Performance-based Comparative Analysis of Open Source Vulnerability Testing Tools for Web Database Applications

Alekha Kumar Mishra
Department of Computer Applications
National Institute of Technology Jamshedpur
Jamshedpur, India - 831014
Email: alekha.ca@nitjsr.ac.in

Arun Kumar
Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, India - 769008
Email: kumararun@nitrkl.ac.in

***Abstract*—In the last two decades, with the increase of information communications technology, the cybersecurity threats to web-based data-bases have seen exponential growth. Though the web database resources are tested for possible cyber threat vulnerabilities, it is challenging to identify the security weak points and back-doors of web database and applications to achieve data confidentiality and integrity. The vulnerability testing tools are used to find specific vulnerabilities on software or applications. This paper studies, analyses and compares the open-source vulnerability testing tools, SQLMAP and JSQL, for web database applications. A set of dummy URLs are introduced to evaluate the penetrating ability of both tools, and an analysis is presented to aid the users to choose a suitable tool based on security requirements.**

***Index Terms*—Cybersecurity, Cyberthreats, SQL Injection Attack, Vulnerability testing, SQLMAP, JSQL**

I. Introduction

With the advancement of computing and communication technologies, the cyberthreats to confidential and financial data residing in web databases are increased to a higher level [1]. There is a different kind of cyber threats, but the most common cyber threats include Denial of Service (DoS), Phishing, Spyware and Malware, and SQL injection (SQLi) [2]. SQLi is a simple yet very serious cyberthreat to the valuable data on the web. The SQLi threat occurs due to designing negligence and providing low-level security protection to webpages. An SQLi attack aims to break into the database to retrieve user credentials like tables, user id, password etc. [3]. These information leakages cause substantial financial loss to the victims. It is estimated that the database developed with MySQL and PHP environment are most vulnerable to SQLi attack [4]. The Open Web Application Security Project (OWASP) [5] reported that the SQLi attack is the topmost threat covering almost 97% of all the data penetrations threats.

Therefore, it is essential to identify the security weak points and back-doors of web data and applications to achieve data confidentiality and integrity [6]. A vulnerability testing tool can help to accomplish this goal by summarizing details of vulnerabilities on a software or application [10]. The primary task of a vulnerability tool is to quantify the existing threats [12]. In order to achieve this, a number of tests are performed for the crucial entities of an infrastructure at various levels. There are several tools available to test SQLi vulnerabilities of a web database both in licensed and open source categories. It is cumbersome to choose the right tool based on application data types, parameters, and database design.

This paper aims to support the task of choosing the right tool for testing SQLi vulnerabilities through a detailed experimental comparison. The two most popular tools preferred by the security analyzers, SQLMAP [7] and JSQL [8], [9], are selected for this purpose. Both of these tools are open source and are available with Kali Linux distribution. Also, this paper aims to study, analyze, compare and infer important conclusions about the ability of these tools to exploit the various weaknesses of given URL for SQLi attack. A list of parameters is used for comparison against a set of test URLs.

The remainder of the paper is organized in the following order. Section II provides the characteristics and process involved in a SQLi attack. Section III presents the literature survey on tools used and tested for SQL injection. Section IV and V summarizes the information about JSQL and SQLMAP respectively. The experimental parameters used for comparison are illustrated in Section VI. Section VII presents the comparative results of the tools followed by the conclusion in Section VIII.

II. SQL Injection (SQLi)

In a SQLi attack, an attacker uses an SQL query to collect the sensitive information from the database [13]. The victims of this threat are the webpages that are designed without following the proper guidelines and testing process. In this attack, the attacker creates a partial malicious SQL query, which is then inserted to input fields of the web page form [3]. Once the HTTP request to the server is generated, it becomes the part of the complete query which is executed at the database server. Since this type of queries are not checked and validated due to imperfect designing process of webpages, it leads to the retrieval of sensitive data from the database.

SQLi has variations based on the partially injected query type upon the original formed query. This part of the query can be an invalid logical condition, a tautology, a piggy-backed or any other query. A successful SQLi may lead to the loss of access controls, data integrity, and confidentiality [11].

