Methods of Identifying and Preventing SQL Attacks

Bojken Shehu¹, Aleksander Xhuvani² and Shqiponja Ahmetaj³

¹ Computer Engineering Department, Faculty of Information Technology University, Polytechnic University of Tirana
Tirana, Albania

² Computer Engineering Department, Faculty of Information Technology University, Polytechnic University of Tirana
Tirana, Albania

³ Computer Department, Faculty of Informatics, Vienna University of Technology Vienna, Austria

Abstract

The paper begins by identifying the organizations which are vulnerable to the SQL attack referred to as an SQL injection attack (SQLIA). The term “SQL injection attack” is defined and a diagram (Fig.1) is used to illustrate the way that attack occurs. In another section, the paper identifies the methods used to detect an attack to SQL, whereby the techniques are discussed extensively using relevant diagrams for illustration. The other sections cover the preventive methods, where the methods are also discussed with an illustration using diagrams.

Keywords: SQLIA, WebSSARI, WAVE, AMNESIA, SQL DOM, tautology.

1. Introduction

There are numerous web applications used by various companies and organizations in order to provide services to users, such as online banking and shopping, hence establishing a need to develop a database. These web applications contain confidential information like the customer’s financial records, thus making these applications frequent targets for attackers. The attack to the SQL is referred to as the SQL injection, which gives attackers unauthorized access to the databases of underlying Web applications (Huang, Yu, Hang and Tsai, 148). Therefore, these attackers are able to leak, modify and delete information, which is stored on these databases, thus resulting to problems for the organization. In this case, the paper will discuss issues related detection and prevention of SQL attacks.
intrusion from the Internet before and after penetrating the firewall.

The other way of detecting the SQL injection attacks is through the Taint Based Approach, which uses the WebSSARI for the detection of input-validation concerning the errors through an analysis of the information flow. Moreover, this approach uses static analysis in checking the taint flows against preconditions for the sensitive functions. In fact, this analysis detects the points that have failed to meet preconditions, hence suggesting the filters and sanitization function, which is added to the application in order to satisfy the preconditions. The WebSSARI system functions through consideration of sanitizing the input, which as passed through predefined set of filters. In this way, the system can detect vulnerabilities in the application, though there are drawbacks associated with assumptions of adequacy in preconditions for sensitive functions that are accurately expressed by typing system.

The other method is the Black Box Testing, which is used for testing the vulnerabilities of the Web applications for the SQL injection attacks, through a technique that applies the Web crawler too identify the points that can be used by an attacker. The method also builds attack-targeting points that are based on a list of pattern attack techniques, while WAVE monitors the response of application to the attacks, by use of machine learning techniques, in order to improve the methodology of attacks. Moreover, this attack improves over the penetration testing through approaches of machine learning in order to guide the testing, though its limitation is that testing techniques cannot provide a guarantee of completeness.
the clear box there are constraints and existing data, which are used to refer to access codes to the database.

The other method is the Static Code Checkers, which uses techniques of statistically checking correction of SQL queries that are dynamically generated (Gould, Su and Devanbu, 654). This approach was developed for detecting the attacks that exploit the mismatches that occur in the dynamically generate query string (Haldar, Chandra and Franz, 303). The Checker detects the cause of the SQL injection attack vulnerabilities through a code improper form of checking input. Nevertheless, the system lacks the ability to detect general types of SQLIAs, since most of the attacks comprise of syntactically and queries that are correct. In addition, this approach uses static analysis, which is integrated with automated reasoning for verification of SQL queries that are generated by an application layer that entails a tautology, though the approach is limited to detecting tautologies and not other forms of attacks.

```java
public class SomeOtherClass
{
    int _someField;

    public SomeOtherClass(int someField)
    {
        _someField = someField;
    }
}
```

Fig 5. Example of a code detected.

The diagram (Fig.5) shows an example of a code detected through techniques of statistically checking correction, whereby the code is underlined.

3. Methods of Preventing SQL Attacks.

There are methods used in order to prevent SQL attacks, and one of them is the use of Proxy Filters, which is a system of enforcing input validation rules on data that are flowing the to a web application. The developers offer constraints through the Security Policy Descriptor Language (SPDL), thus specifying the transformations that are applied for application of parameters that flowing Web page to the application server (Boyd and Keromytis, 292). This method also allows developers to express their policies since SPDL is highly expressive, though the approach is human-based and defensive programming, thus requiring the developers to identify the data that require filtering.

The other preventive method relates to the use of Combined Static and Dynamic Analysis, through a model referred to as AMNESIA, which is a technique integrating static analysis and monitoring runtime. AMNESIA applies statistical analysis that develops models of different forms of queries that are generated by an application at a point of access to the database (Halfond and Orso, 174). In fact, this model intercepts queries sent to the database, and checks query against the model that is built statically, thus providing a basis for identifying the queries that violate the model, hence preventing them from executing on the database, though this model has a constraint associated with dependence on the precision of the static analysis for developing the models.

The other preventive method is the New Query Development Paradigms, which entails two resent approaches: SQL DOM and Safe Query Objects, and application of encapsulation of database queries that offer a safe and reliable way of accessing the database. This method provide an effective way of avoiding SQL attacks, by altering the process of building the query from an unregulated process that utilizes strings concatenation to a process involving a type check of API (McClure and Krüger, 88). Therefore, this method allows a systematic application of best coding practice like filtering of input and checking of user input, thus altering the development of paradigm that create SQL queries can eliminate the coding practice that facilities vulnerabilities SQLIAs. Nevertheless, this method has a drawback associated with the requirement of a developer to learn and apply new programming paradigm or query development process. Consequently, focusing on the use of a new development process, there is no provision of any form of protection for a legacy system.
4. Conclusions

In conclusion, the paper has explored issues related to the detection and prevention of SQL injection attacks, whereby there are several methods that are identified and discussed that are aimed at detecting or preventing the attacks. Most of the methods discussed are commonly used by organizations such as commercial and the government institutions, which are more subjected to the risk of SQL attacks; hence, the paper has met the objective set by the thesis statement at the beginning of the paper.

References


