ISBN 978-981-11-3671-9
Proceedings of 2017 the 7th International Workshop on Computer Science and Engineering (WCSE 2017)
Beijing, 25-27 June, 2017, pp. 569-573
doi: 10.18178/wcse.2017.06.098

A SQL Injection Vulnerability Penetration Test Approach Based on Response-Driven Attacking Model

Lei Liu[1], Jing Xu[1+], Biao Zhang[1], Jiehui Kang[1], Chenkai Guo[1], Sihan Xu[1] and Guannan Si[2]

[1] College of Computer and Control Engineering, Nankai University, Tianjin, China

[2] School of Information Science and Electrical Engineering, Shandong Jiaotong University, Jinan, China

Abstract. Penetration Test is one of the most important test techniques for exposing the top-ranked web vulnerability SQL Injection Vulnerability (SQLIV). However, the insufficiency of testing accuracy and efficiency has become increasingly severe since the continual emerging of many new kinds of SQLIV types with different response analyzing methods. To solve this problem, this paper proposes a Response-Driven Attacking Model (RDAM), which is based on systematic response analyzing and attacking tree model. The empirical study shows the effectiveness of the proposed approach.

Keywords: response analyzing, attacking tree, model based, SQL injection vulnerability, penetration test

1. Introduction

SQL Injection Vulnerability (SQLIV) has become the top-ranked security threat to web application according to OWASP [1], which may cause serious security problems, such as information leakage or authentication by passing [2]. Penetration test is one of the most important test techniques for exposing SQLIV because of its high usability and efficiency, which is a kind of black-box test that detects vulnerabilities through tester's mock attacking before real attacks [3].

However, along with the continual emerging of many new kinds of SQLIVs that must be identified by different response analysing methods, the testing complexity and difficulty are increased sharply. Because black-box penetration test exposes vulnerabilities by analysing testing HTTP responses, the complexity between different test cases and their specific responses may cause the insufficiency of testing coverage and accuracy, especially reflected in high False Negatives (FN) and False Positives (FP) [4].

Many former researches focus on improving SQLIV penetration test coverage and accuracy by improving the response analysing capabilities [5]. Some approaches use combined static&dynamic techniques to increase the ability of analysing responses by analysing and altering source code or background servers in a white-box way [6][7]. But the need of source code greatly limits its applicable scopes in a lot of real scenarios that cannot access the source code or background servers. A lot of black-box penetration test methods apply experienced error patterns or information extraction technology to analyse test responses, which usually conduct their test cases or test patterns in sequence [8][9]. However, the traditional black-box penetration test methods in enumeration way hardly cope with the increasing complexity of different kinds of responses of different test cases. Meanwhile, more combinations of test cases and responses increase test scale dramatically, which also bring about the trade-off problem between coverage, accuracy and efficiency.

To solve the above problems, this paper proposes a Response-Driven Attacking Model (RDAM) for SQLIV penetration test, which is built on testing response analysing and attacking tree model. The entire RBAM testing procedure is composed of multiple Response-Based Test Stages (RBT Stage), which include a series of Response-Based Test Units (RBT Unit).

+ Corresponding author: Jing Xu. Tel.: +86 022-23500350;
E-mail address: xujing@nankai.edu.cn

The remainder of this paper is structured as follows: Section 2 and Section 3 present the backgrounds and basic definitions. Section 4 describes the proposed Response-Driven Attacking Model (RDAM). Section 5 gives the empirical study and its result analysis. Finally, the conclusion is presented in Section 6.

2. Backgrounds

SQLIV is launched through specially designed input on database-driven web application, which occurs when attacker's inputs with modified SQL clauses can be injected to a SQL query that is transferred to the background database [5]. SQLIV can provoke a series of serious security problems, such as authentication bypassing, information leakage, database damaging or even system hijacking, etc [6]. SQLIV can exist in parameters of URL (GET method), inputs of webpage FORM (POST method), HTTP Cookie or Header parameters. A classic example of SQLIV is as follow:

URL Injection Point: *HTTP://www.VulWeb.com/test?customer=Geng&id=123* ***AND 1=1***

In this case, the injection point is in the parameter "id=123" in a URL, and the test case (injection payload) is "AND 1=1 --", and the altered background SQL query transferred to the database is:

Background Query: *SELECT * FROM Costumers WHERE customer = Geng AND id = 123* ***AND 1=1***

The original SQL query is the part without payload "AND 1=1". Obviously, the logic of the altered SQL query is the same as the original query, because the tautology clause "AND 1=1" cannot change the logic of a SQL statement. If we change the test case to "AND 1=2", the HTTP response will change because of the changing logic caused by contradictory clause "AND 1=2". Through observation of the similarity change of HTTP responses, we can obtain the existence evidence of a SQLIV.

