Methods of Identifying and Preventing SQL Attacks

Bojken Shehu[1], Aleksander Xhuvani[2] and Shqiponja Ahmetaj[3]

[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.

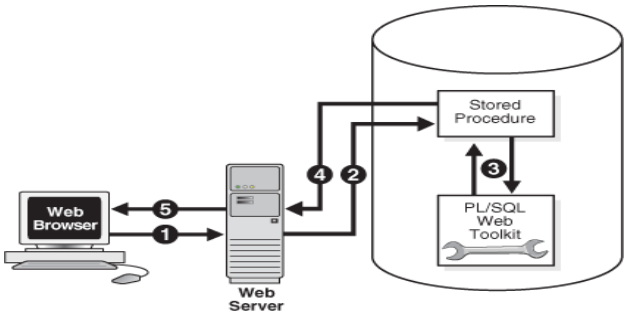

Fig 1. The way that attack occurs.

Commercial and governmental institutions are the common victims of SQL Injection Attacks (SQLIAs), due to the insufficiency in the input validation. In fact, these cases occur when Web application receives a user input, thus using it for building a database query without ample validation, hence creating a chance for an attacker to utilize the vulnerability. The vulnerability of the databases to SQL injections has been regarded as the most serious threats for Web application (Wassermann and Su, 78). This creates a form of vulnerability to SQL injection, thus allowing the attacker to have accessibility to the underlying databases, and it results to security violations since the information in these databases is sensitive. The implications of SQL injections are issues like loss of credentials, theft and fraud, and in other cases the attackers are able to use the vulnerability to acquire control and corrupt the system hosting the Web application.

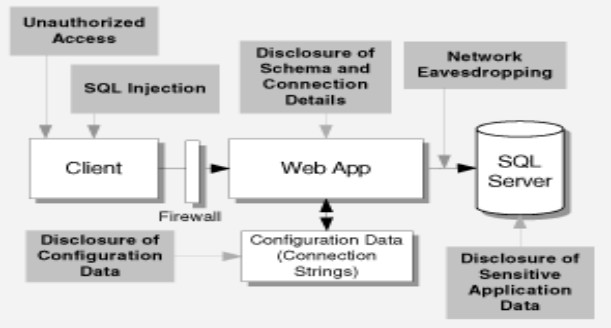

Fig 2. SQL injection.

The diagram (Fig.2) illustrates an SQL injection, whereby an attacker uses a client through the firewall into the web application where access to SQL server is achieved and sensitive application data is disclosed.

2. Methods of Identifying SQL Attacks

Numerous methods can be applied in detecting SQL injection attacks, and one of them is the Intrusion Detection System (IDS), which is based on a machine learning technique and application of a set of distinctive application queries. Moreover, this technique relates to a model of distinctive queries and a function of monitoring application at runtime in order to identify the queries that are not matching the model (Pietraszek and Vanden, 2). Therefore, this makes the system be have the ability of detecting attacks effectively, though there are basic demerits of learning based techniques, since does not offer guarantee concerning the detection abilities. In fact, the detection abilities are dependent on the quality of the training set applied; thus, a poor training set can result to generation large numbers of false positive and negative by the learning technique (Valeur, Mutz, and Vigna, 40).

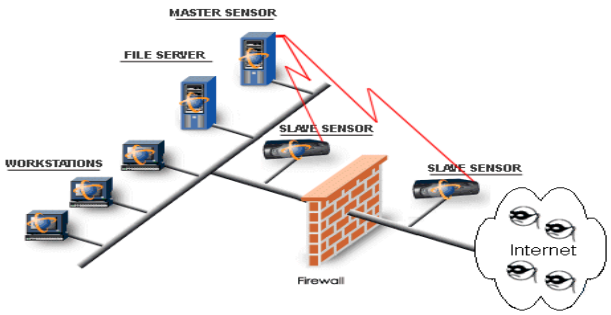

Fig 3. Intrusion Detection System (IDS).

The diagram (Fig.3) shows the locations of the Intrusion Detection System (IDS), whereby there are two sensors located at both sides of the firewall in order to detect any 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.