SQLi attacker actions include retrieving sensitive data, overriding authorization, deleting essential data and tables from the database. In-band SQLi variation involves the insertion of malicious code into a web application and retrieving the database results [14]. It may be either an error based or union based in-band SQLi attack [15], [16]. Blind SQLi variation is also known as inferential SQLi attack [3], [11]. In this variation, the attacker rebuilds a local database schema using payload and analyses the reply of the database server. It requires more time compared to other varieties but possesses the highest level of threat. The Blind SQLi attack can be either a Boolean or Time-based attack [11]. Finally, the Out-of-bound SQLi variation involves a direct conversation between the database server and the attacker's machine under its supervision [3], [16].

The SQLi can be prevented by following the best code development techniques and abiding by the rules and policies for developers [11]. The SQLi attack detection approaches perform several tests to the databases for exposed vulnerabilities. The run-time protection of SQLi attack requires a variety of intrusion detection concerning the queries requested to the database [11].

III. Background

Simon *et al.* [17] used a combinatorial testing approach to identify SQL injection vulnerabilities in web applications. Also, the authors have developed an automated SQL injection vulnerability scanner that can perform tests using covering array vectors, called SQLINJECTOR. The tool is designed with three sets of attack vectors for vulnerable testing. Ojagbule *et al.* [16] used Nikto and SQLMAP for penetration testing on three popular content management systems: WordPress, Drupal, and Joomla. The SQLMAP tool is used to test the common SQLi variations. Thakre *et al.* [12] compared and analyzed a number of tools such as SQLMAP, Acunetix, VEGA, IronWasp, WebCruiser and others for their ability to test system vulnerabilities. The comparison is done with a dummy banking application created for this purpose. The results indicate only the ability of the tools to perform various vulnerability tests such asSQLi, Cross-Site Scripting (XSS), Security Mis-configuration, Sensitive Data Exposure.

To the best of our knowledge, there exist no research work in the literature which experimentally compares the two most popular open-source testing tools SQLMAP and JSQL for their ability to identify the SQLi vulnerabilities. In this work, SQLMAP and JSQL are studied, analyzed, and evaluated experimentally to compare their performance.

IV. JSQL

JSQL is a lightweight cross-platform SQLi testing tool [8], [9]. It is available as a built-in tool in Kali Linux operating system and used for performing SQLi attack. The tool has a graphical user interface which is developed in Java. It also supports command-based utility for vulnerability scanning of URL database. JSQL can traverse a database through remote host, seek for an administrative page, upload a file, decode/encode a string, and perform brute-force hashes. In addition to these tasks, it can also generate SQL query through an open terminal, and test several URLs for SQLi attack.

V. SQLMAP

SQLMAP [7] is an open-source tool for testing SQLi vulnerability. It is designed to explore SQLi vulnerabilities and to find a way to control the database server. It is a powerful database scanner [16]. The tool can be used for database fingerprinting and executing OS commands.

The critical functions of SQLMAP can be listed as follows: i) Website scanning for identifying SQL Injection flaws, ii) Analysis of SQLi vulnerabilities, and iii) sensitive information extraction from databases. The procedures are also available for the error-based check, boolean-based blind, time-based blind UNION query, out-of-band and stacked queries. Also, it can break a password stored in hash format and provide useful fields from the database such as database and table names, column list, user name, password hashes, privileges, and roles.

VI. Experiment and Results

In this section, the list of testing parameters used during vulnerability test are provided. A set of fifteen test URLs are picked up and listed below for evaluating the performance of SQLMAP and JSQL. These are

1) http://testphp.vulnweb.com/listproducts.php?cat=1
2) www.icdcprague.org/index.php?id=10
3) http://www.durgabernhard.com/book.php?bookID=32
4) http://www.futurefins.com/fin-detail.php?id=173
5) http://www.davidshop.com/showcat.php?id=55
6) http://www.ipic.com/shopping-centre.php?id=1
7) http://www.ud.org.tw/web/news.php?id=50
8) http://www.handikitchen.com.au/about.php?id=1
9) http://tasaceramic.vn/product_detail.php?id=122
10) www.altcine.com/details.php?id=1980
11) http://www.afss.org/sports.php?id=12
12) http://www.smelisting.net/corner_category.php?id=15
13) http://www.polymery.ru/material.php?id=18
14) http://www.smtmax.com/category.php?id=15
15) www.tunesoman.com/product.php?id=200

The list of JSQL parameters used for testing is provided below.

