Prevention Mechanism For Sqlia Using Hash Key With XML

M. Shanmugham
Assistant Professor
Christ College of Engg & Tech.

C. Maleappane Lawrence
Assistant Professor
Christ College of Engg & Tech.

Abstract - The past decades SQL Injection attacks have been used to read sensitive data form database and preventive methods are blocking the SQLIA. The present research has been given a cryptographic solution with the mechanism through HASH Function which has been stored in XML File and verified directly.

Keywords - SQL Injection attacks, HASH function, XML Path.

1. INTRODUCTION

In present life, the information is the most important asset for every organization and provides proper information security. SQL Injection Attacks (SQLIA's) are considered as top level threats for web application and windows application. For example: A group of hackers hacked the website using SQL Injection and other breached Administration Site. The Evolution of security used in detecting and preventing SQLIA's. SQL Injection is the hacking technique which attacks through web application's input field to gain access the resources in the database using SQL Statements. Most of the web applications are secured through Login Page which provides Username and Password. The attacker's use vulnerable user inputs for embedding the commands and gets executing the application. The attacker using the direct access the database for underlying application or alter the confidentially information. So, overcome this situation the prevention should be made on SQL Injection Attacks (SQLIA).

SQL injection is a kind of malicious attack so that it can read sensitive data from the database. Different types of SQL injection and preventing methods are available nowadays. The proposed novel techniques which provide a cryptographic solution to this issue in the login phase. The advantage of new approach against the existing mechanism are the user details are stored to the database along with cipher text (hash key value) which is generated by the hash function using username and password. The fixed length hash function value is simultaneously stored to the xml file. In the login phase, the authorized persons hash key value is checked in the xml file instead username and password are directly executed in the database.

2. RELATED WORK

One of the techniques prevents the SQL injection using web service and other technique involves Encryption based solution.

3. PROPOSED METHODOLOGY

3.1 Proposed Technique Executed in Cryptographic solution in XML File:

This Technique is used to prevent SQLIA's using XMLPATH Cryptographic Solution. At the user registration phase, the Hash Function writes the Fixed length Cryptographic value to the XML File using Name and Password. The user login application is redirected to XML file instead of moving to the database. Moreover, this technique proved to be efficient and a low overhead on the database. The contribution of this work is as follows:

A new automated technique for preventing SQLIA's where no interaction and no dealing with the sensible data in the database which requires the KeyGen function returns Cipher text value and Generate XML file. It is used for the temporary storage of non-sensitive data's. The hash key generation function and the generating the XML file. Innovative technique (figure: 1) monitors dynamically generating the Hash key values and stored it to the XML file. It may also possible to violate the sensible data at the time of the verification. So, we are verifying the authorized person in the XML file.

3.2 This proposed technique consists of two filtration models

```
private string KeyGen(string sName,string sPassword)

{
    var x = "@@" + sName + sPassword + "@@";
    var hash = x.GetHashCode();
    return hash.ToString();
}
private void GenerateXml()
```

```
   {
SqlConnection con = new SqlConnection(@"server=.;
database=Injection; Integrated security= true;");

DataSet ds = new DataSet();

try
{
   con.Open();
   SqlDataAdapter da =new SqlDataAdapter("Select cKey
From Customer1", con);
  da.Fill(ds,"Hash");

ds.WriteXml(Server.MapPath("Xml")+"\\HashFile.xml");
 }
 catch (Exception ex)
 {
     Response.Write(ex.Message);
  }
 finally
 {
    con.Close();
  }
}
```

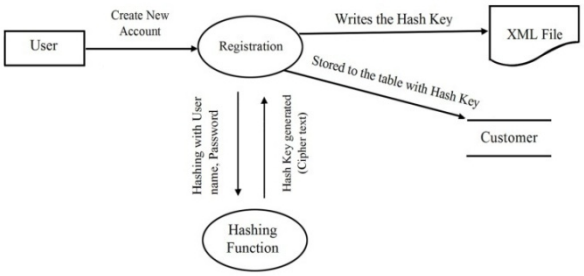
Fig 1: Shows the XML File generation of Hash Key value protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)

```
{
   string logingKey = KeyGen(Login1.UserName,
Login1.Password);
DataSet ds = new DataSet();
ds.ReadXml(Server.MapPath("Xml") + "\\HashFile.xml");
bool validateLogin = false;
foreach (DataRow item in ds.Tables[0].Rows)
   {
    if (item["ckey"].ToString() == logingKey)
    {
         validateLogin = true;
          break;
     }
   }
 if (validateLogin)
 {
     Response.Redirect("CustomerInfo.aspx");
  }
 else
 {
   Response.Write("Login Failed");
  }
}
```

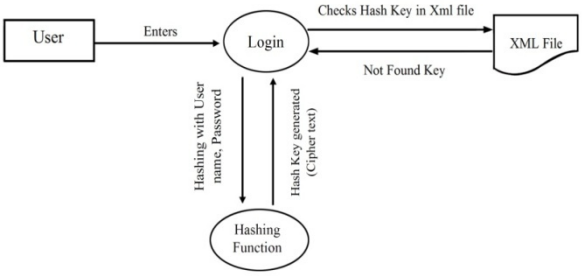
Fig 2: Verifying the authentication of Hash Key in the XML File.

4. SYSTEM MODEL

4.1 Registration Phase



4.2 Login Phase



5. CONCLUSION

The sensible data are not checked directly, they are verified in the form of cipher text in the XML file. The Hash Key values are also stored to the database at the time of registration for future use.

6. FUTURE SCOPES

We intend to analyze the input string which is given as input to the web form by a user. The independent analysis of the input string will give the greater performance to protect SQL Injection. If user input is properly analyzed, we can protect SQL injection in a better way. These results show that our technique represents a promising approach to countering SQLIA's and motivate further work in this direction.

REFERENCES

[1] Indrani Balasundaram et al. "An authentication scheme for preventing SQL injection attack using hybrid encryption

(PSQLIA-HBE)" Euro journal publishing, 2011.

[2] XuePing-Chen "SQL injection attack and guard technical research" 2011.

[3] F. Bouma, "Stored Procedures are Bad, O'kay," Technical report, Asp.Net Weblogs, November 2003. http://weblogs.asp.net/fbouma/archive/2003/11/18/38178.aspx

[4] Mayank Namdev "Review of SQL Injection Attack and Proposed Method for Detection and Prevention of SQLIA" July 2012.

[5] Manju Khari "SQLIA Detection and Prevention Approaches: A Survey" May 2013

AUTHOR'S PROFILE

M. SHANMUGHAM has received his Master of Computer Applications degree from Pope John Paul II college of Education, Pondicherry in the year 2007. At present he is working in Christ College of Engineering and Technology. His area of interest Database Management Systems and Object Oriented Programming System.

C. MALEAPPANE LAWRENCE has received his Master of Computer Applications degree from Rajiv Gandhi college of Engineering, Pondicherry in the year 2010 and also M.Tech degree in Computer Science from SRM University, Chennai in the year 2015. At present he is working in Christ College of Engineering and Technology. His area of interest Software Engineering and Object Oriented Programming Concepts.