Security Testing of Web Applications: A Search-Based Approach for Detecting SQL Injection Vulnerabilities

Muyang Liu
University of Electronic Science and Technology of China
Chengdu, China
muyangliu@std.uestc.edu.cn

Ke Li
University of Exeter
Exeter, UK
k.li@exeter.ac.uk

Tao Chen
Nottingham Trent University
Nottingham, UK
tao.chen@ntu.ac.uk

ABSTRACT

Web applications have become increasingly essential in many domains that operate on confidential data related to business. SQL injection attack is one of the most significant web application security risks. Detecting SQL injection vulnerabilities is essential for protecting the underlying web application. However, manually enumerating test cases is extremely challenging, if not impossible, given the potentially infinite number of user inputs and the likely nonexistence of one-to-one mapping between user inputs and malicious SQL statements. This paper proposes an automatic security test case generation approach to detect SQL injection vulnerabilities for web applications, following a search-based software engineering (SBSE) paradigm. Particularly, we propose a novel fitness function that evaluates the similarity between the SQL statements produced by feeding user inputs in the system under test and a known malicious SQL statement. For the search algorithm, we exploit differential evolution, which is robust in continuous optimization but it is under-investigated in SBSE. Based on three real-world web applications, we conduct experiments on 19 configurations that are of diverse forms of SQL statements and types of attacks. Results demonstrate that our approach is more effective, with statistical significance and high effect sizes, than the state-of-the-art.

CCS CONCEPTS

• **Software and its engineering → Software reliability**; **Search-based software engineering**;

KEYWORDS

Security, SQL injection, test case generation, search-based software engineering, differential evolution

ACM Reference Format:
Muyang Liu, Ke Li, and Tao Chen. 2019. Security Testing of Web Applications: A Search-Based Approach for Detecting SQL Injection Vulnerabilities. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3322026

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3322026

1 INTRODUCTION

Web applications have become increasingly ubiquitous and important in various domains [3]. They often host a large amount of confidential data. Unfortunately, they are vulnerable to a variety of security threats. In particular, SQL injection attack (SQLIA) is recognized as one of the most significant web application security risks according to Open web Application Security Project[1]. Typically, the SQL injection is to cheat the server to execute malicious SQL commands by inserting SQL commands into user inputs.

Since SQLIAs are derived from user inputs, one simple but widely used defending technique is input validation that escape user inputs through security coding [5]. Note that input validation largely depend on the underlying functionality logic of the underlying web application. There are not exist *thumb rules* to design universally effective input validation. Furthermore, since user inputs can, in principle, be infinite, it is difficult (if not impossible) to enumerate a comprehensive set of test cases (i.e., user inputs) to fully test the vulnerabilities of the underlying web application. Search-based software engineering (SBSE) [4], which has shown strong performance on automated software test case generation [2], is a promising baseline for automating the security test in web applications. For example, in [6], search-based approaches are proposed to exploit XML injection vulnerabilities in an automatic manner. Inspired by this work, we propose a search-based approach to automatically generate security test cases for detecting SQL injection vulnerabilities. To facilitate our experiments, malicious SQL statements for the underlying web application are known a priori by using the SQL injection application benchmark developed by Halfond et al. [3]. However, generating the user inputs that perfectly match the malicious SQL statement is very challenging, because a malicious SQL statement can be derived from various user inputs and the input validation implies that there is often no one-to-one mapping between the user inputs and the SQL statements produced by the web application.

In this paper, we propose a new fitness function, called similarity matching distance (SMD), to evaluate the closeness between the SQL statement produced by executing the user input across the underlying web application and a known malicious SQL statement. In particular, we treat this closeness evaluation as a spelling error check problem that covers insert, deletion, replacement and transpose of adjacent characters with respect to input strings.In addition, we exploit the power of differential evolution (DE) to serve as the search algorithm for detecting SQL injection vulnerabilities.

[1] https://www.owasp.org/

Note that DE has been recognized as one of the most successful algorithms for continuous problems whereas it is unfortunately under-exploited in the SBSE community.

2 A SEARCH-BASED APPROACH

Our approach generate a comprehensive set of test cases, which mimic potential user inputs, to validate the vulnerabilities of the system under test (SUT) including both the front- and back-end.

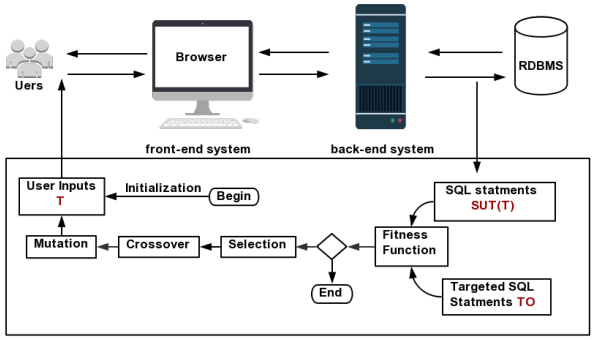

Figure 1: Work flow of our proposed security test case generation approach.

