Chapter 1

Introduction

The application “Salted Password Hashing” could be used to employ a hack free environment in the field of web development, Bank accounts, etc. The activities performed and information stored by the employees in the above-mentioned fields are obviously too confidential. Hence, the account database must be protected efficiently.

The most commonly used technique to protect database is Hashing. But the passwords can still be hacked using hacking techniques like rainbow table, dictionary method, etc. Thus, salted password hashing serves the purpose of avoiding hacking. The application includes the process of generating a random string called salt, which is appended to the password and its hash is stored in the database.

1.1 Motivation of the Project

The presently adopted way of protecting the databases is Hashing technique. There is a little chance of passwords being hacked using hash i.e., by guessing the password and comparing with hash. Is there way to avoid this chance of password hacking? The application “Salted password hashing” can be used in such case and

- Protect account database
- Can be implemented to secure important folders and files
- Double security; username also is protected using the application

1.2 Problem Statement

If the user is using his/her information without any security, there are number of chances that the information may be leaked. Hence, by implementing the application “Salted password hashing” user is protected from hacking.

Chapter 2

System Requirements

2.1 Hardware System Configuration:

Processor - Intel Core i3

Speed - 1.8 GHz

RAM - 256 MB (min)

Hard Disk - 10 GB

2.2 Software System Configuration:

Operating System - Windows 7/8/10

Programming Language - C#

Compiler - Microsoft Visual Studio 2017

Chapter 3

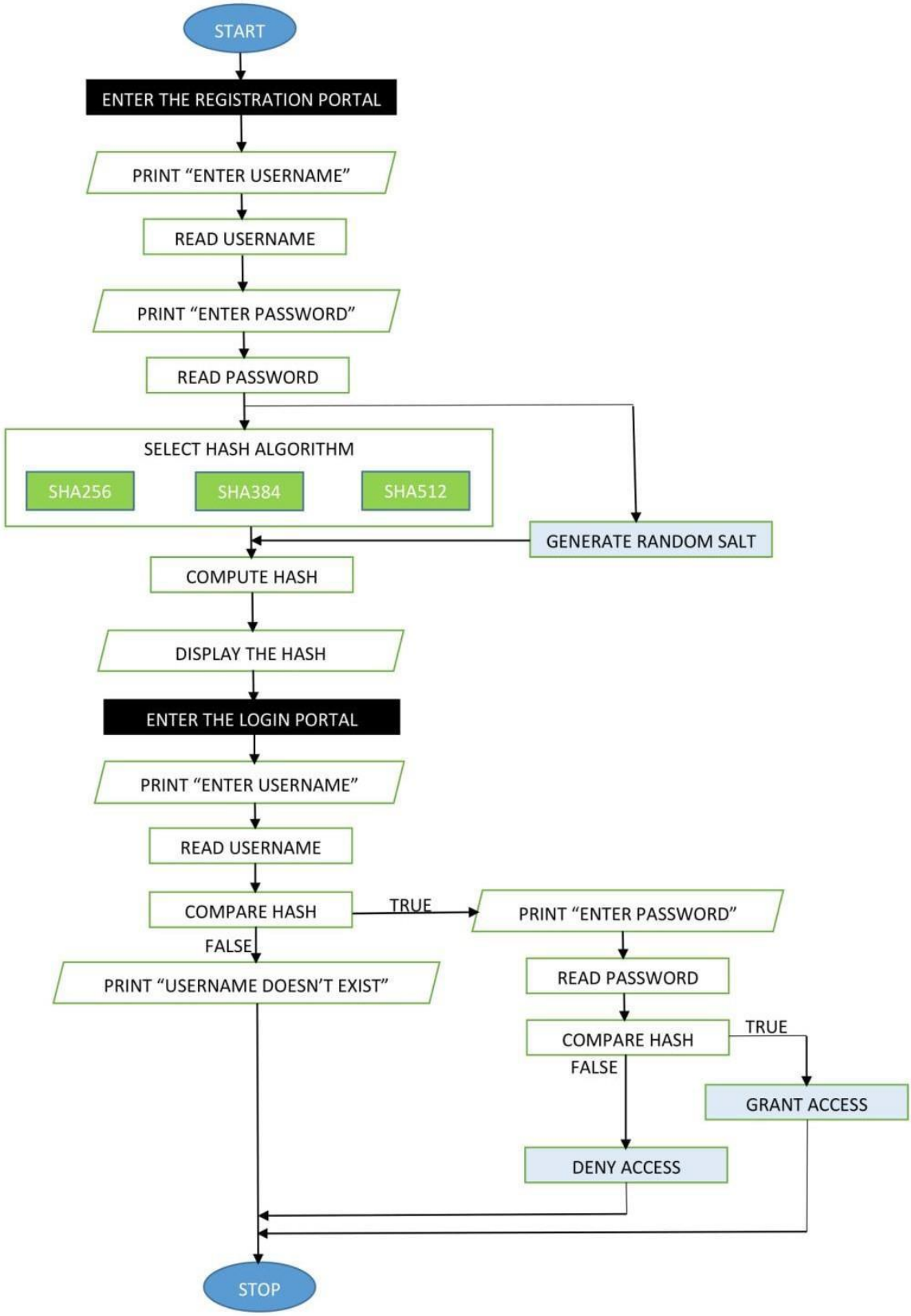
System Design

3.1 Procedure

- Assume the password to be “JOHN123” which is to be stored in the database.
- Generate a random string so called salt. Assume the salt generated to be “ABCD”
- Append or prepend the salt to the password i.e., “ABCDJOHN123” or “JOHN123ABCD”.
- Generate its respective hash code.
- The salted hash for the “JOHN123ABCD” would look like

  B109F3BBBC244EB82441917ED06D618B9008DD09B3BEFD1B5E07394C706A8BB980B1D7785E5976EC049B46DF5F1326AF5A2EA6D103FD07C95385FFAB0CACBC86
- Only way of hacking; rainbow table; guess the password.
- Salt: random; impossible to guess.
- Hence, revealing the hash of salt also doesn’t help in hacking.

3.2 Architecture



3.3 Algorithm

Step 1 :- Start

Step 2 :- Enter the Registration portal.

Step 3 :- Enter the username and password.

Step 4 :- Select the hash algorithm

- SHA256, SHA384, SHA512; Submit

Step 5 :- Compute the selected hash and display.

