International Journal of Soft Computing 11 (2): 103-106, 2016
ISSN: 1816-9503

Test Input Generation for Detecting SQL Injection Vulnerability in Web Application

[1]Nor Fatimah Awang, [2]AzizahAbd Manaf and [3]Siti Fatimah Abidin
[1]Faculty of Defence Science and Technology, National Defence
University of Malaysia, Kuala Lumpur, Malaysia
[2]Advanced Informatics School (UTM AIS), UTM
International Campus, Kuala Lumpur, Malaysia
[3]MySEF Lab, Cyber Security Malaysia, The Mines Resort City,
Seri Kembangan, Selangor, Malaysia

Abstract: In software testing, one of the critical issues is the selection of adequate test input. In this study, we formulate a method to generate test input by using permutation and combination algorithm technique in order to generate a set of test input automatically. We also develop a tool called an input generator that automatically generates the test input. The tool is a text list generator based on permutation algorithm on combination of pattern. The text list is generated based on the pattern given in a file template and combination of pattern is generated from the files to form list of text or statement. This attack pattern is formulated based on SQL attack type such as tautology, illegal and piggy-backed type. This tool is able to generate a large number of test inputs based on pattern given by tester at a lesser time. Finally, we show how ideas derived from our method will generate a set of test inputs and able to perform an attack and produce the results.

Key words: Security testing, penetration testing, test input generation, web, SQL

INTRODUCTION

Web application systems are one of the most ubiquitous software systems is used nowadays. Since, they have grown quickly and have evolved faster than other software systems, the growth of these applications gave a high impact and business opportunity to the organization. Thus, the security of the web applications should be emphasized by the organization. The web application will be exposed to any existing vulnerabilities such as SQL injection and cross site scripting. With the majority of vulnerability exists in web applications, it is important to evaluate and detect the vulnerability of web application before it is sent to production (Lei *et al*., 2013). Many organizations are starting to take initiatives to prevent these types of attacks. To minimize the probability of vulnerabilities exist in web applications, organizations need some methodologies or approaches to increase efforts to protect against web-based application attack (Halfond *et al*., 2006; Stuttard and Pinto, 2007). One of the methodology is security testing. The purpose of security testing is to find any security weaknesses or vulnerabilities within an application and document all the vulnerabilities to help developer to fix them. However, the selection of adequate test input is very critical issues in testing phase (Djuric, 2013; Akrout *et al*., 2014). Therefore, in this study we propose an automation of generation of test inputs based on attack pattern. We separated the method into three stages. Formulate test input based on attack pattern, develop input generator. This generator will be used to generate a set of test inputs automatically; test execution. The goal of this phase is to perform testing executionby sending automated test inputs into web server. The main contributions of this study are as follows:

- We formulate test input based on attack pattern
- We develop a tool called an input generator in order to generate a set of test inputs automatically using permutation and combination algorithm

Literature review: This section provides a brief background on SQL injection vulnerabilities and reviews previous work on SQL injection testing based on test input generation.

Corresponding Author: Nor Fatimah Awang, Faculty of Defence Science and Technology,
National Defence University of Malaysia, Kuala Lumpur, Malaysia

Table 1: SQL attack pattern

Attack type	Description	Sample of attack pattern
Tautologies	Inject code to one or more SQL query and these query will always evaluate to true. Attackers can use tautologies to exploit this peace of code by inserting this value 'or 1 = 1' to the text box of login page in where username clause	'Or 1 = 1 #
Illegal/logical	Queries will be used to display an error message that can give detailed information	Insert or delete
Piggy back	Queries will be used to alter, delete or disable the application	
Union	Inject query using the keyword union. UnioN statement allows an attacker to combine two separate SELECT statements into one result	'Union all select @@ version--

Back ground: SQL injection is an attack setting in which attackers inject malicious SQL code query into the input parameters. SQL injection is also one part of code-injection attacks which are data provided by the user is included in SQL query (Awang *et al*., 2014; Halfond *et al*., 2006). By using SQL query, the attacker might construct input values in a way that changes the error or the behavior of the resulting response from a web server and enabling the attacker to perform actions on the database server. These actions could lead to exposure of sensitive data, insertion or alteration of data without authorization, loss of data or even taking control of the database server (Stuttard and Pinto, 2007; Bisht *et al*., 2010). Let us discuss the basic idea of SQL injection in web applications. The example makes use of web form with multiple strings inside in the HTML code. Users are able to select the appropriate strings based on their choices. In this scenario, by using method POST in HTML format, users are needed to input both a valid username and password in order to access the application.

