SQL INJECTION ATTACK DETECTION AND PREVENTION

PERUMALSAMY RAMASAMY†
MCA Department, Adhiyamaan College of Engineering, Hosur-635 109, Tamilnadu, India‡
ace09mca055@gmail.com§

Dr. SUNITHA ABBURU,
MCA Department, Adhiyamaan College of Engineering, Hosur-635 109, Tamilnadu, India
ausunithaa@yahoo.co.in.

Abstract:
Nowadays SQL injection attacks (SQLIAs) and cross scripting increased in real web applications very much, and the SQL injection attacks damages the databases through web applications. Injection queries are different type of way to attack the databases. This paper address the issue of SQLIA’s and script in an efficient way. The proposed approach detects and prevents all the Injection queries as well as the cross scripting through the Framework and Static analysis and Dynamic analysis. The contribution is twofold 1. SQL Injection Attack Detection and prevention 2. SQL Injection Reports. These techniques have been implemented in ASP.Net and SQL Server and tested by conducting various experiments and prove that the web applications and database is protected from scripting and SQL injection queries.

Keywords: Web framework; Static Analysis; Dynamic Analysis; SQL injection.

1. Introduction
Web application is a part of internet. Web application uses to retrieve and store the data in databases. Web application contains web forms, web server and backend. As the number of web application users are increasing the vulnerability of the web application are also increasing due to the attacks and scripting.

The web application attacker damages database information with help of SQL injection queries. SQLIA attack is damage the original and protected data in web applications. This is achieved through passing malicious codes and queries with normal user input through web applications. The proposed system is used to detect and prevent SQLIA and cross scripts with the use various techniques. Static analysis can used to detect and prevent SQL injection attacks in compile time and Dynamic analysis can used to detect and injection queries prevent in runtime with automatically.

2. SQL Injection Attack (SQLIA)
A SQL injection attack makes harm to the databases through web applications. SQL injection can access, modify and delete original and protect data from databases. Halfond and others [2006] have specified the SQL injection types and how the database is damaged.

Example:

');
declare @tbl_name as varchar(256)
declare @flag as int
declare @name as varchar(256)
set @flag=(select count(*) from sysobjects where type='U')
while(@flag>0)
begin
set @name=(select top 1 name from sysobjects where type='U')
set @tbl_name = 'drop table ' + @name
exec(@tbl_name)
set @flag=@flag-1
end;--

The above statement drops all tables link with current application.

3. Related Work

Here how to detects and prevents all the Injection queries as well as the cross scripting through the Framework and Static analysis and Dynamic analysis will be discussed and these techniques are very difficult to implement in web applications.

3.1. Web framework support

Web frameworks [Kosuga, Y (2007)] [http://www.php.net/] are used to prevent web applications from the special character [';/@] and meta Characters. But it does not prevent tautologies, illegal, union, piggyback, stored procedures attacks. So attackers can bypass this method.

3.2. Static analysis

Static analysis [Wassermann (2006)] proposed an approach that uses a static analysis can detect and prevent SQL injection attacks in compile time. This technique only focus on tautologies but it cannot detect illegal, union, piggyback, stored procedures SQL injections attacks.

3.3. Dynamic analysis

Dynamic analysis is different from Static analysis. However, Static Analysis to detect the queries in compile time. So the attacker passes advanced queries to web application and it is easily attacked. But Dynamic analysis [Paros] can used to detect and injection queries prevent in runtime with automatically. But this method do not addresses the issues of illegal, union, piggyback, stored procedures SQL injections attacks.

3.4. Combined static analysis and dynamic analysis

The advantage of combined static analysis and dynamic analysis method is to detect SQL injection attacks. It is more complex compared to other techniques. It can solve the problem which cannot be solved by other methods. AMNESIA [Halkunf, W.G (2005)] is a model-based technique that combines the static and dynamic analysis.

4. SQL Injection Detection Algorithm

This proposed approach detect all the Injection queries as well as the cross scripting through the Framework and Static analysis and Dynamic analysis and implement in sample applications. The aim of this research is two fold
1. SQL Injection Attack Detection and prevention.
2. SQL Injection Reports.

4.1. SQL injection attack detection and prevention

The proposed system detects and prevents the special characters and Injection of black spots in the web applications query. A list of possible special characters and words which are injected in to the SQL queries that may cause damage to the database are identified. And this list is considered to be as a bag of black spots that damages the database. The proposed method detects the black spots in the submitted SQL query and sends an SQL injection attack message to the system administrator which in turn generates and sends a warning message to the user and stops all transaction that is issued from that user and blocks that IP address of the user. This prevents the database damage.