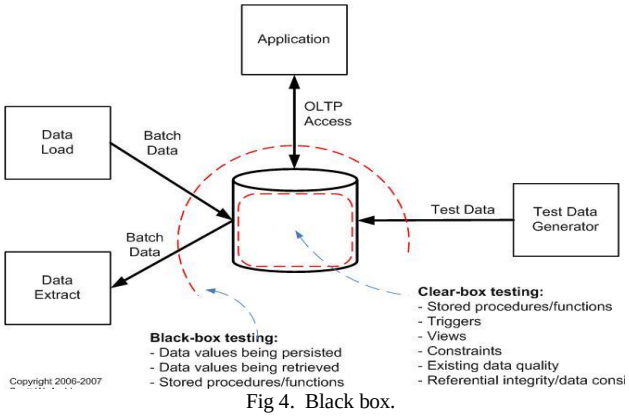

Fig 4. Black box.

The diagram (Fig.4) indicates the location of the black box, which is at the centre of the network, whereby data values are persistently retrieved and stored. Moreover, in

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.

```
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.

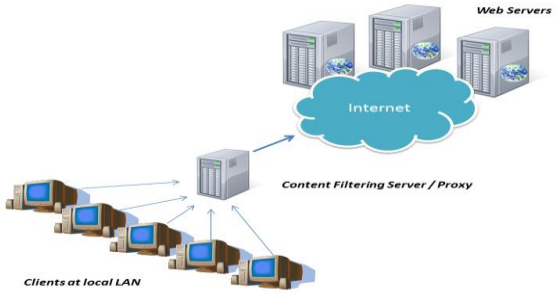

Fig 6. Depicts the filtering which occurs between the server and the Internet

The diagram (Fig.6) depicts the filtering which occurs between the server and the Internet in order to prevent an injection by an attacker using the clients to the web servers with SQL database.

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 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

[1] Boyd, Stephen, and Keromytis, Angelos. "SQLrand: Preventing SQL injection attacks". *In Proc. of the 2nd Applied Cryptography and Network Security. Conf. (ACNS 2004)*, pages 292–302, Jun. 2004.

[2] Gould, Carl, Su, Zhendong and Devanbu Premkumar. "Static Checking of Dynamically Generated Queries in Database Applications". *In Proc. of the 26th Intern. Conf. on Software Engineering (ICSE 2004)*, pages 645–654, May 2004..

[3] Haldar, Vivek, Chandra, Deepak and Franz, Michael. "Dynamic taint propagation for java". *In Proc. of the 21st Annual Computer Security Applications. Conf. (ACSAC 2005)*, pages 303–311, Dec. 2005.

[4] Halfond, William and Orso Alessandro. "AMNESIA: Analysis and Monitoring for Neutralizing SQL-Injection Attacks". *In Proc. of the IEEE and ACM Intern. Conf. on Automated Software Engineering (ASE 2005)*, pages 174–183, Nov. 2005.

[5] Huang Yao-Wen, Yu Fang, Hang Christian and Tsai Chung-Hung. "Web Application Security Assessment by Fault Injection and Behaviour Monitoring". *In Proc. of the 12th Intern. World Wide Web Conf. (WWW 2003)*, pages 148–159, May 2003.

[6] McClure Russell, and Krüger, Ingolf. "SQL DOM: Compile Time Checking of Dynamic SQL Statements". *In Proc. of the 27th Intern. Conf. on Software Engineering (ICSE 2005)*, pages 88–96, May 2005.

[7] Pietraszek Tadeusz and Vanden Chris. "Defending Against Injection Attacks through Context-Sensitive String Evaluation". *In Proc. of Recent Advances in Intrusion Detection. (RAID 2005)*, Sep. 2005.

[8] Valeur Fredrik, Mutz Darren, and Vigna Giovanni. "A Learning-Based Approach to the Detection of SQL Attacks". *In Proc. of the Conf. on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA 2005)*, Jul. 2005.

[9] Wassermann Gary and Su Zhendong. "An Analysis Framework for Security in Web Applications". *In Proc. of the FSE Workshop on Specification and Verification of Component-Based Systems (SAVCBS 2004)*, pages 70–78, Oct. 2004.

Bojken Shehu. He is a pedagogue in Polytechnic University of Tirana, Faculty of Information Technology, in Computer Engineering Department. In 2010 he has finished the Master Thesis in Bauman Moscow State Technical University, Russia and now he is a PhD student in Polytechnic University of Tirana. His PhD topic according to database security direction.

Aleksander Xhuvani. He is a chief of Computer Software Department in Polytechnic University. He has finished the PhD study at Bordeaux in France. At 2004 he is graduated as Prof.Dr.

Shqiponja Ahmetaj. She is a Master of Science student in Vienna University of Technology, Faculty of Informatics. In 2011 she has finished the Bachelor Thesis in Saint Petersburg State Polytechnical University, Russia.