Penetration Test is one of the most used black-box vulnerability testing techniques to test SQLIV in real practice, which can simulate attackers' inputs to expose SQLIV dynamically. Because of lacking background information, penetration test exposes SQLIV through the analysing of the testing HTTP responses. From the aspects of response analysing, there are three main kinds [5]:

- **Error-Based Technique:** analyzing the database errors in testing HTTP responses caused by the violation of SQL syntax during penetration test.
- **Similarity-Based Technique:** analyzing the similarity variation of testing HTTP responses caused by the logic changing of the target SQL query.
- **Time-Based Technique:** analyzing the time delay of testing HTTP responses caused by time or calculating related function of the SQL query in test cases or payloads.

Table 1: Response Symbols of SQLIV Penetration Test

Symbol	Response Name	Response Description
ERR	Database Error	Including a database error in the testing HTTP response
NERR	No Database Error	Including no database error in the testing HTTP response
SIMI	Similar to Origin	The testing HTTP response is similar to the original HTTP response
DIFF	Different to Origin	The testing HTTP response is different to the original HTTP response
DLY	Time Delay	The testing HTTP response time is apparently delayed
NDLY	No Time Delay	The testing HTTP response time is not delayed
NE	No Evidence	There is no evidence of SQLIV in a unit test or a test stage
VE	Vulnerability Evidence	There are evidences of SQLIV in a unit test or a test stage

3. Basic Definitions

This paper proposes a penetration test approach based on Response-Driven Attacking Model (RDAM), from which we can obtain testing strategies based on the analysing of testing responses. To describe testing responses formally, we introduce eight symbols, as showed in Table. 1. Symbols "ERR" and "NERR" are Error-Based response analysing symbols, the "SIMI" and "DIFF" are Similarity-Based response symbols, and "DLY" and "NDLY" are Time-Based symbols. We also introduce "VE" and "NE" to present

intermediate results of a test unit and a test stage as a type of testing response, as showed in Table. “VE” means the possible evidence of a SQLIV, which is also defined as the response of a test unit or test stage. “VN” means that there is no evidence of SQLIV in this test unit or test stage.