4.1.1. Detection coding

private static string[] blackList = {"--",";--",";","/*","*/","@@" "char","nchar","varchar","nvarchar","fetch","insert","kill","open","select","alter","begin","cast","create","cursor " ","declare","delete","drop","end","exec","execute","sys","sysobjects","syscolumns","table","update"};

public static bool CheckInput(string parameter)
The above statement will detect most of the vulnerabilities in web applications. The vulnerabilities are cross scripting and SQL injection queries. A Meta character represents a meta-character that divides a vulnerable spot and a parenthesis determines whether or not to insert parentheses {"--","\n","\r","/","\*","\@"}. In another case SQL injection queries. Whenever these key words are {"char","nchar","varchar","nvarchar","fetch","insert","kill","open","select","alter","begin","cast","create","cursor","declare","delete","drop","end","exec","execute","sys","sysobjects","syscolumns","table","update"} add in normal user input to damage the entire database in web applications. The proposed system used the techniques of combined Static and Dynamic analysis to detect these keywords in web applications. Our system used the static analysis technique to protect the login page from the tautologies attack.

4.1.2. Prevention codding

if (objDS_MAS_UserRegistrations.Count > 0)
{
    Session["LoginName"] = objDS_MAS_UserRegistrations[0].Username;
    if (objDS_MAS_UserRegistrations[0].IsAdmin == "Y")
    {
        Response.Redirect("~/AdminPages/AdminHomePage.aspx");
    }
    else
    {
        BS_TRN_SQLInjectAttackDetails objBS_TRN_SQLInjectAttackDetails = BS_TRN_SQLInjectAttackDetailBase.SelectByField("InjectUserName", Session["LoginName"].ToString());
        if (objBS_TRN_SQLInjectAttackDetails.Count > 3)
        {
            Alert.Show("Your IP Blocked by Intrusion Detector Admin");
        }
        Else
        {
        }
    }
}
else
{
    Alert.Show("Invalid UserID/Password");
}
}

The proposed system used the Dynamic analysis technique to detect and protect the webpages from tokens.

private bool IsValidSubmitData()
{
    bool r = false;
    if (!string.IsNullOrEmpty(TxtName.Text))
    {
        if (!Regex.IsMatch(TxtName.Text, @"^[a-zA-Z'\-']\s{1,40}$"))
        {
            Alert.Show("Invalid First Name format");
        }
    }
}
return r;
}
else if (IntrusionDetector.CheckInput(TxtName.Text))
{
    Alert.Show( "suspicious Statements / Scripts Inputs are found ");
    TraceInject();
    return r;
}

Experimental results: Various experiments have been conducted to detection and prevent from SQL injection attacks. Previous SQL detection techniques are not capable to detect all injections. Like AMNESIA method is used to detect the tautologies and illegal/logical queries as well as union queries. But AMNESIA Method can’t detect the Stored Procedure method [Halfond, W.G (2005)]. SQL rand techniques can’t detect the illegal queries and stored procedure queries [Boyd, S. W (2004)]. WAVES techniques used to detect all possible Injection queries [Huang, Y; (2003)].

Figure 3.1 shows the user passes injection queries and scripts to message field in web page. The query is ‘'); drop table stud;--’. This query delete or drop the table structure and table content in stud table and damage the database.

Figure 3.2 shows the system administrator which in turn generates and sends a warning message to the user and stops all transaction that is issued from that user and blocks that IP address of the user. This prevents the database damage.

Fig 3.1. SQL Injection Attack

Fig 3.2. SQL injection Attack Detection and prevention
4.2. SQL injection reports

The administrator can maintain web application attacker’s details. This is one of the major advantages of the proposed system. This helps the administrator to view injection reports. This injection reports contains injection identification number, injection type, user identification, date, time, Internet protocol (IP) address, the web page number where the injection has taken place and so on. With the help of the injection reports the administrator warns the user. And the administrator will block the ip (Internet Protocol) address and user log in case of multiple injection attempts. The SQL injection reports will store the injection queries and cross scripts with the help of intrusion detector method the normal queries will not be stored under this SQL injection reports.

5. Result

The proposed system has been implemented and proves that the techniques fully detect and prevent the Injection attacks and cross scripting in web applications and databases.

<table>
<thead>
<tr>
<th>Application</th>
<th>Injection Queries</th>
<th>Detection and Prevention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample application</td>
<td>58</td>
<td>100%100%</td>
</tr>
</tbody>
</table>

The huge majority of solutions are based on centralized mechanisms with little capacity to work in distributed and dynamic environments and the above techniques implementation is too complex in real web applications. The Existing System does support some specific languages [Asp.net and java] to detect the injections.

This system technique is implemented in real web applications. The proposed System is used to identify user that who passes the unwanted queries to the web application and automatically detects and prevents it and administrator can directly view the injection reports. This injection reports contains injection Identification, injection name, user, date, time, Internet Protocol address ,injection page .By the reports the administrator warn the user by message by repeating the process(passing injection queries) again and again administrator will block the ip address and user log in. This is very effective for blocking SQL injection attacks as the mechanism uses a strategy based on anomaly detection and it is possible to exploit the advantages of both strategies in order to classify the SQL queries in a more reliable way.

6. Conclusion

SQL injection attacks and cross scripting increased in real web applications. The proposed system detects and prevents SQL injection queries and cross scripts, and view SQL injection attacking reports.SQL injection attacking reports used to identify user that who passes the unwanted queries to the web application. Our system represented, Static analysis can used to detect and prevent SQL injection attacks in compile time and Dynamic analysis can used to detect and injection queries prevent in runtime. The current system developed in Asp.net with C# and Microsoft SQL server 2008.
References


