Detection and Prevention of SQL Injection Attacks Using Semantic Equivalence

Sandeep Nair Narayanan, Alwyn Roshan Pais, and Radhesh Mohandas

Department of Computer Science & Engineering,
National Institute of Technology Karnataka,
Surathkal, India
{sandeepnairnarayanan,alwyn.pais,radhesh}@gmail.com

Abstract. SQL injection vulnerability is a kind of injection vulnerability in which the database server is forced to execute some illicit operations by crafting specific inputs to the web server. Even though this vulnerability has had it's presence for several years now, most of its popular mitigation techniques are based on safe coding practices, which are neither applicable to the existing applications, nor are application independent. Here we propose a new application logic independent solution to prevent SQL injection attacks which can be applicable to any dynamic web technology. The new solution detects SQL injection by considering the semantic variance between the queries generated by the query function with safe inputs and injection inputs. We have implemented the complete solution in ASP.NET with C# web applications using a custom written tool, SIAP, which patches the SQL Injection vulnerabilities in an existing web application by instrumenting the binaries.

Keywords: Injection Attacks, Injection Vulnerability, SIA, SQL Injection, Web Technology.

1 Introduction

SQL Injection attack is a type of code injection exploit in which the attacker gets unauthorized access to data or bypasses authentication by injecting crafted strings through web inputs. In web applications, some of the data from the web forms are used for constructing Structured Query Language (SQL) statements and are then executed on the database server. In a SQL Injection Attack(SIA), the attacker injects data into the web application, in such a way that the resultant SQL statement will produce malicious outcomes. In advanced SQL injection techniques, the queries are crafted to produce syntax errors on execution. The information disclosed in the error messages are used further for crafting more specific SQL statements which eventually leads to the finger printing of the complete database schema of the application. Injection vulnerability has been listed as the top vulnerability by OWASP [1]. Due to the high impact of the attack and the low cost and skill required to launch the attack, the net risk associated with

© Springer-Verlag Berlin Heidelberg 2011

SQL injection vulnerabilities are very high. Even after the attack being well known, there are many enterprise web sites in the Internet which are still vulnerable to such attacks.

The popular solutions for the prevention of SQL injection attacks include coding best practices, input filtering, escaping user input, usage of parameterized queries, implementation of least privilege, white list input validation etc., These solutions should be employed usually during the development of an application. This is the major limitation of such solutions as they do not cover the millions of Web applications already deployed with this vulnerability. Manual patching of the vulnerability at each possible point is quite expensive with regard to the incurred cost and time consumption. This signifies the need for an automated tool to patch this vulnerability.

2 Related Work

We broadly classify the solutions for the prevention of SQL injection attacks as Developer centric solutions and Maintenance centric solutions. In the developer centric solution, the developer tries to prevent the attacks. Following safe coding practices [2], randomized query based methods and using Plug-in's which detect spots of possible vulnerabilities during the development phase are examples of such kind of solutions. SQLrand [3] is a technique in which the developer uses some randomized instructions instead of normal keywords. SAFELI [4] is a framework which uses a symbolic execution engine, a library of attack patterns and a constraint solver to detect the vulnerabilities in a web application. The drawbacks with the Developer centric approach are the need for individual attention for every project to be patched, the cost associated and the inability to handle the existing applications without modifying the code.

In Maintenance centric solutions we patch the deployed application using a third party tool or add new components to the existing system to prevent the attacks. Solutions in this category include Data tainting based solutions [5], Intrusion Detection System (IDS) based solutions, Black box testing solutions, Machine learning based techniques etc., Waves [6] is a black box testing tool which uses machine learning techniques to test for SQL injection vulnerabilities. Some techniques like SQL Guard [7] and SQL Check [8] check the queries for SQLI vulnerabilities based on some models. Another solution is using AMNESIA [9] [10] which uses a hybrid approach having both static analysis for building the model and dynamic analysis for preventing the injection. Yet another solution is CANDID [11] in which the intended query is created by running the application on candidate inputs. Our solution is a Maintenance centric solution, which is applicable to existing applications and can also be used during the development of new applications.