Chapter 10

SQL Injection Attacks

Countermeasures

Kasra Amirtahmasebi
Chalmers University of Technology, Sweden

Seyed Reza Jalalinia
Chalmers University of Technology, Sweden

ABSTRACT

Due to the huge growth in the need for using Web applications worldwide, there have been huge efforts from programmers to develop and implement new Web applications to be used by companies. Since a number of these applications lack proper security considerations, malicious users will be able to gain unauthorized access to confidential information of organizations. A concept called SQL Injection Attack (SQLIA) is a prevalent method used by attackers to extract the confidential information from organizations’ databases. They work by injecting malicious SQL codes through the web application, and they cause unexpected behavior from the database. There are a number of SQL Injection detection/prevention techniques that must be used in order to prevent unauthorized access to databases.

INTRODUCTION

Web applications basically work by getting some input/information from outside users. Using string operations, this information is later used and serialized into a textual representation and ultimately, a database-specific command is created and sent to the database for execution. The input from outside users is the basis of SQL injection attacks and can be malicious; therefore they must be monitored and controlled thoroughly. The 2002 Computer Security Institute and FBI revealed that on a yearly basis, over half of all database experience at least one security breach and an average episode results in close to $4 million in losses (C.S. Institute. Computer crime and security survey. http://www.gocsi.com/press/20020407.jhtml, 2002). As a result, input-validation vulnerabilities which are incorrect assumptions of input data must be carefully considered and monitored in order to reduce the risk of injection attacks.

Pietraszek and Berghe (2005) classify input validation vulnerabilities into the following categories.
SQL Injection Attacks Countermeasures

- Buffer overflow vulnerabilities are caused by incorrect assumptions on the input’s maximum size. As a result, buffer’s boundary will be overrun and an attacker will be able write to the adjacent memory space.
- Integer overflow vulnerabilities are a result of insufficient assumptions on the range of the input. In this case, the result of an arithmetic operation will be larger than the memory space provided for the aforementioned variable.
- Injection vulnerabilities result from invalid assumptions on the type of input provided by the users. A user can enter syntactic content in his/her input and cause the database to perform unauthorized operations.

We have presented six SQL injection prevention techniques in this chapter which will cover a wide range of SQL injection attacks. A combination of these prevention techniques may lead to a more secure and reliable database system.

BACKGROUND

Many web pages ask users to input some data and make a SQL queries to the database based on the information received from the user i.e. username and passwords. By sending crafted input a malicious user can change the SQL statement structure and execute arbitrary SQL commands on the vulnerable system. Consider the following username and password example, in order to login to the web site, the user inputs his username and password, by clicking on the submit button the following SQL query is generated:

```
SELECT * FROM user_table WHERE user_id = 'john' and password = '1234'
```

Now consider what will happen if the user input the following password:

```
' or 1=1 --
```

The SQL query will become:

```
SELECT * FROM user_table WHERE user_id = 'john' and password = '' or 1=1 --'
```

The “or 1=1” will result in returning all the records in the “user_table” and the “--” comments out the last ‘ appended by the system. Therefore, the query will return a non-empty result set without any error. SQL injection problem can be solved by checking all SQL statements before sending them to the database; however, with respect to dynamic generation of SQL queries by web applications, each statement might be different, so we are not able to predefine the allowable SQL statements (Sam, 2005).

A LEARNING-BASED APPROACH TO DETECT INJECTION ATTACKS

Intrusion Detection Systems (IDS) are a common way of protection against malicious behavior and unauthorized access in regular desktop IT systems. An ID plays an important part in detecting and preventing attacks in regular desktop IT systems. Extensive research has been conducted on developing efficient and effective Intrusion Detection Systems. They can become handy when there are no prevention techniques defined for special types of attacks. In this case, intrusion detection systems can alert administrators about the attack so that appropriate measures can be taken.

In typical desktop IT, Intrusion Detection System is categorized as network based or host based IDS. Host based IDS monitors the activities on end systems such as system threads and processes. Network based IDS, on the other hand, monitors and analyzes traffic on the whole network by e.g. packet inspection to find attacks and malicious behavior (Hoppe, 2009).