Detection And Prevention of SQL-Injection Attacks of Web Application Using Comparing Length of SQL Query

Kishori Lal Bansal[1] & Sunil Kumar[2]

Himachal Pradesh University, Department of Computer Science,
Himachal Pradesh University, Shimla-171005, India.
E-mail : [1]kishorilalbansalhpu@rediffmail.com & [2]sunilkumar301hpu@rediffmail.com

***Abstract* – Now a days we are using the web applications, but the application code is not secured so there is existence of the SQL injection attacks. In web applications with the help of the internet explorer the user tries to access the information. But most of the web applications are affected by the SQL-injection attacks.**

Our main aim of the research is to find the method which is able to detect and prevent our web applications from the SQL Injection attacks .In our approach firstly we check the length of the original SQL Query and store its length value. If there is another Query which is used for SQL-Injection attacks, then we too check the length of this SQL Query and store its value. If the length of both the queries is same then the second Query is not the SQL-Injection Query, else the other Query is SQL-injection Query. If there is SQL-Injection attack then we don't allow attacker to access the database by giving the access deny or by giving the error message.

Keywords: PII, SQL Injection, SQL Query, SQL Injection attacks.

I. INTRODUCTION

In recent years the internet technologies are going to advance and become popular. The internet is used for many purposes **l**ike Emai**l** Information**,** Business**,** Social Networking**,** Shopping**,** Entertainment**,** E-Commerce etc. Even people are going to shopping, banking and order any item over the Internet.

Personally Identifiable Information (PII), as used in information security, is information that can be used to uniquely identify, contact, or locate a single person or can be used with other sources to uniquely identify a single individual. The abbreviation PII is widely accepted, but the phrase it abbreviates has four common variants based on personal, personally, identifiable, and identifying. Not all are equivalent, and for legal purposes the effective definitions vary depending on the jurisdiction and the purposes for which the term is being used [1]. PII can be found in the investment

Accounts, credit and debit card accounts and retirement accounts also. But this PII is not safe due to SQL-I attack's to the databases.

Many web applications are using the three tier architecture. In presentation Tier with the help of the web browser users access the web server and the database server.

All the information residing on the database. Un-authorized users want to access this information with the help of SQL-I attacks. The Structural Query Language Injection (SQLI) attack occurs when an attacker changes the logic, semantics or syntax of a SQL query by inserting new SQL keywords or operator's .SQL injection vulnerability allows attackers to destroy user's information and data confidentiality. The SQL-I attacks have many methods, but the main aim of the hacker is to hack the username and password on the internet.

The SQL Injection attacks are the most dangerous attacks for databases we can also conclude it from the following surveys.

Acunetix ,a leading vendor of web application of the web application security solutions reported that 50% of the websites with instances of high vulnerabilities were susceptible to SQL injection[2].In the 2002 CSI and FBI revealed that on a yearly basis, over half of all databases experience at least one security breach and an average episode results in close to $4 million in lose[3].According to the web application security consortium 26.38% of the total attacks incident reported in the media until the end of the 2006 were due to the SQL injection. more recent data from this research shows that about 50% of the websites that were scanned are susceptible to SQL injection Vulnerabilities [4].Another study provided by the Gartner Group reveals that 75% of the attacks

occurs at the application layer [5].Secure Work an Internet security company reported that from January through march they will be able to bock 100 to 200 SQL injection attacks per day where as in April the number jumped from 1000 to almost 8000 per day[6].

Through a SQL Query attacker can add,modify or retrieve data in a database.SQL Injection enables attackers to access modify or delete critical information in a database without proper authorization,attackers can also execute arbitrary commands with hogh system privilege in the worst case[7].SQL Injection has recently been one of the top issues in software security[8].

Types of attacks:

Tautology attacks:

In Tautology-based attacks the main intention of the attacker is to make the conditional statements that are always evaluate to true. Attacker mostly uses the where clause of the query. Tautology attack is successful when the attacker is able to return all the records of the table or at least is able to return one of the records from the database.

e.g. SELECT accounts FROM users WHERE

Login=”or1=1--AND pass=” AND pin=

In this example the code injected in the conditional (or 1=1) will transfer the WHERE clause in to a tautology and the returned set evaluates to a value which will be not null, which results the application consider that the user authentication was successful.

Logically incorrect query attacks:

These types of attacks are mainly used for to know the structure of the database and the type of the backend databases. The returned error messages are helpful for attacker to know the structure and type of the database used.

e.g. SELECT accounts FROM users WHERE login=” AND

Pass=” AND pin=convert (int, (select top 1 name from

Sysobjects where xtype=’u’))

In this example firstly the query will try to extract the first user table that is xtype=’u’ .After that the query will try to convert the table name into an integer. The database will give an error due to not a legal type conversion. If we are using the Microsoft SQL Server then the error will be like “Microsoft OLE DB provider for SQL Server (0x80040E07) Error converting nvarchar value ‘CrditCards’ to column of data type int”. The attacker is able to know that the database used is a Microsoft SQL Server database and secondly the value of the string cause the Type conversion to occur.

Union Attack:

In Union Query the attacker uses the union operator. In this the attacker has the complete control of the second /injected query, attacker can use that query to retrieve information from any desired table in the database by making the guess of the table names. The result of the union attack is returned in the form of dataset which is result of the combination of the original query and the result of the second query that is union attack query.

e.g. SELECT accounts FROM users WHERE login=”UNION

SELECT cardno from creditcards where

AcctNO=100 -- AND pass=” AND pin=

In this example there is no login whose value is equal to” ”, the first query will return the null set of values, and the second query will return the data from the creditcards table. The database will return “cardno” for the account “100”.

