International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 06 | June - 2021 ISSN: 2582-3930

SQL INJECTION VULNERABILITY DETECTION

Mansi Jindal*
Department of CSE
HMRITM, GGSIP University
New Delhi, India

Anshul Gupta*
Department of CSE
HMRITM, GGSIP University
New Delhi, India

Tejsi Sharma
Department of CSE
HMRITM, GGSIP University
New Delhi, India

Shivam Singhal
Department of CSE
HMRITM, GGSIP University
New Delhi, India

Sakshi Singh
Assisstant professor
Department of CSE
HMRITM, GGSIP University
New Delhi, India

Abstract - In this SQL Injection attacks pose a big threat to the protection of web applications. Because these databases often contain sensitive consumer or user information, the resulting security violations can include identity thefts, loss of wind and fraud. Existing approaches don't support object oriented approach that renders these approach to safeguard the important world web apps like wordpress, joomla or drupal against SQL attacks. In some cases, attackers can even use an SQL Injection vulnerability to require control of and corrupt the system that hosts the online applications. SQL Injections refers to a category of code injections attacks during which data provided by the user is included in an SQL query in such the way that a part of the user's input is treated as SQL code. This paper presents SQL injection principles, SQL injection attacks forms, it's different types and how SQL injection can be prevented.

Key Words: ***SQL injection, vulnerability, SQL injections Attacks***

1. INTRODUCTION

Malicious SQL statements may be introduced into a vulnerable application using many alternative input mechanisms like Injection through user input, Injection through cookies, Injection through server variables, Second order injection.

Various types of SQL Injection attacks are available like tautologies, illegal/logically incorrect queries, UNION queries, Piggy-backed queries, Stored Procedures, Blind SQL, Timing Attack, Alternate Encoding and etc. [1]. Defecting these styles of attacks isn't simple since the attacker actually changes the behavior of predefined SQL queries.

The reason for SQL injection vulnerabilities is comparatively simple and well understood: insufficient validation of

user input. to handle this problem, developers have proposed a spread of coding guidelines that promote defensive coding practices, like encoding user input and validation [2]. However, in practice, the applying of such techniques is human-based and, thus, at risk of errors. Furthermore, fixing legacy code-bases which may contain SQL injection vulnerabilities are often a very labor-intensive task. Although recently there has been an excellent deal of attention to the matter of SQL injection vulnerabilities, many proposed solutions fail to deal with the total scope of the matter [3]. There are many sorts of SQLIAs and countless variations on these basic types.

Web applications that are liable to SQL Injection Attacks (SQLIAs) are widespread—a study by Gartner Group on over 300 Internet websites has shown that the majority of them can be prone to SQLIAs [4]. In fact, SQLIAs have successfully targeted high-profile victims like Travelocity, FTD.com, and Guess Inc.

2. OVERVIEW OF SQL INJECTION

The Classic SQL injections were easy to forestall and detect and much of procedures, method- ologies were discussed to beat SQL injections. the various methodologies to beat the attack is writing secure code to keep with an intensive investigation.

The concept of attacks by injection is to insert or inject a malicious code into a program to change SQL query structure. Such an attack is completed by adding malicious character strings within the info values within the fashion of arguments values within the URL [5]. SQL injection attack or SQLIA (Structured source language Injection Attack) could also be a method of code injection attacks consisting of malicious injection SQL commands through client data file to application that is passed to the instance of the database for its execution and with the target of affecting the execution of predefined SQL commands.

SQL injection attack (injection) is that the commonest and easiest style of vulnerability technique adopted by the on-line attackers through data-driven web applications [6]. By using simple SQL commands like Select, Where, Insert, Delete and Update, the malicious attackers efficiently re-structure the actual SQL code (statements) and execute vulnerable code into the online applications. Once nasty attacker attains their goal, they'll easily access sensitive information, modify secured data, execute the data, and even they'll collapse the entire application. Since the database administrator's privacy has lost their role by accesses of malicious unauthorized softwares [7,8].

