basic authentication, NTLM authentication, Form authentication, Setting an HTTP cookie[30] Our SQLIV detection tool uses form authentication. We choose specific text search approach, before choosing this approach. meanwhile we found it very common for most of Web application to use text like"logout","signout" "log out","sign out", "profile" , after user had log into the application and it is difficult to find Web application that uses these text before user log into the system. Each time our tool injects tautology attack it checks to confirm if one of the above texts exists in page response. Our tool continues to inject attack of this type for the purpose of bypassing login authentication until all attacks in database exhausted. Once above text found in page response the tool break the loop and take next link in question.

C. Component 3-Analysis

Study show shows that most of the SQIV detection tool implementing dynamic approach rely on returned SQLI related errors in order to predict whether application is vulnerable or not[11,30]. Our SQLIV

detection tool is capable of performing three different analyses, based on SQL related error messages, similarity between two requests on one page and based on appearance of specific texts as explained (See B in section III).

Reporting Component: each time our tool find SQL query related error response it store the link or form in database called vulnerable page database and store the attack that trigger that SQL related error in database exploits database it performs same action if two request (valid and invalid) request have different response so also same technique apply when one of the following texts"logout","signout" "log out","sign out", "profile" found during authentication bypassing. When all links or forms that are potential to SQLIA is been tested the tool give choice for user choice to generate report in PDF or Word file displaying vulnerable link or form and what attack trigger that vulnerability.

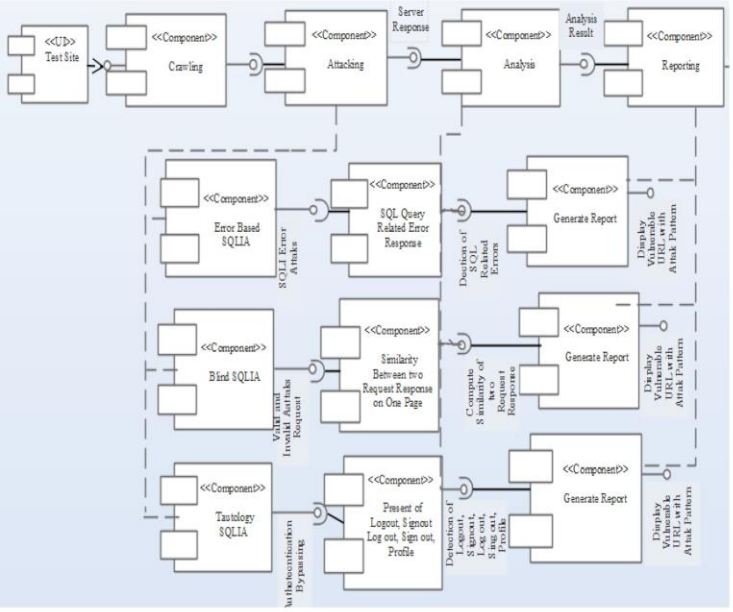

Fig. 1 Component Based SQLIV Tool

III. EXPERIMENTS

During our penetration testing we used open source and commercial Web penetration tools as shown in Table 1 to evaluate our tool against known numbers of vulnerabilities Web applications. Despite there are number of vulnerable applications designed to allow individual or vendor to validate their tool against required vulnerabilities we choose to design three custom Web applications (See Table II) due to the fact that most of the individual or vendors adjust the effectiveness of their tool with respect to vulnerabilities in these applications which may not predict effectiveness of the tool in other application as different developer have different ways of writing same query[9,11]. First is online human resource application consisting of six 6 known SQLIV (see table) one 1 error based SQLIV three 3 blind SQLIV and two 2 vulnerable login authentication, second is online birds farming application with four (4) known SQLIV (see table) 1 error based, 3 blind SQLIV and one vulnerable login authentication and third is online news application with three know SQLIV (see table) one error based SQLIV and 1 blind SQLIV and one vulnerable login authentication, using PHP, all applications are running on window 7 32 bit operating system and 6GB Ram, first and second application running on apache 2.4 with MySQL 5.5.19, and third application running on Apache 2.2.3 with MySQL 5.0.77. Against this background, it may be argued that the major contribution of our work is ability to bypass login authentication on different scenarios as failed by most of the tools we compared to proposedtool. The query that connect employee to application using user.php was developed without enforcing any rule and remaining three login authentication admin.php in HR, login.ph in farm and login.php in news was developed using rule that enforce rule that user should be connected to application if only one row in database is returned.