Piggybacked Query:

In this attack the attacker tries to inject some extra types of queries in the original query, named as” piggy-back” This technique relies on the server configurations that allow the several different queries with a single string of code. The attacker uses the delimiter”;” for this attack, he adds some extra queries after the delimiter and these queries are run on the database.

e.g. SELECT accounts FROM users WHERE login=’abc’ AND

Pass=”; drop table users –‘AND pin=123

After completion of the first query the database would recognize the delimiter that is”;” and lateral will continue execution and will try to drop the table users, if the table exists then it can destroy the information of that particular table of database.

II. Our Approach

In our work we try to make a method in which we find the length of original SQL Query i.e. the Query which has to be executed on the Database server. We will store value of the length of this SQL Query and if there is arrival of the SQL Query from web application form then instead of sending it to the database we will compare the value of the total length of the coming SQL query. If the length of the SQL query is same than we consider that the coming SQL Query is not SQL Injected else it is SQL Injected and we not give access this Query to access the database. This is used for to detection of the SQL Injection attack and for to prevent from this attack we can deny the access to database from this query by giving the error message.

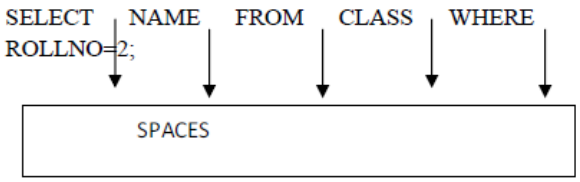
e.g.

If we are using database in which there is name ColumnName and class is table name and we want to access the information of the name of the student where rollno=2

Then the Query e.g. SQL Query1 will be like that:

SELECT NAME FROM CLASS WHERE ROLLNO=2;

We will calculate the total length of this Query1.

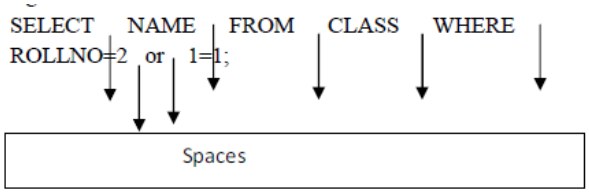


Dig no1.

Length of SELECT=6
Length of NAME=4
Length of FROM=4
Length of CLASS =5
Length of WHERE=5
Length of ROLLNO=6
Length of = OPERATOR=1
Length of 2 is 1
Length of; is 1
Length of 5 SPACES =5
Total length of the Query1 will be=6+4+4+5+5+6+1+1+1+5=38

If the attacker wants to change the intent of the SQL Query by entering some other keywords than the length of the SQL Query will be change.

e.g.



Dig no 2

In this SQL Query the attacker add some extra keywords like 1=1(Tautological Attack) this always evaluates to true and it will give some information, when we will compare the length of this SQL Query then it will become different to its original Query.

Length of SELECT=6

Length of NAME=4

Length of FROM=4

Length of CLASS =5

Length of WHERE=5

Length of ROLLNO=6

Length of = OPERATOR is 1

Length of 2=1

Length of or=2

Length of 1 is1

Length of = is 1

Length of 1 is 1

Length of; is 1

Length of 7 SPACES =7

Total length of the Query2 will be=6+4+4+5+5+6+1+1+2+1+1+1+1+7=45

Original SQL Query1 length=38

Injection attempt SQL Query2 length=45

(38! =45)

III. PROGRAM

The code of the program is:

```
<?php
$query1='SELECT NAME FROM CLASS WHERE ROLLNO=2';
$query2='SELECT NAME FROM CLASS WHERE ROLLNO=2 or 1=1';
echo ("Length of Query1 is :");
echo strlen ($query1);
echo("Length of Query2 is: ");
echo strlen ($query2);
if( strlen ($query1)!=strlen ($query2));
echo("This Query is SQL Injected Query.");
else
echo ("This Query is not SQL Injected Query.");
?>
```

IV.TABLE

Original Query(q1)	**SELECT NAME FROM CLASS WHERE ROLLNO=2;**
SQL Injected Query(q2)	SELECT NAME FROM CLASS WHERE ROLLNO=2 OR 1=1;
Length of q1	38
Length of q2	48
Result	Length of q1!=Length of q2 38!=48 This Query is SQL Injected Query.

V. OUTPUT

The output of the following program in the Internet Explorer will be like:
Length of Query1 is: 38 Length of Query2 is: 45
This Query is SQL Injected Query.

VI. CONCLUSION

By comparing the length of two Queries (Original Query and SQL Injection related Query) we can detect the SQL-Injection attack; if there is SQL Injection attack then by not giving the access to the database we can also prevent our web application from SQL-Injection attacks, the work presented in this paper has been implemented using PHP codes.

VII. REFERENCES

[1] PII Personal Identifiable related Information. http://en.wikipedia.org/ wiki/Personally_ identifiable_ information

[2] Web Application security Information. www.acunetix.com/ news/ security-audit-results.html

[3] Computer security institute, computer crime and security survey http://www.gocsi.com/ press/ 20020407.jhtml,2002

[4] Testing, security audits and other activities made by companies which were members of WASC www.webappsec.org/projects/statistics

[5] Analysis, Research, Event, Consulting company survey. www.gartner.com

[6] CVE information security vulnerabilities and exposures survey. http://cve.mitre.org

[7] C. Anley, Advanced SQL Injection in SQL Server Applications, White paper, Next Generation security software limited. http:/// www.ngssoftware.com/ papers/ advanced_sql_injection.pdf

[8] OWASPD-Open Web Application Security project. Top ten most critical Web Application Vulnerabilities.

◈◈◈