Once a browser runs an HTML code, web form such as login.php will be executed and special functions in login.php are used to send the SQL statement to the database server to be executed. From the web form, the variables login and password is an input provided by the legitimate users or malicious users. Valid user might enter valid login and password as shown here. To detect whether there are vulnerabilities inside the applications or not, the attacker might inject some malicious codes into the variables. The attacker might inject some basic codes such as (') single quote (") double quote and 1 = 1 as shown below in order to observe the response and gain the information from the database.

The attackers can infer the database structure by exploiting error messages from SQL command failure (Ezumalai and Aghila, 2009; Kindy and Pathan, 2013) or simply by trial and error by using single quote or double quote. If the input login has the value or the resulting response might be an error message written by the database server. If the input has the value, 'or 1 = 1--, the resulting response might bypass the password and gain access to the application and database. The first quote closes the existing quote in the statement and the double dash at the end comments out the final quote, making the resulting SQL statement syntactically valid. The clause 'or 1 = 1 is a tautology type and make the condition will always be true, bypassing the original condition in the where clause and resulting unauthorized database access. Therefore, the malicious user has been authenticated without a valid login id and password. The use of tautology is a well-known SQL attack (Wodarz, 2008; He *et al*., 2008)

SQL attack patterns: This section will describe various types of SQLI attack patterns that frequently used by the attackers (Wodarz, 2008; He *et al*., 2008; Wassermann *et al*., 2008; Alata *et al*., 2013; Bozic and Wotawa, 2014; Chen and Wu, 2010; Awang *et al*., 2014) with the sample of attack pattern (Table 1).

MATERIALS AND METHODS

In this study, we briefly describe our proposed method in order to formulate an attack pattern, generate attack pattern automatically and execute it.

Formation of attack pattern: This phase is to identify SQL attack pattern to formation of test input. Since, SQL attack patterns have been discussed in previously, for example tautology type, several pattern files are defined and created to store all the possible data that can be used to generate SQL injection attack. The attack pattern will be generated based on simple patterns such as comments, line separator, operators and SQL syntax. A combination of simple patterns can build many different attack patterns.

Input generator: The main purpose of the input generator is to generate and develop test inputs automatically. We present an approach to detect SQL injection vulnerabilities by generating test inputs automatically by using a combination of injection attack type and all test inputs are generated using permutation algorithm. Input generator program consist of three main classes. These classes will process the input and generate permutation result based on the master_template.txt file.

Table 2: Input generator component

Pattern File	Attack string/pattern template
Master_template.txt	#sql#operator#non_alphanumeric#sql#operator
#inline_comment	
#sql	OR, AND
#operator	1 = 1
#non_alphanumeric	‘,“
#inline_comment	--, #

- InputGenerator(): constructor function hat initializes the required inputs pattern such as alpha numerics, numerics, SQL, separators, operators, etc
- ProcessInput(): function that process the inputs read from txt file
- GeneratePattern(): function that read master_ template.txt and replace the pattern with required input characters

Based on sample in Table 2, we can see, there are two lines in master_template.txt and input generator will generate the test input separated and saved in different file. By applying the permutation and combination technique algorithm we shall get the following results to perform the testing execution.

Test execution: In this section, the test input generated from previous section will be used to execute tests. In order to perform the execution an Apache HttpClient Java library was installed and implemented in order to retrieve response page from a web server. This proposed framework will observe the behavior of the application by using HTTP message to send and HTTP response to retrieve from the web server. This technique is extremely fast to retrieve a response from the web server.

RESULTS AND DISCUSSION

This section presents the results of tests carried out to verify the model of study. WackoPicko website was chosen for this experiment. WackoPicko is an online photo sharing website that allows users to upload, comment and purchase pictures. This experiment focuses only on SQL injection vulnerability. Our method was deployed by setting up the Eclipse development environment with Java Program. The Apache HTTP Client API library was installed in the machine to extract HTTP header from response pages. In the test input generation phase, 800 attack test injections were generated against SQL injection attack. We ran the method with injecting test input and the results are summarized in Table 3. The results are categorized into types of SQL attack and types of error messages. HTTP response reply from webserver

Table 3: Vulnerability detection result

Application	Target parameter	Type of SQL attack	Type of error	Vulnerability detected (%)
WackoPicko	Username password	Tautology	Bypass application	2.97
		Illegal/logical	SQL error	69.23
		Union	Unknown column, unknown table	2.01
		Piggy back	Shutdown server	0.00

will indicate the vulnerability if error messages such as SQL error and bypass authentication results were appearing inthe HTML document header. Table 3 shows at least 2.97% of generating test inputs were successfully by passed the application and entered the application. Based on the results of the test it could be concluded that all input forms are vulnerable to the website.