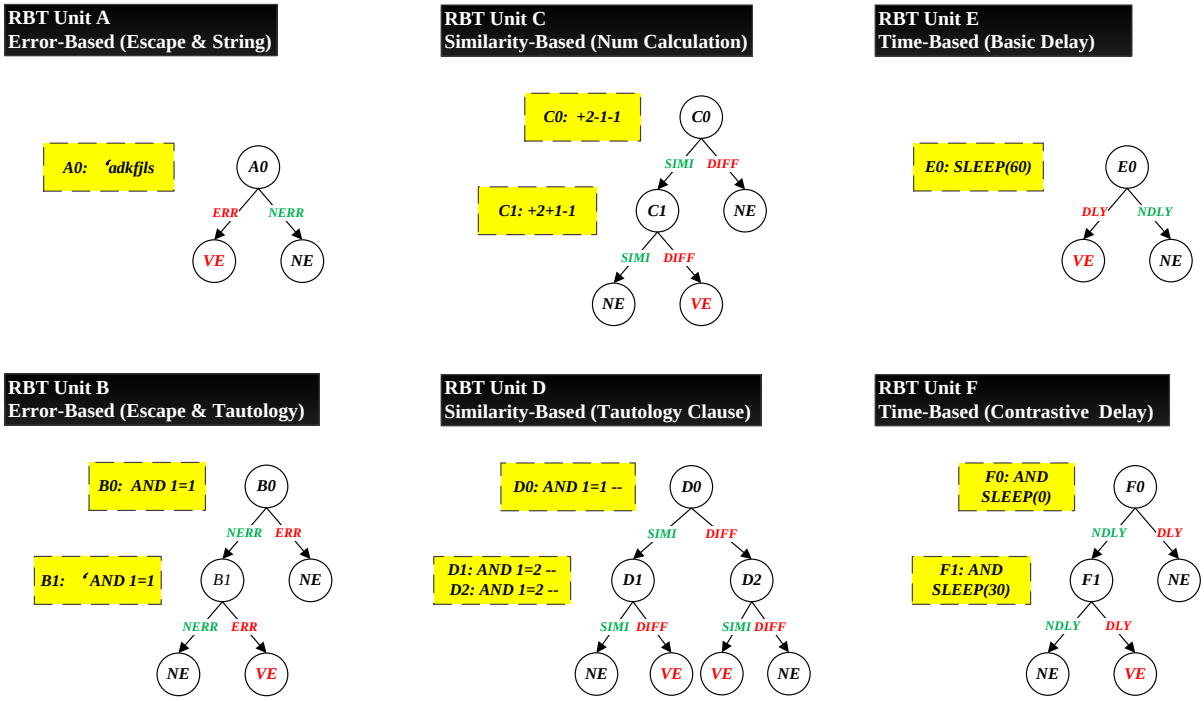

Fig. 1: Examples of the Response-Based Test Unit (RBT Unit)

4. Response-Driven Attacking Model (RDAM)

4.1. Response-Based Test Units and Stages (RBT Unit and RBT Stage)

In SQLIV penetration test, testers usually use a series of test patterns to identify the existence of vulnerabilities, which include multiple test cases (payloads), such as the pattern composed by “AND 1=1 → SIMI” and “AND 1=2 → DIFF”. Traditional SQLIV penetration test patterns usually are concluded by experiences, which are insufficient and not suitable for complicated situations.

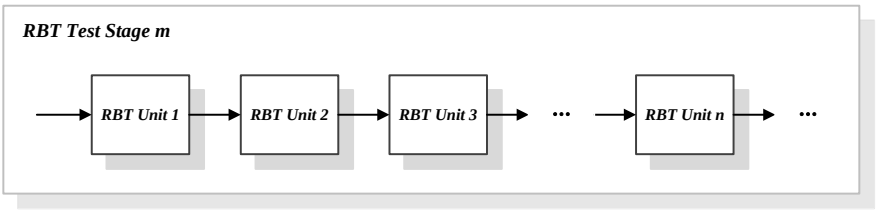

Fig. 2: Response-Based Test Stage (RBT Stage)

In this paper, we define Response-Based Test Unit (RBT Unit) to describe SQLIV penetration test patterns, which can analyse the complicated regularity of test cases and HTTP responses, as showed in Fig. 1. Each RBT Unit is a response based binary tree, in which a node (including the root node) represents a test case, an edge describes a kind of response of its father node, and a leaf node represents the intermediate result of this test unit. The leaf nodes include “VE” and “VN”, which also are the response of the entire Response-Driven Attacking Model (RDAM). A route to leaf node represents the combination of test cases and their corresponding responses to a intermediate result “VE” or “VN”, such the route “D0 → SIMI → D1 → VE” in RBT Unit D of Fig. 1 represents one procedure to discover the possible evidence of a SQLIV. Fig. 1 presents examples of RBT Units, including Error-Based Unit (Unit A and B), Similarity-Based Unit (Unit C and D) and Time-Based Unit (Unit E and F). For instance, RBT Unit D indicates a Similarity-Based tautology testing unit, including test cases “AND 1=1--” (D0) and “AND 1=2--” (D1 and D2). There are two routes have the leaf nodes that represent vulnerability evidence “VE”. This paper introduces Response-Based

Test Stage (RBT Stage) to present a test phase, as showed in Fig. 2. Each RBT Stage contains a number of RBT Units with similar characteristics. RBT Units in a RBT Stage are performed in sequence.