Display message: Click here to login.

Step 6 :- Enter the Login portal.

Step 7 :- Enter the username and password; Submit.

Step 8 :- Compute the entered password's hash

-Compare with hash of the registered password.

Step 9 :- if comparison results equal

-grant the access.

if comparison results not equal

-display error check the username and password.

Step 10 :- Stop

3.4 Code and Implementation

Form 1:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SALTED_HASHING;
```

```
namespace WindowsFormsApp6
{
   public partial class Form1 : Form
   {
      public static string setvaluefortext1 = "";
      public static string setvaluefortext2 = "";
      public static string setvaluefortext3 = "";
      public static string setvaluefortext4 = "";

      public Form1()
      {
         InitializeComponent();
      }


      private void label1_Click(object sender, EventArgs e)
      {
         this.Hide();
         setvaluefortext1 = textBox2.Text;
         setvaluefortext2 = comboBox1.Text;
         setvaluefortext3 = textBox4.Text;
         setvaluefortext4 = comboBox2.Text;

         Form frm2=new Form2();
         frm2.Show();


      }

      private void button1_Click(object sender, EventArgs e)
      {
         switch (comboBox1.Text)
         {
            case "SHA256":
               textBox2.Text = HASHING.computehash(textBox1.Text,
supported_ha.SHA256, null);
               MessageBox.Show("Please go to the Login Page");
               break;

            case "SHA384":
               textBox2.Text = HASHING.computehash(textBox1.Text,
supported_ha.SHA384, null);
               MessageBox.Show("Please go to the Login Page");
               break;

            case "SHA512":
               textBox2.Text = HASHING.computehash(textBox1.Text,
supported_ha.SHA512, null);
               MessageBox.Show("Please go to the Login Page");
```

```
                break;
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {

            switch (comboBox1.Text)
            {
                case "SHA256":
                    textBox4.Text = HASHING.computehash(textBox3.Text,
                                    supported_ha.SHA256, null);
                    MessageBox.Show("Enter the Password");
                    break;

                case "SHA384":
                    textBox4.Text = HASHING.computehash(textBox3.Text,
                                    supported_ha.SHA384, null);
                    MessageBox.Show("Enter the Password");
                    break;

                case "SHA512":
                    textBox4.Text = HASHING.computehash(textBox3.Text,
                                    supported_ha.SHA512, null);
                    MessageBox.Show("Enter the Password");
                    break;
            }

