IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 1, Feb-Mar, 2014
ISSN: 2320 - 8791
www.ijreat.org

Defeating SQL Injection through Efficient Scanner Tool

Princella Amirtha D[1], Dr.E.R.Naganathan[1]

[1]*PG student, M Tech CSE, Hindustan University, Chennai*

[2]*HOD, CSE Department, Hindustan University, Chennai*

Abstract

The main aim of this project is to develop an efficient scanner tool for web security against defensive coding practices with vulnerability detection and runtime attack prevention method. SQL injection is one of the most common application layer attack techniques used today. The SQL injection attack takes the advantages of improper coding of your web application that allows hacker to inject SQL character or keywords or commands into say a login form to allow them to gain access the data held within your database. So the hacker gets the original data from database and corrupts it. So the scanner tool is used to check the code and aids in code recovery and guarantees a secure website.

Keywords: Efficient Scanner Tool, Website Security, Defeating SQL Injection, Recovery Tool for SQL Injection.

1. Introduction

Structured query language injection is one form of attack through queries. SQL injection attack is a one of the most popular web application hacking method, in which the attacker or hacker steals the data from the organization. In other words through SQL injection attack an unauthorized person can access the database of the website. So the attacker can easily extract the data from the database. SQL injection is a standard for database manipulation. The main use of SQL is used to communicate with database. Through queries injection we can easily modify the data in the database which can lead to the leak of confidential information such as credit card numbers, passwords, commercial information and table structure.

The Web programming language such as Java, ASP.NET and PHP provide various steps for Constructing and executing SQL statements but due to lack of training the developers misuse these methods causing SQL vulnerabilities. The developers commonly rely on dynamic query building with string concatenation to construct SQL statements. During the execution the system forms queries with inputs directly received from external sources. This method makes it possible to build different queries based on varying conditions set by users. This may cause many SQLIVs and using parameterized queries or stored procedures is a more secure method but inappropriate use can still result in vulnerable code.

2. Related Work

Learning from the previous works as described in [3] Boyd, keromytis-2004 checks the SQL injection attack developed by SQL rand, which uses a proxy to append key to SQL keyword. It uses set randomization of SQL statement to find whether it is affected by SQL injection attack by using de-randomizing proxy. It converts the randomized query to proper SQL queries for the database. The attacker does not know the key so the code harmed by the attacker is treated as expressions and undefined keywords. This causes runtime exception and the database does not receive any query. The main disadvantage of this system is security of the key and its complex configuration. If the key is not secure the attacker can easily formulate the queries and enter into the database.

Ke Weietal-2006[4] proposed a novel method to defend against the attacks targeted at stored procedures. The static application code analysis with runtime validation is combined in this method to eliminate the happenings of such attacks. For any SQL statement the stored procedure parser are designed in the static part which is based on the user inputs. The parser is used to compare the original SQL statement structure including the user inputs. The result of the method is proposed in [1] based on several stored procedures in SQL Server 2005 database. The owner of the website will not see the security vulnerabilities in their websites written in ASP.NET; therefore the suggested algorithm performs a scanning process for all websites/application files.

The scanner tool proposed here checks the source code of the application on the Asp.net files. The scanner tool describes most leaks and vulnerabilities types. It provides suggestion but not the recovery of code where the malicious SQL statements are inserted into an entry field execution. Example: to dump the data base content to the attacker. The suggested scanner tool will help organizations to fix the vulnerabilities and improve the overall security.

3. Proposed System

The proposal here about developing the efficient scanner tool for website security, not only identifies the malicious statement and making recovery of that code through scanner tool which converts insecure code into secure code. It analyses and monitors a solution against SQL injection where attackers use all types of SQLIA defenses. There is a no way can an attacker can modify SQL statements. No false positives and it is used a set of real web application.

4. System Architecture

The system architecture is classified into four parts: defensive coding, SQLIV detection, prepared statement replacement algorithm, model based testing and SQLIA runtime prevention.

4.1. Defensive coding:

It is a straight forward solution, the developers easily consequence the SQLIVs for insecure coding practices.

Incorrect filtered escape characters:

This form of SQL injection occurs when user input is not filtered for escape character and is then passed into a SQL statement. This result in the potential manipulation of the statement performed on the database by the end-user of the application.

The following line of code illustrates this vulnerability:

Statement="SELECT*FROM users WHERE name='"+username+"';"

This query is designed to pull up the records of the specified username from its table of users. If the "username" variable is crafted by a malicious use, the SQL statement may do more than the code author intended. For example, setting the "username" variable as:

'or'1'=1'

In other case by using comments to even block the rest of the query. There are three types of SQL comments, they are a follows

'or'1'='1'--'
'or'1'='1'({'
'or'1'='1'/*'

Here the SQL code is used in an authentication procedure then this example is used to select a valid username because the evaluation of '1'='1' is always true.

Incorrect type handling:

This form of SQL injection occurs when a user field is not strongly typed or is not checked for type constraints. this could take place a numeric field is to be used in a SQL statement but the programmer makes no check prove that the user supplied input is numeric. For example:

Statement:= "SELECT * FROM userinfo WHERE id="+a_variable+";"

It is clear form this statement that the user trying a_variable to be a number connection to the "id" field. However, if it is in fact a string then the end-user may disapprove the statement as they choose, there by bypassing the need for escape characters.

White list filtering:

The bad special characters such as ‘ and ; are used by the developers to reject by using the bad list filtering from the parameters to avoid SQL injection. However, accepting only inputs know to be acceptable is safer. Developers could keep a list of data patterns and accept only matching input data.