Figure 1 provides the work flow of our proposed search-based security test case generation approach. Specifically, we apply DE, a robust search algorithm to automatically generate user inputs that can lead to SQLIAs. During the search process, the SUT is fed with various user inputs, denoted as T, generated by our proposed search algorithm. After the execution of the SUT, the generated SQL statement, denoted as SUT(T), is compared with the known malicious SQL statement, denoted as TO.

In our security test case generation scenario, the ultimate goal is to find a solution T that leads to a SQL statement matching the TO after being executed across SUT. Therefore, the fitness function measures the distance between TO and SUT(T). Inspired by the real-coded edit distance (RD) developed in [6], we develop a new distance measure called similarity matching distance (SMD) to serve our purpose.

Instead of editing operations, measuring the distance between SUT(T) and TO can also be treated as checking the spelling errors caused by SUT(T). As discussed in [1], insert, deletion, replacement and transpose of adjacent characters cover 80% of the causes of spelling errors. Therefore, we add the transformation operations that RD(Real) [6] does not take into account. Specifically, given two strings S_l and $\hat{S}_{l'}$, their SMD, denoted as $d_{\text{SMD}}(S_l, \hat{S}_{l'})$, is calculated as:

$$d_{\text{SMD}}(S_l, \hat{S}_{l'}) = (1 - L \times p) \times d(S_l, \hat{S}_{l'}) \tag{1}$$

$$d(S_l, \hat{S}_{l'}) = \min \begin{cases} d(S_{l-1}, \hat{S}_{l'}) + 1, \\ d(S_l, \hat{S}_{l'-1}) + 1, \\ d(S_{l-2}, \hat{S}_{l'-2}) + \frac{|s_l - \hat{s}_{l'}|}{1+|s_l - \hat{s}_{l'}|}, if\ s_{l-1} = \hat{s}_{l'} \ \&\ s_l = \hat{s}_{l'-1} \\ d(S_{l-1}, \hat{S}_{l'-1}) + \frac{|s_l - \hat{s}_{l'}|}{1+|s_l - \hat{s}_{l'}|} \end{cases} \tag{2}$$

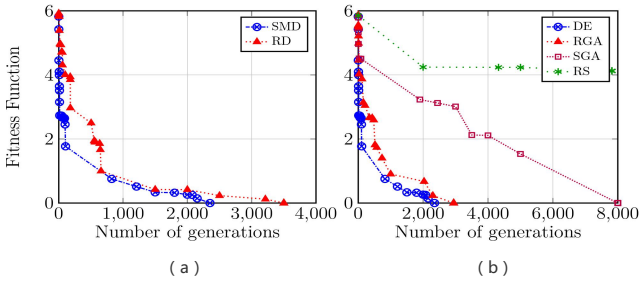

Figure 2: Convergence rate of a TO for (a) DE with SMD and RD; (b) DE, RGA, SGA, RS with SMD.

where $p \in [0, 1]$ is a parameter and L is the number of same characters S_l and $\hat{S}_{l'}$ have from the beginning position.

3 EXPERIMENT AND ANALYSIS

Based on three real-world web applications, we conduct experiments on 158 TOs in configurations that are of diverse forms of SQL statements and types of attacks. As shown in Figure 2, we randomly choose a TO and plot the trajectories across the number of generations. we can see that the algorithm converges faster by using the SMD in Figure 2(a) . This can explained as the RD does not change for a long time when it is close to 0. In contrast, the SMD can differentiate those strings which are very similar to each other. Thus, it is more effective to accelerate the convergence. Figure 2(b) is clear that DE is consistently better than the other peer algorithms across the search process. In particular, the trajectory of DE is the steepest compared to the others. This indicates that the convergence speed of DE is the fastest, where it only cost approximates around a quarter of the given computational budgets to the optimum.

4 CONCLUSION

In this paper, we propose a search-based security test case generation approach to automatically detect SQL injection vulnerabilities in web applications. In particular, we propose a novel fitness function, namely SMD. Unlike existing work, we exploit the power of DE to serve as the search engine. In the future work, we will expand the approach to take the semantic aspects of the SQL statements into account.

REFERENCES

[1] Fred Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM* 7, 3 (1964), 171–176.
[2] Saswat Anand et al. 2013. An orchestrated survey of methodologies for automated software test case generation. *Journal of Systems and Software* 86, 8 (2013), 1978–2001.
[3] Halfond, William, Orso, Alex, Manolios, and Pete. 2008. WASP: Protecting web applications using positive tainting and syntax-aware evaluation. *IEEE Trans. Software Engineering* 34, 1 (2008), 65–81.
[4] Mark Harman, S Afshin Mansouri, and Yuanyuan Zhang. 2012. Search-based software engineering: Trends, techniques and applications. *Comput. Surveys* 45, 1 (2012), 11.
[5] Michael Howard and David LeBlanc. 2003. *Writing secure code.* Pearson Education.
[6] Sadeeq Jan, Annibale Panichella, Andrea Arcuri, and Lionel Briand. 2017. Automatic Generation of Tests to Exploit XML Injection Vulnerabilities in Web Applications. *IEEE Trans. Software Engineering* (2017).