Model Based Hybrid Approach to Prevent SQL Injection Attacks in PHP

Kunal Sadalkar, Radhesh Mohandas, and Alwyn R. Pais

Department of Computer Science and Engineering
National Institute of Technology Karnataka, Surathkal,India - 575025
kunalsadalkar@yahoo.co.in, {radhesh,alwyn.pais}@gmail.com

Abstract. SQL Injection vulnerability is ranked 1st in the OWASP[1] top 10 vulnerability list and has resulted in massive attacks on a number of websites in the past few years. Inspite of preventive measures like educating developers about safe coding practices, statistics shows that these vulnerabilities are still dominating the top. Various static and dynamic approaches have been proposed to mitigate this vulnerability. In this paper, we present a hybrid approach to prevent SQL injection attacks in PHP, a popular server side scripting language. This technique is more effective to prevent SQL injection attack in a dynamic web content environment without use of complex string analyzer logic. Initially, we construct a Query model for each hotspot by running the application in safe mode. In the production environment, dynamically generated queries are validated with it. The results and analysis shows the proposed approach is simple and effective to prevent common SQL injection vulnerabilities.

Keywords: SQL injection attack, static analysis, dynamic analysis, web vulnerabilities, unauthorized access, authentication bypass,input validation,database mapping.

1 Introduction

The industry is moving fast towards web based technologies. Ubiquity and cost effective remote services are the driving factors for the growth of the web based industries. Companies and organizations are adapting these upcoming technology to reduce cost and to satisfy their customers. Services like online shopping, e-banking, e-reservation, e-governance etc. have made daily life more productive. In a race to support more and more flexible solutions, web developers are skipping steps of best coding practices leading to serious web vulnerabilities. Lack of proper code review and increased complexity of configuring the security policies of web applications, enable malicious users to misuse these services and achieve monitory gains. Testing for these security vulnerabilities requires considerable time and resources and even known attacks are not addressed completely.

[1] Open Web Application Security Project.

M. Joye et al. (Eds.): InfoSecHiComNet 2011, LNCS 7011, pp. 3–15, 2011.

Nowadays, PHP is well-known for server-side web development and widely used general-purpose scripting language on 75% of web servers. The overall proportion of PHP-related vulnerabilities on the database amount to: 20% in 2004, 28% in 2005, 43% in 2006, 36% in 2007, 35% in 2008, and 30% in 2009 [12,13]. According to SANS [14] statistics, the total numbers of vulnerabilities exploited in web applications are more than those of stand-alone system vulnerabilities. SQL Injection attacks continue to remain among the top three popular techniques used for compromising web sites. In SQL injection vulnerability the data from untrusted sources is used in a trusted manner and that allows execution of unintended queries.

Although the causes for SQL injection have been known for a long time and various solutions including defensive coding practices, static code reviews and runtime checks have been proposed, the problem persists for several reasons. Human code reviews are time consuming and expensive. Moreover the quality of code review strongly depends on the expertise of the reviewer. Defensive programming and input filtration prevents from malicious command to get executed, but this approach is fairly oblivious to new unanticipated patterns. Performance of runtime checks is sensitive to the size of the applications and also increases the false positive rates as the application logic gets updated. Finally any solution based on hybrid technique [1] to prevent SQL injection attacks suffers from the limitation of string analysis precision and execution performance penalty.

This paper addresses the comparative analysis of static and dynamic approach, highlights the inability of existing hybrid approaches[2] to prevent SQL injection attacks in PHP. Furthermore we propose a new alternative model based approach to counter the SQL injection attack in PHP. An empirical analysis shows that the proposed approach is extremely simple and does not need complex logic of static analysis based on a string analyzer.

2 Related Work

Our approach is a simple variant of the existing model based hybrid approach AMNeSIA [2]. This approach combines static analysis and runtime monitoring. In the static phase, it builds a query model for legitimate SQL with the help of java string analyzer (JSA)[8]. Query models are constructed as NDFA (Non Deterministic Finite Automata) whose nodes are SQL keywords and operators with special symbols for user input. During the run time, queries are intercepted with the instrumented code and crosschecked with the statically built query models. A limitation of AMNeSIA[2] tool is that it cannot be used for web applications other than those built on JSP. As the tool makes implicit use of JSA[8] library to build query model, the proposed approach does not work for PHP applications. Moreover this tool's success is dependent on the accuracy of the string analyzer.

Similarly JDBC checker[9] statically validates the correctness of dynamically generated queries. SQLDOM [7] and Safe Query Object [15] make use of encapsulation of database query in order to access the database safely. But in these