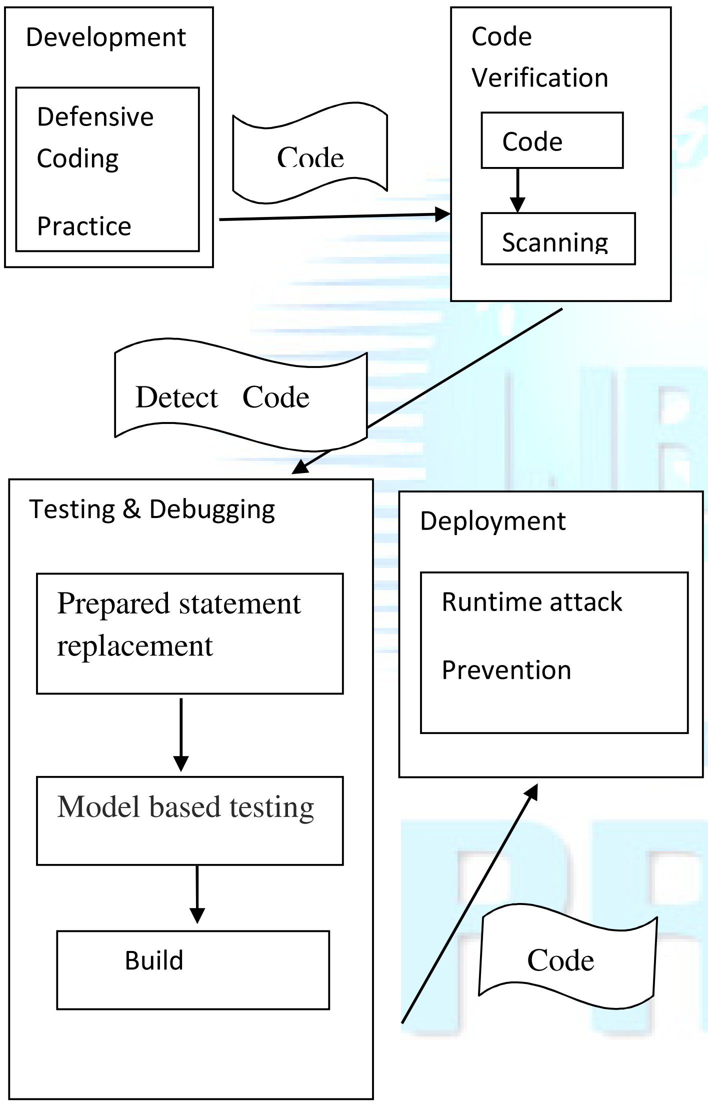

Fig: 1. Efficient Scanner Tool Architecture

4.2. SQLIV Detection:

You can configure code analysis to run for each build of a managed code project. You can set different code analysis properties for each visual studio configuration. The reported risk level is set automatically by the tool with no manual verification by the test vendor. This can be supplemented with training based scanning that looks to remove some common false positives by using supplied training to authenticate with a service.

4.3. Prepared statement replacement algorithm:

The PSR-algorithm is used to remove the SQLIVs that contains in the existing source code. The PSR-algorithm analyzes the source code containing SQLIVs and it generates a specific useful code structure containing prepared statements. The PSR algorithm separates the SQL statement’s input from the SQL structure. For each string object, it creates the SQL statement by the additional string object of the PSR algorithm. The new string object contains the raw string data of the original string object and any identifiers found in the original string object the PSR algorithm identifies as SQL structure. An assistant vector for each new string object is created by the PSR algorithm. The PSR algorithm generates string objects can contain other PSR algorithm generated string object based on how the original string are used. Therefore, assistant vector can contain other assistant vector, creating a tree. The tree contains proper variables for each decision path of the conditional.

4.4. Model based testing:

Model-based testing approaches the system is modeled by a set of logical expression specifying the system’s behavior.

Constraint logic programming and symbolic execution:

Constraint programming can be used to select test cases satisfying specific constraints by solving a set of constraints over a set of variables. The system is described by the means of constraints. Solving the set of constraints can be done by Boolean solvers. A solution found by solving the set of constraints formulas can be can serve as a test cases for the corresponding system. Constraints programming can be combined with symbolic execution. In this approach a system model is executed symbolically, that is collecting data constraints over different control path, and then using the constraint programming method for solving the constraints and producing test cases .Model checkers can also be used for test case generation. Originally model checking was developed as a technique to check if property of a specification is valid in a model.

4.5. SQLIA runtime prevention:

Input validating techniques can prevent some vulnerability. The static query statement is not good when we use of dynamic query, and finally the IDS Intrusion detection system on protection in firewalls, proxy and gateway.

5. Conclusion

SQL injection attack is web based application attack used by the hacker to access the secret data, modifying the contents of website, by passing the logins and shutting down the My SQL server. For solving the problem this proposes an efficient scanner tool which used to scan the source code affected by the SQLIA and make the code secure through which the attacker does not enter into the website.

References

[1] J.C. Lin, J.M. Chen, and C.H Liu: An Automatic Mechanism for Sanitizing Malicious Injection. The 9th International Conference for Young Computer scientists, IEEE Computer Society 2008.

[2] E. Bertino, A.Kamra, nd James P. Early: Profiling Database Application to Detect SQL Injecction Attacks, IEEE Conference 2007.

[3]S.W. Boyd and A.D. keromytis: SQL Rand: Preventing SQL Injection Attacks. In Proceedings of the 2nd Applied Cryptography and Network Security(ACNS) Conference, Spring-Verlag. 2004.

[4]Ke Wei, m. Muthuprasanna, Suraj Kothair: preventing SQL Injection Attacks in Stored Procedure, Australian Software Engineering Conference(ASWEC’06), IEEE Computer Society 2006.

[5]Lwin Khim Shar and Hee Beng Kuan Tan, Nanyuang Technogical University of Singapore: Defeating SQL injection, IEEE Computer Society 2013.