ISSN No. 0976-5697

Volume 2, No. 1, Jan-Feb 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Classification of SQL Injection Attacks and using Encryption as a Countermeasure.

Kuldeep Rana
B.tech (purs.)
Bhagwan Parshuram Instt. of Technology (GGSIPU)
ranakuldeep1989@gmail.com

Abstract: This paper is devoted to evaluation of the vulnerabilities of web application to the various classes of SQL injection attacks and the countermeasures that can be used to put a check on them. The reason for the vulnerability is the use of limited set of attack pattern by the web developers during evaluation. This paper also presents an improvement over the existing countermeasure techniques by proposing a new defence mechanism that includes encryption and decryption of data entered in the login parameters.

Keywords: SQL Injection, Countermeasure, Encryption, Web Applications, Security

I. INTRODUCTION

SQL is short for **S**tructured **Q**uery **L**anguage and is a widely used database language, providing means of data manipulation (store, retrieve, update, delete) and database creation. Almost all modern Relational Database Management Systems like MS SQL Server, Microsoft Access, MSDE, Oracle, DB2, Sybase, MySQL, Postgres and Informix use SQL as standard database language. Queries are the primary mechanism for retrieving information from a database and consist of questions presented to the database in a predefined format. In SQL, these queries are of two types- Data Definition Language Statements (DDL) and Data Manipulation Language statements (DML). DDL statements can modify the structure of databases while DML statements can manipulate the contents of databases.

SQL injection is a code-injection attacks in which input data is included in dynamically constructed SQL query and treated as SQL code. For database-reliant web sites, SQL injection vulnerabilities are frequently exploited by attackers since they are easy to find and penetrate. A study conducted by the Gartner Group found that on over 300 tested web sites, 97% were vulnerable to SQL injection attacks. By detecting the SQL injection vulnerabilities, attackers commonly extract and modify data by constructing DDL and DML queries. [2]

This paper presents various defense mechanisms that can be used as countermeasures to the SQL injection attacks. It proposes a new approach to prevent attacks made to the system i.e. using encryption and decryption techniques.

II. TYPES OF SQL INJECTION ATTACKS

The impact of SQL injection attacks may vary from gathering of sensitive data to manipulating database information, and from executing system-level commands to denial of service of the application. The impact also depends on the database on the target machine and the roles and privileges the SQL statement runs with. There are four main categories of SQL Injection attacks against databases:-

A. *SQL Manipulation*

Manipulation is process of modifying the SQL statements by changing the where clause of the SQL statement to get different results. This type of attack is the simplest form of attack and can be used easily by even the naïve hackers. Another way of implementing SQL Injection using SQL Manipulation method is by using various set operators such as UNION, INTERSECT.

B. *Code Injection*

In general code injection is the exploitation of a computer bug that is caused by processing invalid data. It can be used by an attacker to introduce (or "inject") code into a computer program to change the course of execution. The results of a code injection attack can be disastrous. SQL code injection is process of inserting new SQL statements or database commands into the vulnerable SQL statement. One of the code injection attacks is to append a SQL Server EXECUTE command to the vulnerable SQL statement. This type of attack is only possible when multiple SQL statements per database request are supported.

C. *Function Call Injection*

Function call injection is process of inserting various database function calls into a vulnerable SQL statement. These function calls could be making operating system calls or manipulate data in the database.

D. *Buffer Overflows*

Buffer overflow is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory. Buffer overflows can be triggered by inputs that are designed to execute code, or alter the way the program operates. This may result in erratic program behavior, including memory access errors, incorrect results, a crash, or a breach of system security. They are thus the basis of many software vulnerabilities and can be maliciously exploited. Attackers use buffer overflow to corrupt the execution stack of a web application. It can be caused by using function call

injection. For most of the commercial and open source databases, patches are available. This type of attack is possible when the server is un-patched. [9]

III. SOME COMMON SQL INJECTION TECHNIQUES USED BY ATTACKERS

A. Tautology

This technique relies on injecting statements that are always true so that queries always return results upon evaluation of a WHERE conditional. A common example would to be inject an "or1=1" into the login parameter.

B. End of Line Comment

After injecting code into a particular field, legitimate code that follows are nullified through usage of end of line comments. An example would be to add "--" after inputs so that remaining queries are not treated as executable code, but comments.

C. Illegal/Logically Incorrect Query

This technique is usually used by the attacker during the information gathering stage of the attack. Through injecting illegal/logically incorrect requests, an attacker may gain knowledge that aids the attack, such as finding out the injectable parameters, data types of columns within the tables, names of tables, etc.

D. UnionQuery

Threat attackers use this technique to guide servers to return data that were not intended to be returned by the developers. A common example would be to add the statement "UNION SELECT ",along with an additional target dataset so that queries return the union of the intended dataset with the target dataset.

E. Piggy-backed Query

The attackers add additional queries beyond the intended query, effectively "piggy-backing" the attack on top of a legitimate request. This technique relies on server configurations that allow several different queries within single string of code. For example, the threat agent may add a query delimiter such as ";", and then follow up with a command of his/her own, such as "droptable <name>", which effectively deletes the table specified.

F. System Stored Procedure

Database server often ship with system stored procedures that programmers may use when developing application. If the threat agent has knowledge of which back-end server is running, he/she may be able to exploit these stored procedures to perpetrate their attacks. Stored procedures may yield results that go beyond the database itself, but also interact with the OS, for example.