        }


    }
}
```

Form 2:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SALTED_HASHING;

namespace WindowsFormsApp6
{
```

```
public partial class Form2 : Form
{
    OpenFileDialog os = new OpenFileDialog();

    public Form2()
    {
        InitializeComponent();

    }

    private void button1_Click(object sender, EventArgs e)
    {
        int a = 1;

        switch (Form1.setvaluefortext2)
        {
            case "SHA256":
                a = (HASHING.confirm(textBox1.Text, Form1.setvaluefortext1,
                    supported_ha.SHA256)) ? 0 : 1;
                if (a == 0)
                {
                    MessageBox.Show("Authenticated");
                    os.OpenFile();

                }
                else
                    MessageBox.Show("Authentication failed");
                break;

            case "SHA384":
                a = (HASHING.confirm(textBox1.Text, Form1.setvaluefortext1,
                                    supported_ha.SHA384)) ? 0 : 1;
                if (a == 0)
                {
                    MessageBox.Show("Authenticated");
                    os.ShowDialog();
                }
                else
                    MessageBox.Show("Authentication failed");
                break;

            case "SHA512":

                        a = (HASHING.confirm(textBox1.Text, Form1.setvaluefortext1,
                                    supported_ha.SHA512)) ? 0 : 1;
                if (a == 0)
                {
                    MessageBox.Show("Authenticated");
                    os.ShowDialog();
```

```
            }
            else
               MessageBox.Show("Authentication failed");
            break;
      }
   }



   private void button2_Click(object sender, EventArgs e)
   {
      int a = 1;

      switch (Form1.setvaluefortext2)
      {
         case "SHA256":
            a = (HASHING.confirm(textBox2.Text, Form1.setvaluefortext3,
                                 supported_ha.SHA256)) ? 0 : 1;
            if (a == 0)
            {
               MessageBox.Show("Enter the Password");
            }
            else
               MessageBox.Show("Username doesn't exist");
            break;

         case "SHA384":
            a = (HASHING.confirm(textBox2.Text, Form1.setvaluefortext3,
                                 supported_ha.SHA384)) ? 0 : 1;
            if (a == 0)
            {
               MessageBox.Show("Enter the Password");
            }
            else
               MessageBox.Show("Username doesn't exist");
            break;

         case "SHA512":
            a = (HASHING.confirm(textBox2.Text, Form1.setvaluefortext3,
                                 supported_ha.SHA512)) ? 0 : 1;
            if (a == 0)
            {
               MessageBox.Show("Enter the Password");
            }
            else
               MessageBox.Show("Username doesn't exist");
            break;
      }
   }
```

```
        private void label3_Click(object sender, EventArgs e)
        {
            MessageBox.Show(Form1.setvaluefortext4);
        }

    }
}
```

Hashing code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Security.Cryptography;

namespace SALTED_HASHING
{
    public enum supported_ha
    {
        SHA256, SHA384, SHA512
    }
    class HASHING
    {
        public static string computehash(string plaintext, supported_ha hash, byte[] salt)
        {
            int minsaltlength = 4;
            int maxsaltlength = 16;

            byte[] saltbytes = null;

            if (salt != null)
            {
                saltbytes = salt;
            }

            else
            {
                Random r = new Random();
                int saltlength = r.Next(minsaltlength, maxsaltlength);
                saltbytes = new byte[saltlength];
                RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();
                rng.GetNonZeroBytes(saltbytes);
                rng.Dispose();
            }
```

```
        byte[] plaindata = ASCIIEncoding.UTF8.GetBytes(plaintext);
        byte[] plaindataandsalt = new byte[plaindata.Length+saltbytes.Length];

        for (int x = 0; x < plaindata.Length; x++)
            plaindataandsalt[x] = plaindata[x];

        for (int n = 0; n < saltbytes.Length; n++)
            plaindataandsalt[plaindata.Length + n] = saltbytes[n];

        byte[] hashvalue = null;

        switch (hash)
        {
            case supported_ha.SHA256:
                SHA256Managed sha = new SHA256Managed();

                hashvalue = sha.ComputeHash(plaindataandsalt);
                sha.Dispose();
                break;

            case supported_ha.SHA384:
                SHA384Managed sha1 = new SHA384Managed();
                hashvalue = sha1.ComputeHash(plaindataandsalt);
                sha1.Dispose();
                break;

            case supported_ha.SHA512:
                SHA512Managed sha2 = new SHA512Managed();
                hashvalue = sha2.ComputeHash(plaindataandsalt);
                sha2.Dispose();
                break;
        }

        byte[] result = new byte[hashvalue.Length + saltbytes.Length];

        for (int x = 0; x < hashvalue.Length; x++)
            result[x] = hashvalue[x];

        for (int n = 0; n < saltbytes.Length; n++)
            result[hashvalue.Length + n] = saltbytes[n];

        return Convert.ToBase64String(result);


    }

    public static bool confirm(string plaintext, string hashvalue, supported_ha hash)
    {
        byte[] hashbytes = Convert.FromBase64String(hashvalue);
```