1) *BIGINT:exp check*: This is an overflow testing mostly done in MySQL 5.5.5 and higher versions. If the highest value of BIGINT data type is taken and used in an arithmetic expression, it may lead to out of range condition.
2) *Javascript Object Notation (JSON)*: This includes check for JSON format.
3) *XML:extract value*: This check explores for XML injection vulnerabilities.

4) *Strategy Time based blind*: It finds the specific amount of time that a back-end database waits for generating a report. This information can be helpful to guess a query's possible outcomes.
5) *Strategy Blind*: This parameter is tested for blind injection.

The parameters to test SQLMAP are listed below.

1) *Boolean-Based Blind*: It checks for Boolean-based Blind SQLi which is also referred as a content-based inferential SQLi.
2) *Error-Based check*: This checks for error-based SQLi possibilities to extract data.
3) *Time-Based Blind*: This checks for the test which is performed, where there are no much options [PLEASE CHECK THIS SENTENCE]. This test-fires a large number of queries and depending on the server's response time information is deduced.
4) *Heuristic Basic*: This test infers the vulnerabilities of GET parameters.
5) *UNION-based*: In this check, union-based query vulnerabilities are tested.

TABLE I
VULNERABILITY TESTS SUMMARY FOR JSQL.

Test parameter lists	URLNo.(s) Found Vulnerable
BIGINT:exp check:	None
Double:exp check	None
Groupby:float_req_check	1,6,7,8,10,12,14
JSON	None
XML:extract value	1,6,7,8,10,12,14
Strategy time check	3,7,13
Strategy blind check	3,7,13

The summary of the SQLi vulnerability test using JSQL is shown in Table I. Similarly, the vulnerability test result of SQLMAP is shown in Table II. Here, the second column provides the URL number(s) from the URLs listed previously in this section that are found vulnerable, if any for the mentioned test/check using the corresponding tool.

VII. ANALYSIS

This section presents the analysis of comparison results of SQLMAP and JSQL in detail. The vulnerable indicates that the tested URL is vulnerable to SQL injection. In other word, the URL is injectable. Table I refers that none of the test URLs are found vulnerable under the test parameters BIGINT:expcheck, Double:expcheck, and JSON. Though seven URLs are found vulnerable to XML: extract_check and Groupby::float_req_check. The Strategy time check and Strategy blind check indicates that only three out of fifteen URLs are found vulnerable by JSQL. It is also observed that JSQL is able to scan all the URLs.

It can be observed from Table II that none of the URLs are found vulnerable by Heuristic-basic check. Whereas, URL No. 6 is found vulnerable for the MySQL>5.X AND/OR error-based check. A maximum of eight URLs is found vulnerable for the Boolean-Based Blind check. Both Error-Based check and UNION query test found four URLs as vulnerable by SQLMAP. Six URLs are found vulnerable for AND/OR Time-Based Blind check and only five URLs are found vulnerable for the MySQL> 5.0.X Time-based blind check. It can be concluded from the above numbers that SQLMAP is able to report vulnerable for a larger number of URLs for various tests performed compared to JSQL.

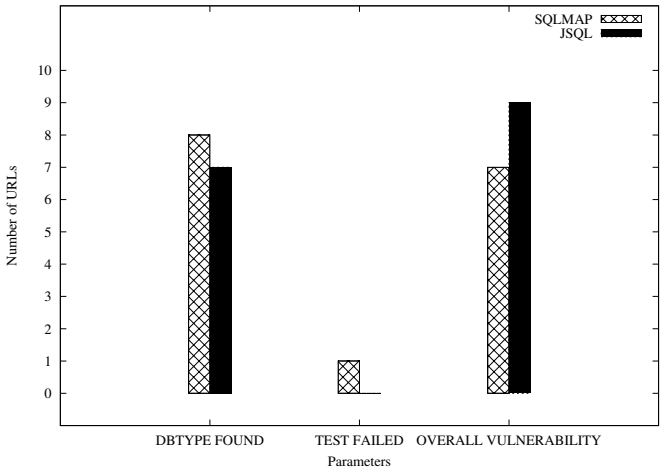

Fig. 1. The overall comparison summary between JSQL and SQLMAP.

