Chapter 10
Retrofitting Existing Web Applications with Effective Dynamic Protection Against SQL Injection Attacks

San-Tsai Sun
University of British Columbia, Canada

Konstantin Beznosov
University of British Columbia, Canada

ABSTRACT

This paper presents an approach for retrofitting existing Web applications with run-time protection against known, as well as unseen, SQL injection attacks (SQLIAs) without the involvement of application developers. The precision of the approach is also enhanced with a method for reducing the rate of false positives in the SQLIA detection logic, via runtime discovery of the developers' intention for individual SQL statements made by Web applications. The proposed approach is implemented in the form of protection mechanisms for J2EE, ASP.NET, and ASP applications. Named SQLPrevent, these mechanisms intercept HTTP requests and SQL statements, mark and track parameter values originating from HTTP requests, and perform SQLIA detection and prevention on the intercepted SQL statements. The AMNESIA testbed is extended to contain false-positive testing traces, and is used to evaluate SQLPrevent. In our experiments, SQLPrevent produced no false positives or false negatives, and imposed a maximum 3.6% performance overhead with 30 milliseconds response time for the tested applications.

INTRODUCTION

SQL injection attacks *(SQLIAs)* are one of the foremost threats to Web applications (Halfond, Viegas, & Orso, 2006). According to the WASP Foundation, injection flaws, particularly SQL injection, were the second most serious type of Web application vulnerability in 2008 (OWASP, 2008). The threats posed by SQLIAs go beyond simple data manipulation. Through SQLIAs, an attacker may also bypass authentication, escalate privileges, execute a denial-of-service attack, or execute remote commands to transfer and install malicious software. As a consequence of SQLIAs,

DOI: 10.4018/978-1-4666-1580-9.ch010

parts of entire organizational IT infrastructures can be compromised. As a case in point, SQLIAs were apparently employed by Ehud Tenenbaum, who has been arrested on charges of stealing $1.5M from Canadian and at least $10M from U.S. banks (Zetter, 2009). An effective and easy to employ method for protecting numerous existing Web applications from SQLIAs is crucial for the security of today's organizations.

State-of-the-practice SQLIA countermeasures are far from effective (Anley, 2002) and many Web applications deployed today are still vulnerable to SQLIAs (OWASP, 2008). SQLIAs are performed through HTTP traffic, sometimes over SSL, thereby making network firewalls ineffective. Defensive coding practices require training of developers and modification of the legacy applications to assure the correctness of validation routines and completeness of the coverage for all sources of input. Sound security practices—such as the enforcement of the principle of least privilege or attack surface reduction—can mitigate the risks to a certain degree, but they are prone to human error, and it is hard to guarantee their effectiveness and completeness. Signature-based Web application firewalls—which act as proxy servers filtering inputs before they reach Web applications—and other network-level intrusion detection methods may not be able to detect SQLIAs that employ evasion techniques (Maor & Shulman, 2005).

Detection or prevention of SQLIAs is a topic of active research in industry and academia. An accuracy of 100% is claimed by recently published techniques that use static and/or dynamic analysis (Halfond & Orso, 2005; Buehrer, Weide, & Sivilotti, 2005; Su & Wassermann, 2006; Bandhakavi, Bisht, Madhusudan, & Venkatakrishnan, 2007), dynamic taint analysis (Nguyen-Tuong, Guarnieri, Greene, Shirley, & Evans, 2005; Pietraszek & Berghe, 2005), or machine learning methods (Valeur, Mutz, & Vigna, 2005). However, the requirements for analysis and/or instrumentation of the application source code (Halfond & Orso, 2005; Buehrer et al., 2005; Su & Wassermann, 2006; Bandhakavi et al., 2007), runtime environment modification (Nguyen-Tuong et al., 2005; Pietraszek & Berghe, 2005), or acquisition of training data (Valeur et al., 2005) limit the adoption of these techniques in some real-world settings. Moreover, a common deficiency of existing SQLIA approaches based on analyzing dynamic SQL statements is in defining SQLIAs too restrictively, which leads to a higher than necessary percentage of false positives (FPs). False positives could have significant negative impact on the utility of detection and protection mechanisms, because investigating them takes time and resources (Julisch & Darcier, 2002; Werlinger, Hawkey, Muldner, Jaferian, & Beznosov, 2008). Even worse, if the rate of FPs is high, security practitioners might become conditioned to ignore them.

In this paper, we propose an approach for retrofitting existing Web applications with runtime protection against known as well as unseen SQL injection attacks (SQLIAs) without the involvement of application developers. Our work is mainly driven by the practical requirement of Web-application owners that a protection mechanism should be similar to a software-based security appliance that can be "dropped" into an application server at any time, with low administration and operating costs. This "drop-and-use" property is vital to the protection of Web applications where source code, qualified developers, or security development processes might not be available or practical.

To detect SQLIAs, our approach combines two heuristics. The first heuristic (labeled as "token type conformity") triggers an alarm if the parameter content of the corresponding HTTP request is used in non-literal tokens (e.g., identifiers or operators) of the SQL statement. While efficient, this heuristic leaves room for false positives when the application developer (intentionally or accidentally) includes tainted SQL keywords or operators in a dynamic SQL statement. This case would trigger an SQLIA alarm, even though the query does not result in an SQLIA. For instance,