Occurrence of SQL injection attacks are possibly when the effect that is intended by an SQL query by inserting operators and the keywords of new SQL inside the query of an attacker changes [9]. This informal definition is meant to incorporate all of the variants of SQLIAs reported in literature and presented during this paper. Interested readers can ask for more formal definition of SQLIAs. within the remainder of this section, we define two important

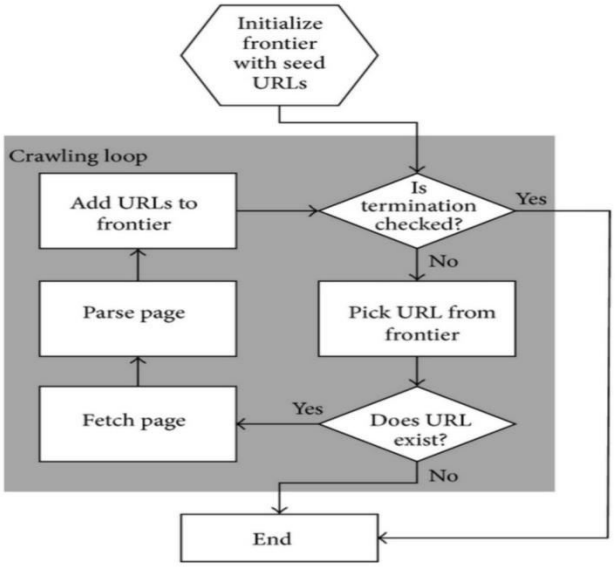
characteristics of SQLIAs that we use for describing attacks: injection mechanism and attack intent.

3. TECHNIQUE TO DETECT AND STOP SQL INJECTION

The foremost chosen techniques are static analysis, dynamic analysis, combined static and dynamic analysis, web framework, defensive programming and machine learning techniques. The strategy of static analysis is extreme where it analyzes the code for vulnerability by without actually executing the code [10]. Software metrics and reverse engineering are some forms of static analysis.

Model checking, data flow analysis, abstract interpretation and use of assertions in ASCII text files are the several techniques of static code analysis. The strategy of dynamic analysis is performed automatically by the analysis of vulnerabilities during the execution of web applications which avoids thousands of tests by doing several times manually

The method of combined static and dynamic analysis can compensate the constraints of each method, which is taken into consideration as highly proficient against SQLIAs but it's very complicated [11]. One among the only examples for such the simplest way is that the AMNESIA tool. It uses static analysis to research the web-application code and automatically build a model of the legitimate queries that the applying can generate. The technique is designed to monitors all the dynamically-generated queries and in order to checks them in case of any compliance at the run-time using the statically-generated model. Whenever these techniques are going to detects an issue which thereby violates the credentials of model, it basically classifies the query as an attack which thereby prevents it from by directly accessing the database, and currently logs the information related to that attack. The web framework method might be a filtering method of user input parameters. This method is proven to be ineffective while its unable to filter some special characters [12]. The machine-learning method is that the most typically



used method whereas the strategy lands up in high false positives and low detection rate. The newly proposed algorithm relies on dynamic technique which violates SQL injection attacks [13]. This algorithm concentrated to develop IF (Injection Free) attacks

Figure 1: Crawling loop

4. INJECTION MECHANISM

Malicious SQL statements is introduced into a vulnerable application using many various input mechanisms. during this section, we explain the foremost common mechanisms.

4.1. Injection through user input

During this case, attackers inject SQL commands by providing suitably crafted user input. an internet application can read user input in several ways supported the environment within which the applying is deployed. In most SQLIAs that concentrate on Web applications, user input typically comes from form submissions that are sent to the online application via HTTP GET or POST requests [14]. Web applications are generally able to access the user input contained within these requests as they'd access the other variable in the environment.

4.2. Injection through cookies

Cookies are basically considered as one of the files which definitely contains the state of information which is generally generated by the Web applications and which are generally stored on the client/server machines. When a client returns to an online application, cookies are often accustomed restore the client's state information. A malicious client hereby could tamper with the contents related to cookies [15]. If an online application uses the cookie's contents to make SQL queries, an attacker could easily submit an attack by embedding it within the cookie [16].

4.3. Injection through server variables

Server variables are a set of variables that contain HTTP, network headers, and environmental variables [17]. Web applications use these server variables in a very type of ways, like logging usage statistics and identifying browsing trends. If these variables are logged to a database without sanitization, this might create an SQL injection vulnerability [18,19]. Because attackers can forge the values that are placed in HTTP and network headers, they'll exploit this vulnerability by placing an SQLIA directly into the headers [20]. When the query to log the server, variable is issued to the database, the attack within the forged header is then triggered.

5. VULLNERABILITY SCANNER

To scan the vulnerability of websites we designed a tool based of Bash language. The scanner is compatible with Linux as well as windows operating system.

5.1 Functionality and Implementation

The scanner scans the given website for SQL vulnerabilities. First it checks for source code vulnerability, if nothing is found there, it proceeds to find vulnerability point through brute forcing and open ports. If any case is breached the website is reported to be vulnerable and if not, it's reported non vulnerable.