TABLE I. TOOLS USED TO TEST SQLIV VULNERABLE APPLICATIONS

Tool	Vendor/owner	Version
Vega	Subgraph	N.A
Zap	OWASP	2.4.0
Nikito	CIRT	2.1.5
Wapiti	Informática Gesfor	2.3.0
Acunetix WVS	Acunetix	7.0
W3af	w3af.org	1.6
AppScan	IBM	9.0

TABLE II. THREE CUSTOME VULNERABLE WEB APPLICATIONS WITH KNOWN VULNERABLIITIES

Test bed	SQLI vulnerability	Login required	Vulnerable page
HR	6	2	Admin.php User.php Users_Seach.php Schedule.php Department.php Emp_History.php
Farm	4	1	Login.php Breed_category.php Find_breed.php Purchase_history.php

News	3	1	Login.php News_description.php Browse.php

TABLE III. RESULT OF TESTING AGAINST THREE VULNERABLE APPLICATIONS USING CHOOSING TOOLS AND OUR APPROACH

TOOL	HR	FARM	NEWS
Vega	1/3/0	1/2/0	0/0/0
Zap	1/3/0	1/0/0	0/0/0
Nikito	1/2/0	1/0/0	0/0/0
Wapiti	2/5/1	3/7/0	/0/0/0
Acunetix WVS	1/3/1	1/2/0	0/0/0
W3af	3/3/1	3/4/0	0/0/0
AppScan	1/3/0	1/2/0	0/0/0
Aliero	1/3/2	1/2/1	0/0/1

Table III shows that Vega and appScan detect both error based and blind SQLIV exist in HR and Farm application but failed to bypass any of login authentication in three applications , with no any detection of vulnerability in News application. Zap and Nikito detect both error based and three blind SQLIV, two blind SQLIV respectively in HR application, also both detect error based in farm application but and failed to bypass login authentication in any of three applications with no any vulnerability detection in News application. Wapiti failed to bypass three login authentications one in each applications, however managed to bypass one login authentication in HR and detect both error based and blind SQLIV exist in HR and Farm with ten false negative SQLIV and with no detection of any vulnerability in news application. Acunetix tool detect all error based and blind SQLIV, bypass one login authentication and failed to bypass another in HR, similarly it detect error based and blind SQLIV in Farm application and failed authentication in Farm with no any detection of vulnerability in news application. W3af detect all error based and blind SQIV in both HR and Farm with Five false negative and with no detection of any vulnerability in news application and also failed to bypass any login authentication in three applications. Our proposed tool detects all vulnerability in HR and Farm and also successfully bypasses all login authentications in all three applications but failed to detect error based and blind SQLIV in news applications. Some of the reasons most of the tools failed to bypass login authentication is that they do not have well constructed tautology attacks to successfully bypass login authentication and other do not use tautology at all in their attack databases. Another reason is that all three authentications queries used in three Website (see table) tested tool on use the check that if return row is not equal to one authentication would not succeed. To confirm failure of authentication we perform similar experiment used in (2) to monitor the communication between tool and target application using wireshark and we found that Vega, Acunetix and AppScan send successful attacks that enable bypassing login authentication but because of the rule applied by query developer authentication get failed all the time. It is important to know that considering all scenarios while developing attacks pattern for bypassing login authentication is very important i.e using such as “LIMIT BY 1” allows returning of one row for successful SQLIA as required by developer . As shown in table 4 non-of the tool was able to detect any error based or blind SQLIV due to the fact that request to the database was designed in such a way that no Http post or get was used to retriev the information from database. This shows that most of the Web penetration tools depend on Http post and get to send request to database.

IV. CONCLUSION

In this paper we proposed component based for SQLIV detection tool for the purpose of enhancing component reusability, fast integration and maintenance in less price. The evaluation of our SQLIV detection tool with seven selected SQLIV detection tools against three testbed applications shows significant difference in which our tool detects most of the vulnerabilities that other scanner ware not able to detect. The future work is to update our tool so that it would be able to detect other web applications vulnerabilities such as XSS(cross-site-scripting) and file inclusion vulnerabilities.

ACKNOWLEDGEMENT

This research was fully supported by “UniversitiTeknologiMalaysia, Johor Bahru, Malaysia. Authors would like to acknowledge Faculty of Computing for supporting this work.

REFERENCES

[1] Tudor, J. 2013. Web Applications Vulnerability statistic 2013. Retrieved on 02/09/2015 from; http://sitic.org/wp-

content/uploads/Web-Application-Vulnerability-Statistics-2013.

[2] OWASPD-Open Web Application Security Project 2014. Top ten most critical Web Application Security Risks. Retrevied on 2/09/2015 from; http://cwe.mitre.org/cwss/archive.htm

[3]Cenzic's. 2014. Application Vulnerability Trends Report: 2014. Retrieved 29/07/2015, from https://www.info-point-security.com/sites/default/files/cenzic-vulnerability-report-2014.pdf

[4] H. Shahriar, andM. Zulkernine 2012. Information-theoretic detection of sql injection attacks. High-Assurance Systems Engineering (HASE),2012 IEEE 14th International Symposium on, IEEE, 40-47.

[5]A.Joshi, V. Geetha2014. SQL Injection detection using machine learning, Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on, IEEE, 1111-1115

[6] M. A Prabakar, M. Karthikeyan, M. and K. Marimuthu 2013. An efficient technique for preventing SQL injection attack using pattern matching algorithm, Emerging Trends in Computing, Communication and Nanotechnology (ICE-CCN),2013 International Conference on IEEE, 503-506.