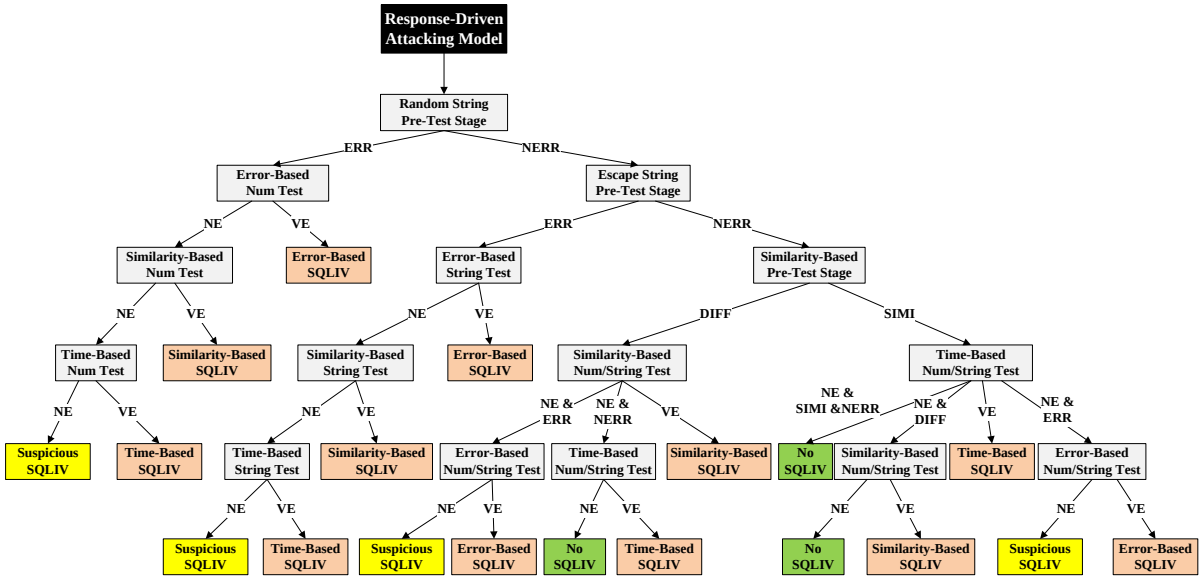

Fig. 3: Response-Driven Attacking Model

4.2. Response-Driven Attacking Model (RDAM)

Based on the defined RBT Stages, this paper introduces Response-Driven Attacking Model (RDAM) for SQLIV penetration test, as showed in Fig. 3. In essence, the RDAM is in the form of attacking tree based on stage testing responses, including HTTP responses and middle test evidences ("VE" and "VN"). In the introduced attacking tree model RDAM, a node (excluding leaf nodes) represents a test stage, an edge describes responses of the previous test stage, and a leaf node represents the final test result of the entire penetration test for one injection point.

Each RBT Stage of RDAM includes a set of RBT Unit which has the same features or functions. For response edges in RBAM, we classify them to two kinds: Positive Responses (including ERR, DIFF, DLY and VE) and Negative Responses (including NEER, SIMI, NDLY, NE). The edges corresponding to Positive Responses mean that at least one of the specific RBT Unit responses occurs during the previous RBT Stage. The edges corresponding to Negative Responses mean that there is no specific RBT Unit response occurs during the previous RBT Stage. One edge may contain a combination of multiple types of response, such as "NE & SIMI & NERR". There are five kinds of leaf nodes that represent final test result, in which "Error-Based SQLIV", "Similarity-Based SQLIV" and "Time-Based SQLIV" represent a specific type of SQLIV discovered, "Suspicious SQLIV" means there is a suspicion of SQLIV existing that need to be verified manually, and "No SQLIV" means there is no vulnerability found in this injection point.

Fig. 3 shows the entire model of RDAM. There are three Pre-Test stages (including "Random String Pre-Test Stage", "Escape String Pre-Test Stage" and "Similarity-Based Pre-Test Stage") to decide which kind of Response-Based stage we should conduct. Different RBT Stages and their distinctive responses lead to different strategies. Finally, each path has its leaf nodes corresponding to the final test result of one injection point. We can test every injection point in sequence to expose SQLIV effectively in a website.