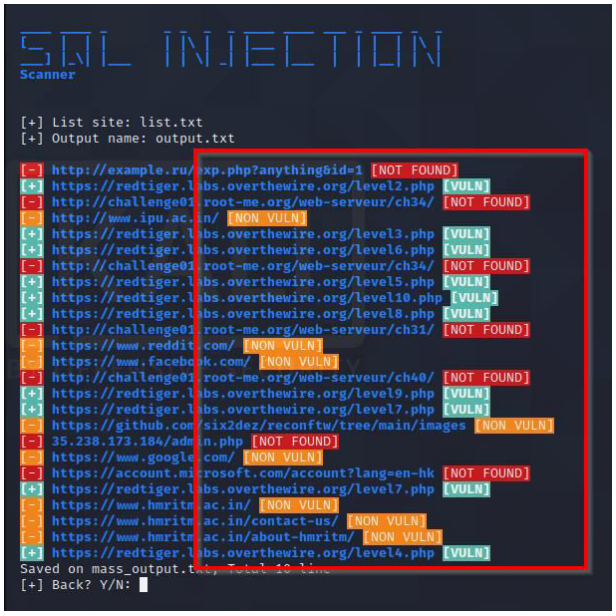
There is possibility that domain injected is outdated or scrapped off the web, in that case the website is reported to be not found. After providing input of list doc and output where user wants to store collection of vulnerable sites among the whole bunch, he/she provided, scanner will start running. All the sites after being scanned will appear on the console in real time. We have added three color coding to distinguish among the result easily. These collared labels with tags

"NOT FOUND"," VULN" and "NON VULN" will appear in the end of each tag.

6. RESULT

The bash vulnerability scanner is showing result according to the vulnerability results. If that site is not found on internet i.e., the site is old and not available on internet, then it will indicate NOT FOUND in red. If the site is found, and it comes out vulnerable, it will be indicated (VULN) in green colour coding. If the site is found but no SQL injection vulnerability is found in it will show NON VULN in orange colour coding.

While the program runs user can see



(+), (-) in the left lane of the console. The (+) indicates the files that will be added in the output files. These are files which are vulnerable to sql injection as detected by our scanner. The ones marked (-) are either not found or non-vulnerable. These won't be added in output file.

Once the process ends, it will indicate how many files are found vulnerable and stored in output file.

Figure 2: Scanned output of sites

7. CONCLUSION

In this paper, we've got presented a survey and comparison of current techniques for detecting and preventing SQLIAs. To perform this evaluation, we first identified the assorted varieties of SQLIAs known so far. In order to detect and to prevent these attacks some of the techniques are considered on the basis of their compatibility. We also studied the various mechanisms through which SQLIAs will be introduced into an application and identified which techniques were able to handle which mechanisms.

The proposed algorithm is considerable in inspection of its simple detection operation against SQL injection attacks [21]. Testing of web applications, mobile application and desktop application for SQL injection attack could also be a big step for ensuring its performance and quality. [22,23]. The proposed algorithm performs much faster and endowed with a proficient solution to resolve against SQL injection attacks [24]. The paper- work has been analyzed with various detection methods and thus the proposed method cannot only be implemented on web applications and might even be used on any applications which interact towards databases.

REFERENCES

1. W. H. Rankothge, M. Randeniya and V. Samaranayaka, "Identification and Mitigation Tool for Sql Injection Attacks (SQLIA)," 2020 IEEE 15th International Conference on Industrial and

Information Systems (ICIIS), 2020, pp. 591-595, doi: 10.1109/ICIIS51140.2020.9342703.

2. K. Zhang, "A Machine Learning Based Approach to Identify SQL Injection Vulnerabilities," 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019, pp. 1286-1288, doi: 10.1109/ASE.2019.00164.

3. I. Tasevski and K. Jakimoski, "Overview of SQL Injection Defense Mechanisms," 2020 28th Telecommunications Forum (TELFOR), 2020, pp. 1-4, doi: 10.1109/TELFOR51502.2020.9306676.

4. Z. C. S. S. Hlaing and M. Khaing, "A Detection and Prevention Technique on SQL Injection Attacks," 2020 IEEE Conference on Computer Applications(ICCA), 2020, pp. 1-6, doi: 10.1109/ICCA49400.2020.9022833.

5. H. Gu et al., "DIAVA: A Traffic-Based Framework for Detection of SQL Injection Attacks and Vulnerability Analysis of Leaked Data," in IEEE Transactions on Reliability, vol. 69, no. 1, pp. 188-202, March 2020, doi: 10.1109/TR.2019.2925415.

6. Y. Xu et al., "Injection-Locked Millimeter Wave Frequency Divider Utilizing Optoelectronic Oscillator Based Optical Frequency Comb," in IEEE Photonics Journal, vol. 11, no. 3, pp. 1-8, June 2019, Art no. 5501508, doi: 10.1109/JPHOT.2019.2916919.