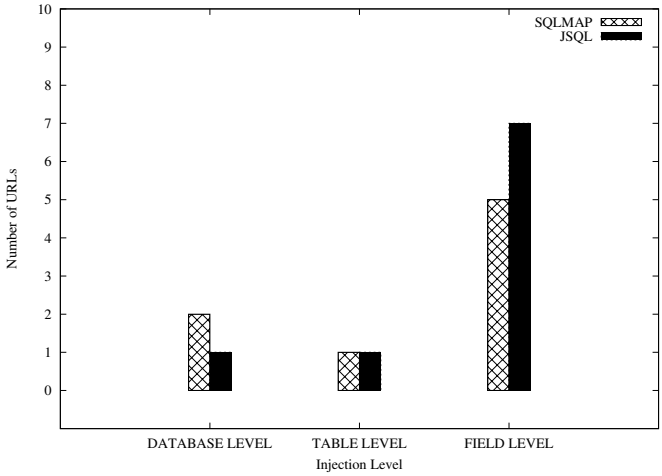

Fig. 2. The comparison database injection level between JSQL and SQLMAP.

Figure 1 shows the comparison of common tests and parameters such as database type found, failed test, and overall vulnerability. It can be observed that the number of URLs for which database type is identified is one more than JSQL. The SQLMAP fails to perform a vulnerability scan for only the 3rd URL, whereas JSQL is able to scan all the URLs successfully. The SQLMAP vulnerable test fails to perform a vulnerability scan for the 3rd URL. The number of URLs found vulnerable to SQL injection is nine for JSQL,

TABLE II
VULNERABILITY TESTS SUMMARY FOR SQLMAP.

Test lists	URLNo.(s) Found Vulnerable
Boolean-Based Blind check	1,2,5,6,8,10,11,13
Error-Based check	1,6,8,10
AND/OR Time-Based Blind check	1,2,5,8,11,13
UNION query	1,5,6,13
Heuristic-basic	None
MySQL>5.X AND/OR error-based check	6
MySQL> 5.0.X Time-based blind	2,5,6,11,13

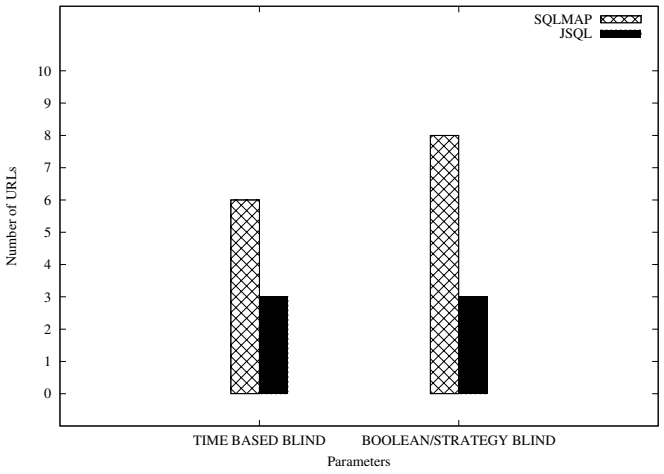

Fig. 3. The comparison blind injection tests between JSQL and SQLMAP.

whereas seven for SQLMAP. Figure 2 shows a comparison of SQLMAP and JSQL in terms of the injection depth. It can be observed that JSQL is able to inject up to field level of the database for seven URLs, whereas SQLMAP is able to scan at field level for only five URLs. Both are able to scan up to the table schema level for only one URL. However, the SQLMAP scan is restricted only up to the database level for two URLs. Therefore, Figure 2 concludes that for the injection depth scan, the JSQL performs marginally better than SQLMAP. Also, It can be concluded that SQLMAP is better in performing individual and overall vulnerability checks, whereas JSQL is better in performing an injection depth scan into the database structure. Figure 3 shows the comparison of blind injection test results. It can be observed that the number of URLs found vulnerable to all blind injection by SQLMAP is higher compared to the number of URLs found by JSQL. That means SQLMAP is efficient for testing an URL for blind injection vulnerabilities.

VIII. CONCLUSION

This paper has presented a performance-based comparative analysis of open source vulnerability testing tools for web database applications. Vulnerability testing is highly essential to identify possible weaknesses and security threats in an organization's data resources and web applications. It is also crucial for an organization to select a suitable tool to protect its databases. Therefore, in this work, two open-source vulnerability testing tools, SQLMAP and JSQL, are studied in detail. The list of the parameters used are listed and tested for a select set of test URLs. It is observed that JSQL is better suited for testing SQLi vulnerability. It is faster than SQLMAP in performing the scan. However, SQLMAP supports more number of tests compared to JSQL. It is helpful for a detailed scan summary apart from the standard SQL injection vulnerability test.