```
            int hashsize = 0;

            switch (hash)
            {
               case supported_ha.SHA256:
                  hashsize = 32;
                  break;

               case supported_ha.SHA384:
                  hashsize = 48;
                  break;

               case supported_ha.SHA512:
                  hashsize = 64;
                  break;
            }

            byte[] saltbytes = new byte[hashbytes.Length - hashsize];

            for (int x = 0; x < saltbytes.Length; x++)
               saltbytes[x] = hashbytes[hashsize + x];

            string newhash = computehash(plaintext, hash, saltbytes);

            return (hashvalue == newhash);
         }
      }

}
```

Chapter 4

Results and Discussion

The application is developed in order to provide the user a hack free environment. Thus, reducing the risks of being hacked frequently. This application finds its main application in IT fields, web developing companies and Space shuttle systems, etc. In this application if the user enters the wrong username it was show you the caution as you have entered the wrong username as well as same in the password section it will show you the caution box as you have entered the wrong password. This application is user friendly in environment and its more reliable which is free of cost. There are various methods to saving data link has, salt SHA256SHA512 .

4.1 Output(Snap-shots)

Enter the registration portal

Fig.4.1.1

Enter the username

Fig.4.1.2

Select the hashing algorithm that is depend on the security level you require

Fig.4.1.3

Once the password is entered and submitted it generates hash code and asks to go to login page.

Fig.4.1.4

Enter the login portal.

Fig.4.1.5

Enter the username and password if the user is not registered database then: error occurs user does not exist.

Fig.4.1.6

If the username and password matches according to the database then authentication is given.

Fig.4.1.7

The user folder opens.

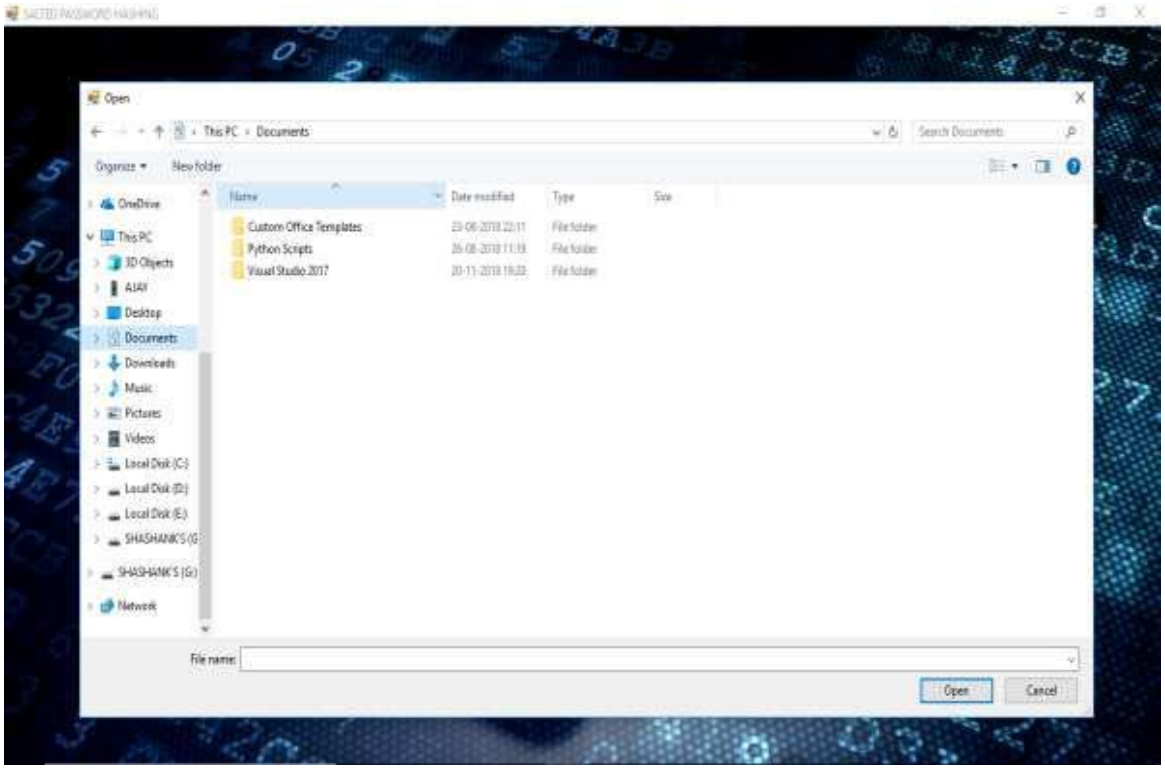


Fig.4.1.8

If the username and password does not matches according to the database then : error occurs authentication fail.

Fig.4.1.9

If you forget the password you can remember from the caution that it will show as forgot password?

Fig.4.1.10

Conclusion

Passwords and information stored are too confidential in the fields of web development, finance, IT companies, etc. This application program makes the password database more secure and provide a hack free environment. It is user friendly also and provides a hassle-free experience of security.

References

The following websites were helpful during the development of the project:

- ✓ ANCI programming
- ✓ Sanfoundry.com
- ✓ Wikipedia.com
- ✓ Stack overflow.com
- ✓ www.youtube.com
- ✓ www.courseera.com
- ✓ https::msdn/microsoft/help.com