CONCLUSION

In this research, we propose a method to generate test input by using an attack pattern technique with applying permutation and combination algorithm to generate it automatically. This method has been successfully tested and including several types of SQL injection vulnerabilities. Our method is able to address the vulnerabilities based on results in Table 3. Future work will also be focused on extending our approach to cover other types of vulnerabilities such as cross site scripting vulnerabilities.

ACKNOWLEDGEMENT

This study was supported by the Advanced Informatics School (AIS), University Technology of Malaysia, National Defence University of Malaysia and Cyber Security Malaysia.

REFERENCES

Akrout, R., E. Alata, M. Kaaniche and V. Nicomette, 2014. An automated black box approach for web vulnerability identification and attack scenario generation. J. Braz. Comput. Soc., Vol. 20. 10.1186/1678-4804-20-4.

Alata, E., M. Kaaniche, V. Nicomette and R. Akrout, 2013. An automated approach to generate web applications attack scenarios. Proceedings of the 6th Latin-American Symposium on Dependable Computing, April 1-5, 2013, Rio de Janeiro, pp: 78-85.

Awang, N.F., A.A. Manaf and W.S. Zainudin, 2014. A survey on conducting vulnerability assessment in web-based application. Commun. Comput. Inf. Sci., 488: 459-471.

Bisht, P., P. Madhusudan and V.N. Venkatakrishnan, 2010. Candid: Dynamic candidate evaluations for automatic prevention of SQL injection attacks. ACM Trans. Inf. Syst. Security, 5: 1-39.

Bozic, J. and F. Wotawa, 2014. Security testing based on attack patterns. Proceedings of the IEEE 7th International Conference on Software Testing, Verification and Validation Workshops, March 31-April 4, 2014, Cleveland, OH., pp: 4-11.

Chen, J.M. and C.L. Wu, 2010. An automated vulnerability scanner for injection attack based on injection point. Proceedings of the International Computer Symposium, December 16-18, 2010, Tainan, pp: 113-118.

Djuric, Z., 2013. A black-box testing tool for detecting SQL injection vulnerabilities. Proceedings of the 2013 2nd International Conference on Informatics and Applications, September 23-25, 2013, Lodz, pp: 216-221.

Ezumalai, R. and G. Aghila, 2009. Combinatorial approach for preventing SQL injection attacks. Proceedings of the International Advance Computing Conference, March 6-7, 2009, Patiala, India, pp: 1212-1217.

Halfond, W.G., J. Viegas and A. Orso, 2006. A classification of SQL-injection attacks and countermeasures. Proceedings of the IEEE International Symposium on Secure Software Engineering, March 13-15, 2006, Washington, DC., USA.

He, K., Z. Feng and X. Li, 2008. An attack scenario based approach for software security testing at design stage. Proceedings of the International Symposium on Computer Science and Computational Technology, December 20-22, 2008, Shanghai, pp: 782-787.

Kindy, D.A. and A.S.K. Pathan, 2013. A detailed survey on various aspects of SQL injection in web applications: Vulnerabilities, innovative attacks and remedies. Int. J. Commun. Networks Inf. Secur., 5: 80-92.

Lei, L., X. Jing, L. Minglei and Y. Jufeng, 2013. A dynamic SQL injection vulnerability test case generation model based on the multiple phases detection approach. Proceedings of the 2013 IEEE 37th Annual Computer Software and Applications Conference, July 22-26, 2013, Kyoto, pp: 256-261.

Stuttard, D. and M. Pinto, 2007. The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws. Wiley Publishing, New York, USA.

Wassermann, G., D. Yu, A. Chander, D. Dhurjati, H. Inamura and Z. Su, 2008. Dynamic test input generation for web applications. Proceedings of the 2008 International Symposium on Software Testing and Analysis, July 20-24, 2008, Seattle, WA., USA., pp: 249-259.

Wodarz, P.N., 2008. Algorithms for generating permutations and combinations. http://www4.uwsp.edu/math/nwodarz/Math209Files/209-0809F-L10-Section06_03-AlgorithmsForGeneratingPermutationsAndCombinations-Notes.pdf.