In the current work, JSQL and SQLMAP comparison is limited to only test web databases. In the future, comparison of the strength and ability of both the tools can be extended against a set of real web databases with necessary permission and procedures. Further, the injection level of a particular database system can be compared and analyzed by using security mechanism at different layers of web database architecture such as permission granularity, data digest, encrypted field etc.

ACKNOWLEDGMENT

The authors would like to thank the NVIDIA Corporation for sponsoring the Titan Xp GPU card as a part of GPU Grant program.

REFERENCES

[1] J. Abawajy, "User preference of cyber security awareness delivery methods," Behaviour & Information Technology, vol. 33, no. 3, pp. 237-248, 2014.
[2] F. L. Greitzer, A. P. Moore, D. M. Cappelli, D. H. Andrews, L. A. Carroll and T. D. Hull, "Combating the Insider Cyber Threat," IEEE Security & Privacy, vol. 6, no. 1, pp. 61-64, 2008.
[3] N. Singh, M. Dayal, R. S. Raw, and S. Kumar, "SQL injection: Types, methodology, attack queries and prevention", In 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, pp. 2872-2876, 2016.
[4] M. Nashaat, K. Ali and J. Miller, "Detecting Security Vulnerabilities in Object-Oriented PHP Programs", In 17th International Working Conference on Source Code Analysis and Manipulation (SCAM), pp. 159-164, 2017.
[5] "Top 10 Web Application Security Risks," https://owasp.org/www-project-top-ten/.
[6] B. Sullivan and V. Liu, "Web Application Security, A Beginners Guide," (1st. ed.) 2011, McGraw-Hill Education Group.
[7] "SQLmap Tutorial and Resources to Learn sql mapping," https://coderseye.com/learn-sqlmap-and-tutorial.
[8] "jSQL Injection usage guide: a multifunctional tool for scanning and exploiting SQL injection in Kali Linux," https://miloserdov.org/?p=1682.
[9] "jSQL Automatic SQL Injection Tool In Java, "https://www.darknet.org.uk/2017/08/jsql-automatic-sql-injection-tool/.

[10] R. M. Parizi, K. Qian, H. Shahriar, F. Wu and L. Tao, "Benchmark Requirements for Assessing Software Security Vulnerability Testing Tools", IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, pp. 825-826, 2018.

[11] A. Sadeghian, M. Zamani, and A. A. Manaf, "A Taxonomy of SQL Injection Detection and Prevention Techniques", In International Conference on Informatics and Creative Multimedia, Kuala Lumpur, pp. 53-56, 2013.

[12] S. Thakre, and S. Bojewar, "Studying the Effectiveness of Various Tools in Detecting the Protecting Mechanisms Implemented in Web-Applications", In International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 1316-1321, 2018.

[13] L. Liu, O. De Vel, Q. Han, J. Zhang, and Y. Xiang, "Detecting and Preventing Cyber Insider Threats: A Survey," IEEE Communications Surveys Tutorials, vol. 20, no. 2, pp. 1397-1417, 2018.

[14] P. A. Sonewar and N. A. Mhetre, "A novel approach for detection of SQL injection and cross site scripting attacks," In International Conference on Pervasive Computing (ICPC), pp. 1-4, 2015.

[15] A. Sadeghian, M. Zamani, and S. Ibrahim, "SQL Injection Is Still Alive: A Study on SQL Injection Signature Evasion Techniques," In International Conference on Informatics and Creative Multimedia, pp. 265-268, 2013.

[16] O. Ojagbule, H. Wimmer and R. J. Haddad, "Vulnerability Analysis of Content Management Systems to SQL Injection Using SQLMAP", *SoutheastCon 2018*, pp. 1-7, 2018.

[17] D. E. Simos, J. Zivanovic and M. Leithner, "Automated Combinatorial Testing for Detecting SQL Vulnerabilities in Web Applications," In 14th International Workshop on Automation of Software Test (AST), pp. 55-61, 2019.

[18] B. Nagpal, N. Singh, N. Chauhan and A. Panesar, "Tool based implementation of SQL injection for penetration testing," In International Conference on Computing, Communication & Automation, pp. 746-749, 2015.