G. Blind Injection

With sufficiently secure systems, attackers may probe for vulnerable parameters or extract data by using this technique. Blind injection allows threat agents to infer the construct of the database through evaluating expressions that are coupled with statements that always evaluate to true and statements that always evaluate to false. For example, the threat agent can add "and1=0–"for one attempt, while "and1=1–" is used for another attempt, both added on to the same query. Through examining the behavior of the server, the threat agent may then deduce whether the particular parameter is vulnerable or not, where the two attempts result in the same behavior, the parameter is secure, while different behavior resulting from the two statements suggest that the parameter is vulnerable. [2]

IV. EXISTING COMMON APPROACHES OF COUNTERMEASURE TO SQL INJECTION ATTACKS AND PROBLEMS ASSOCIATED WITH THEM.

In the broad sense the most common defense mechanism approaches against SQL injection attacks include-

A. Customizing the error messages

This counter measure is particularly used against inference attacks in which no actual data is transferred directly but by observing the differences in the responses from the application the attacker can infer the value of data. If the error messages are customized the attacker may not be able to distinguish between two logically different responses.

Problem-Customizing or removing the error messages make debugging a difficult task.

B. Data validation technique

This technique involves accepting only valid input and rejecting the one known to be harmful e.g. SQL keywords are not allowed in the login parameters.

Problem-There are various techniques that can be used to avoid detection by data validation mechanisms such as use of hexadecimal encoding, in which the SQL keywords are represented in hexadecimal numbers.

C. Checking input data length

By checking for input variable length, malicious code strings beyond certain length limits will not be applicable. Even if the length limitation is long enough to fit a few additional queries, the inability to input an infinitely long string disables the threat agent from employing evasion techniques such as encoding.

Problem- With the advent of new hacking techniques the passwords and usernames are always advised to be sufficiently long. Putting restriction on the length of the input makes application vulnerable to hackers.

V. USING ENCRYPTION AS A COUNTERMEASURE

Encrypting the data internally before using it in the dynamic SQL queries can be used as a handy tool to prevent

applications from SQL injection attacks. In this technique the data entered in the login parameters will be first encrypted using any of the standard encryption techniques and then the encrypted values will be used in the SQL queries. For example-

If the attacker uses any SQL injection attack technique like tautology and enters xyz' or 'x'='x in the password field. The query formed will be-

SELECT password FROM login_table WHERE password = 'xyz' or 'x'='x';

Because the application is not really thinking about the query, merely constructing a string and use of quotes has turned a single-component **WHERE** clause into a two component and the **'x'='x'** clause is **guaranteed to be true** no matter what the first clause is. But before execution of the dynamic query (involving input data), if the input data is encrypted into a new code e.g. poiuytrewqasdf by using any standard encryption scheme, the new query formed will be-

SELECT password FROM login table WHERE password = 'poiuytrewqasdf';

This means most of the SQL injection attack techniques like tautology, union query, piggy backed query, and end of line comment etc will not work. But for this countermeasure to work properly, the database needs to be maintained in such a way that it contains only encrypted values. Having encrypted values in database provides additional security to the system. Also all the existing databases need to be transformed into new ones with encrypted values and the new application should be developed in such a way that before storing user details in the database, the values are first encrypted. Whenever these values are used in the application (or shown to user), they are first decrypted and then displayed. In this way the application can be made secure against most of the attacks. But frequent encryption and decryption affects the performance of the system. So, a trade off needs to be made between application performance and security.

VI. CONCLUSION

`SQL injection attack vulnerability is security hole that is found in most of the web applications. By taking advantages of this vulnerability, the attackers can get insight into the data present in the database server. There are various counter measures available against the numerous classes of attacks. But each counter measure has a problem associated with it (usually performance degradation). This paper presents the use of encryption and decryption as a tool for securing the web application against the SQL injection attacks. This scheme works against most of the attacks but encrypting and decrypting each time the user want to access the database degrades the performance of the application.

VII. REFERENCES

[1] J. V. William G. J. Halfondand A. Orso, "A classification of SQL injection attacks and countermeasures," 2006.

[2] "Classification of SQL Injection Attacks" San-Tsai Sun, Ting Han Wei, Stephen Liu, Sheung Lau Electrical and Computer Engineering, University of British Columbia, 2007.

[3] C.Anley, "Advanced SQL injection in SQL server application," Technical report, NGS Software Insight Security Research (NISR), 2002.[Online]. Available: http://www.nextgenss.com/papers/advanced SQL injection.pdf

[4] Cerrudo, "Manipulating Microsoft SQL server using SQL injection," Technical report, Application Security, Inc., 2003. [Online]. Available: http://www.appsecinc.com/presentations/Manipulating SQL Server Using SQL Injection.pdf

[5] C.Anley, "(more) advanced SQL injection in SQL Server application," Technical report, NGS Software InSight Security Research (NISR), 2002.[Online]. Available: http://www.nextgenss.com/papers/more advanced SQL injection.pdf

[6] K.Spett, "SQL injection: Are your web applications vulnerable?" Technical report, SPI Dynamics, Inc., 2005.[Online]. Available: http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf

[7] W.G. Halfondand A.Orso, "Amnesia:Analysis and monitoring for Neutralizing SQL injection attacks, "in 20th IEEE /ACM International Conference on Automated Software Engineering., Long Beach, California, USA, 2005.

[8] "A new taxonomy of web attacks suitable for efficient encoding" Computers & Security. E. Amoroso, Fundamentals of Computer Security Technology. Prentice-Hall PTR, 1994.

[9] Stephen Kost, "An Introduction to SQL Injection Attacks for Oracle Developers" Integrigy, January 2004.

[10] A. S. Ofer Maor, "SQL injection signatures evasion," White Paper, Imperva Inc., 2005. [Online].Available: http://www.imperva.com/application defense center/white papers/SQL injection signatures evasion.html.