7. B. Chen, H. Li and B. Zhou, "Real-Time Identification of False Data Injection Attacks: A Novel Dynamic-Static Parallel State Estimation Based Mechanism," in IEEE Access, vol. 7, pp. 95812-95824, 2019, doi: 10.1109/ACCESS.2019.2929785.

8. R. T. Prapty, S. Azmin Md, S. Hossain and H. S. Narman, "Preventing Session Hijacking using Encrypted One-Time-Cookies," 2020 Wireless Telecommunications Symposium (WTS), 2020, pp. 1-6, doi: 10.1109/WTS48268.2020.9198717.

9. Li Qian, Zhenyuan Zhu, Jun Hu and Shuying Liu, "Research of SQL injection attack and prevention technology," 2015 International Conference on Estimation, Detection and Information Fusion (ICEDIF), 2015, pp. 303-306, doi: 10.1109/ICEDIF.2015.7280212.

10. A. Jana, P. Bordoloi and D. Maity, "Input-based Analysis Approach to Prevent SQL Injection Attacks," 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 1290-1293, doi: 10.1109

11. K. Yan, Y. Zhang, S. Pan, Q. Bao and Z. Chen, "Design and Implementation of Real-Time Extraction Software for Direct Wave Parameters," 2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC), 2018, pp. 261-265, doi: 10.1109/IMCEC.2018.8469392.

12. M. A. P. Subali and S. Rochimah, "A new model for measuring the complexity of SQL commands," 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE), 2018, pp. 1-5, doi: 10.1109/ICITEED.2018.8534782.

13. M. Yassin, H. Ould-Slimane, C. Talhi and H. Boucheneb, "SQLIIDaaS: A SQL Injection Intrusion Detection Framework as a Service for SaaS Providers," 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), 2017, pp. 163-170, doi: 10.1109/CSCloud.2017.27.

14. A. Shah, S. Clachar, M. Minimair and D. Cook, "Building Multiclass Classification Baselines for Anomaly-based Network Intrusion Detection Systems," 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), 2020, pp. 759-760, doi: 10.1109/DSAA49011.2020.00102.

15. Z. Xiao, Z. Zhou, W. Yang and C. Deng, "An approach for SQL injection detection based on behavior and response analysis," 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017, pp. 1437-1442, doi: 10.1109/ICCSN.2017.8230346.

16. S. Reetishwaree and V. Hurbungs, "Evaluating the performance of SQL and NoSQL databases in an IoT environment," 2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM), 2020, pp. 229-234, doi: 10.1109/ELECOM49001.2020.9297028.

17. R. -Z. Wang, Z. -H. Ling, J. -B. Zhou and Y. Hu, "A Multiple-Integration Encoder for Multi-Turn Text-to-SQL Semantic Parsing," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 1503-1513, 2021, doi: 10.1109/TASLP.2021.3070726.

18. S. B. Misal, P. L. Yannawar and A. T. Gaikwad, "DBQA: Multi-Environment Analyzer for Query Execution Time and Cost," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), 2017, pp. 1050-1055, doi: 10.1109/CTCEEC.2017.8455009.

19. K. Kamtuo and C. Soomlek, "Machine Learning for SQL injection prevention on server-side scripting," 2016 International Computer Science and Engineering Conference (ICSEC), 2016, pp. 1-6, doi: 10.1109/ICSEC.2016.7859950.

20. K. T. Sridhar, "Reliability techniques for MPP SQL database product engineering," 2017 2nd International Conference on System Reliability and Safety (ICSRS), 2017, pp. 180-185, doi: 10.1109/ICSRS.2017.8272817.

21. Y. Zhong and L. Xu, "TagNet: Tag Out the Value Sequence of SQL Statement," 2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS), 2019, pp. 806-810, doi: 10.1109/ICICAS48597.2019.00173.

22. C. Ping, W. Jinshuang, Y. Lanjuan and P. Lin, "SQL Injection Teaching Based on SQLi-labs," 2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE), 2020, pp. 191-195, doi: 10.1109/ICISCAE51034.2020.9236904.

23. A. Singh, A. Sharma, N. Sharma, I. Kaushik and B. Bhushan, "Taxonomy of Attacks on Web Based Applications," 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), 2019, pp. 1231-1235, doi: 10.1109/ICICICT46008.2019.8993264.

24. R. A. Katole, S. S. Sherekar and V. M. Thakare, "Detection of SQL injection attacks by removing the parameter values of SQL query," 2018 2nd International Conference on Inventive Systems and Control (ICISC), 2018, pp. 736-741, doi: 10.1109/ICISC.2018.8398896.