[7] W.G.Halfond, J. Viegas and A. Orso 2006. Classification of SQL Injection Attacks and Countermeasure, IEEE International Symposium on Secure Software Engineering (ISSSE 2006), pp. 87—96.

[8]R. McClure, and I. H. Kruger 2005. SQL DOM: compile time checking of dynamic SQL statements. SoftwareEngineering2005.Proceedings.27th International Conference on IEEE ISBN:1-59593-963-2page 88 – 96.

[9]N.Antunes and M. Vieira 2010. Benchmarking vulnerability detection tools for web services. Paper presented at the Web Services (ICWS), 2010 IEEE International Conference on.

[10] V. Livshits, and M. S. Lam 2005. Finding Security Errors in Java Programs with Static Analysis. In Proceedings of the 14th Usenix Security Symposium, pages 271–286.

[11]Z. Duric2013. A black-box testing tool for detecting SQL injection vulnerabilities. Paper presented at the Informatics and Applications (ICIA), 2013 Second International Conference on IEEE.

[12]A.Liban, and H. Shadi. 2014. Enhancing Mysql Injector vulnerability checker tool (Mysql Injector) using inference binary search algorithm for blind timing-based attack. Paper presented at the Control and System Graduate Research Colloquium (ICSGRC), 2014 IEEE 5th.

[13]S. Kals Kirda E. Kruegel Christopher, and J. Nenad 2006. Secubat: a web vulnerability scanner. Paper presented at the Proceedings of the 15th international conference on World Wide Web.

[14] Y. Huang, S. Huang, T. Lin, and C. Tsai 2003. Web Application SecurityAssessment by Fault Injection and Behavior Monitoring. In Proceedingsof the 11th International World Wide Web Conference (WWW 03),May 2003.

[15] W. G. Halfond, and A. Orso, 2005. AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection attacks", Proceedings of the 20thIEEE/ACM international Conference on Automated softwareengineering, pp. 174-183, 2005.

[16]OWSAP Open Web Security Project. Retrieved 29/06/2015, from https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools

[17]M.F Jnena 2013. Modern Approach for WEB Applications Vulnerability Analysis retrieve on 27/8/2015 fromhttp://library.iugaza.edu.ps/thesis/109553.pdf

[18] Shay Chen 2011. Security Tool Benchmarking available at http://sectooladdict.blogspot.my/2011/08/commercial-web-application-scanner.html

[19] Russell Clarke and David Dorwin 2010. Is Open Source Software More Secure? Retrieved on 28/7/2015 from http://courses.cs.washington.edu/courses/csep590/05au/white paper_turnin/oss(10).pdf

[20]X. Zhang, and Z. Wang. 2010. Notice of Retraction A Static Analysis Tool for Detecting Web Application Injection Vulnerabilities for ASP Program. Paper presented at the e-Business and Information System Security (EBISS), 2010 2nd International Conference on.

[21]L. Zhang, et al. 2010. D-WAV: A web application vulnerabilities detection tool using Characteristics of Web Forms." Software Engineering Advances (ICSEA), 2010 Fifth International Conference on. IEEE, 2010.

[22] F. Jose´ S. Nuno , V. Marco , and M. Henrique"Analysis of field data on web security vulnerabilities."Dependable and Secure Computing, IEEE Transactions on 11.2 (2014): 89-100.

[23] A. Doupé, L. Cavedon,C. Kruegel, and G. Vigna, 2012. Enemy of the State: A State-Aware Black-Box Web Vulnerability Scanner. In USENIX Security Symposium (pp. 523-538).

[24] O. Brandman, O.,J. Cho,H. Garcia-Molina, and S. Shivakumar, (2000). Crawler-friendly web servers. Performance and Architecture of Web Servers (PAWS) 2000.

[25] D. Chad 2012. Practical Identification of SQL Injection Vulnerabilities. Retrieved on 3/8/2015 from https://www.us-cert.gov/sites/default/files/publications/Practical-SQLi-Identification.pdf

[26]S. Kevin 2003. Are your web applications vulnerable? available at http://www.net-security.org/dl/articles/Blind_SQLInjection.pdf

[27] Preventing SQL Injection Attacks. Retrieved on 3/8/2015 from at http://www.applicure.com/solutions/prevent-sql-injection-attacks

[28]R. Joseph Manoj et al 2014. An Approach to Detect and Prevent Tautology Type SQL Injection in Web Service Based on XSchema validation, International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 3 Issue 1, Jan 2014 Page No. 3695-3699

[29] E.H Cheon, Z. Huang, and Y.S Lee 2013. Preventing SQL Injection Attack Based on Machine Learning, International Journal of Advancements in Computing Technology Vol. 5, No. 9, pp. 967 - 974.

[30] http://w3af.org/howtos/authenticated-scans

[31] A. Ciampa, C.A Visaggio, and M. Penta, 2010. A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications. InProceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems (pp. 43-49). ACM.