5. Empirical Study

To conduct the empirical study, we develop a prototype tool to apply Response-Driven Attacking Model (RDAM) in the environment of "Visual Studio 2010 + .net 3.5 + C#". Our prototype can detect the injection points in a website sequential, which are crawled by a third-party spider tool before the experiment. Two state-of-the-practice benchmarking tools are chosen to compare with our approach, including WVS Acunetix[1], IBM Appscan[2], which are referred as Tool A and Tool B (without particular order) to avoid

[1] http://www. cunetix .com/ vulnerabilityscanner
[2] http://www.ibm.com/developerworks/cn/downloads/r/appscan

brand comparison and protect their confidentiality. We also develop a web application *VWebShop* to conduct our empirical study, which has 570 URLs and is implanted 36 different SQLIVs in 11 SQLIV types.

The detailed empirical study results are showed in Table. 2. In the table, "Vul#" represents the number of implanted SQLIV in the target web application, "Rqst#" is the HTTP requests sent by each tool which can assess the testing efficiency, "RP#" is the Real Positive number tested, "FN#" is the False Negative number produced by the tool, "RCR" is the ReCall Rate (*RCR=RP#/Vul#*), "FNR" is the False Negative Rate (*FNR = FN#/Vul#*), "FPR" is the False Positive Rate (*FPR=FP#/(FP#+RP#)*). From the experiment results, we can see that the proposed approach recall all the SQLIVs, and has lower False Negative Rate (2.8%) and False Positive Rate (7.9%). The efficiency is in an applicable range. The results obtained so far are promising, which prove the effectiveness of the proposed RDAM model.

Table 2: Empirical Study Result

Comparison Tools	Vul#	Rqst#	RP#	FN#	FP#	RCR	FNR	FPR
Our Approach	36	33459	35	1	3	97.2%	2.8%	7.9%
Tool A	36	49895	29	7	8	80.6%	19.4%	21.6%
Tool B	36	23480	32	4	6	88.9%	11.1%	15.8%

6. Conclusions

This paper presents a SQL Injection Vulnerability (SQLIV) penetration test approach based on Response-Driven Attacking Model (RDAM). Firstly, this paper introduces the background and basic feedback definitions of the proposed model. Secondly, Response-Based Test Unit (RBT Unit) and Response-Based Test Stage (RBT Stage) are defined. Based on RBT Unit and Stage, we propose RDAM using responsive feedback analyzing and attacking tree model. The empirical study shows the effectiveness of RDAM. In the future work, we will apply RDAM for the testing of more security vulnerability types.

7. Acknowledgements

This research work here is supported by National Natural Science Foundation of China (Grant No. 61402264) and the Science and Technology Planning Project of Tianjin (Grant No. 17JCZDJC30700).

8. References

[1] OWASP. Top Ten Most Critical Web Application Security Risks. *https://www.owasp.org*. Reviewed in 2017.

[2] G. Deepa, P. S. Thilagam. Securing web applications from injection and logic vulnerabilities: Approaches and challenges. *Information and Software Technology*. 2016, 74: 160-180.

[3] N. Antunes, and M. Vieira. Evaluating and Improving Penetration Testing in Web Services. *23rd IEEE International Symposium on Software Reliability Engineering (ISSRE 2012)*. November, 2012: 27-30.

[4] Antunes N., Vieira M. Assessing and Comparing Vulnerability Detection Tools for Web Services: Benchmarking Approach and Examples. *IEEE Trans on Services Computing*. 2015, 8(2): 269-283.

[5] Lawal M. A., Abu B. M. S., Ayanloye O., et al. Systematic Literature Review on SQL Injection Attack. *International Journal of Soft Computing*. 2016, 11(1): 26-35.

[6] W. G. J. Halfond, S. R. Choudhary, A. Orso. Improving penetration testing through static and dynamic analysis. *Software Testing, Verification and Reliability*. 2011, 21(3): 195-214.

[7] N. Antunes, and M. Vieira. Evaluating and Improving Penetration Testing in Web Services. *23rd IEEE International Symposium on Software Reliability Engineering (ISSRE 2012)*. November, 2012: 27-30.

[8] Ciampa A., Visaggio C. A., Di Penta M.. A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications. *Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, ACM.* 2010: 43-49.

[9] Huang YW, Huang SK, Lin TP, et al. Web application security assessment by fault injection and behavior monitoring. *Proceedings of the 12th international conference on World Wide Web, ACM.* 2003: 148-159.