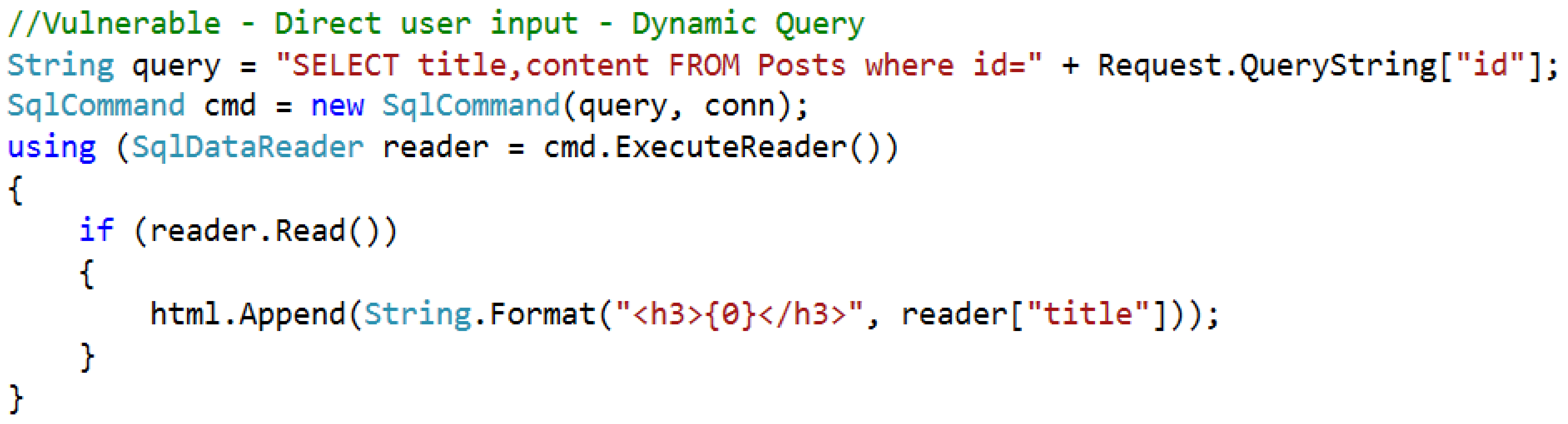
Secure Coding – ASP.Net (C#)

CYBER SECURITY &
PRIVACY FOUNDATION

A1 - SQL Injection

Vulnerable Code

Dynamic Query:



```
//Vulnerable - Direct user input - Dynamic Query
String query = "SELECT title,content FROM Posts where id=" + Request.QueryString["id"];
SqlCommand cmd = new SqlCommand(query, conn);
using (SqlDataReader reader = cmd.ExecuteReader())
{
    if (reader.Read())
    {
        html.Append(String.Format("<h3>{0}</h3>", reader["title"]));
    }
}
```

Prevention

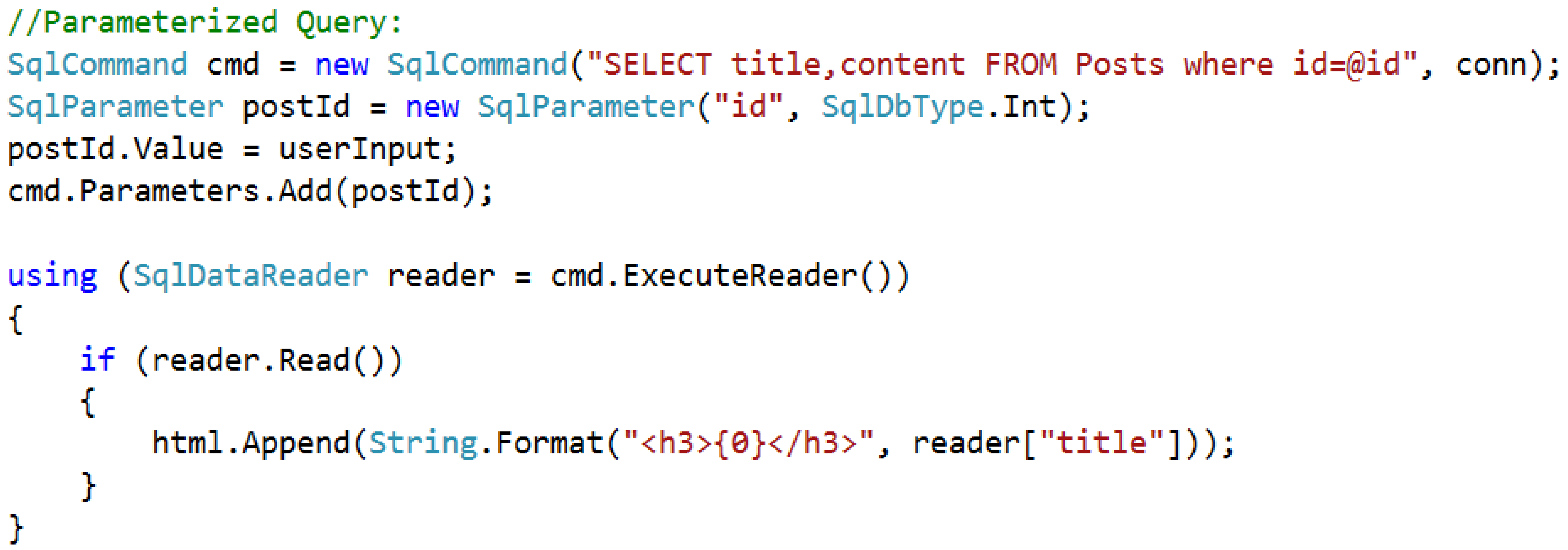
Basic Steps:

Filter user input (remove special characters if not needed)
Convert to integer or related data type(if it is not string)

Recommended:

Parameterized SQL Query(also known as Prepared statement)

Parameterized SQL Query (also known as Prepared statement)



```
//Parameterized Query:
SqlCommand cmd = new SqlCommand("SELECT title,content FROM Posts where id=@id", conn);
SqlParameter postId = new SqlParameter("id", SqlDbType.Int);
postId.Value = userInput;
cmd.Parameters.Add(postId);

using (SqlDataReader reader = cmd.ExecuteReader())
{
    if (reader.Read())
    {
        html.Append(String.Format("<h3>{0}</h3>", reader["title"]));
    }
}
```

Least Privilege Account

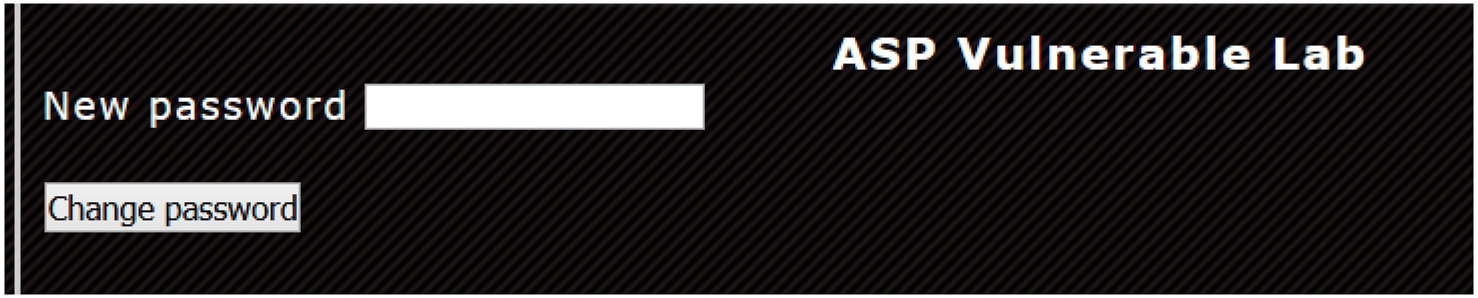
Use Database account with least privilege.

Example:
If you are using account called “user1” to access “db1”,the user1 should only have privilege to “db1” and should not able to access the “db2”

A2 - Broken Authentication and Session Management

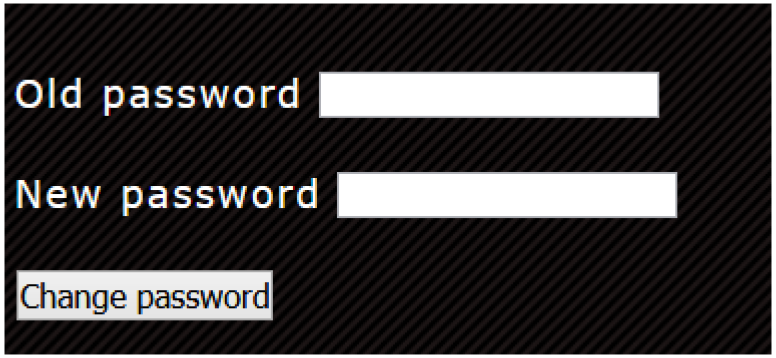
Unverified Password Change

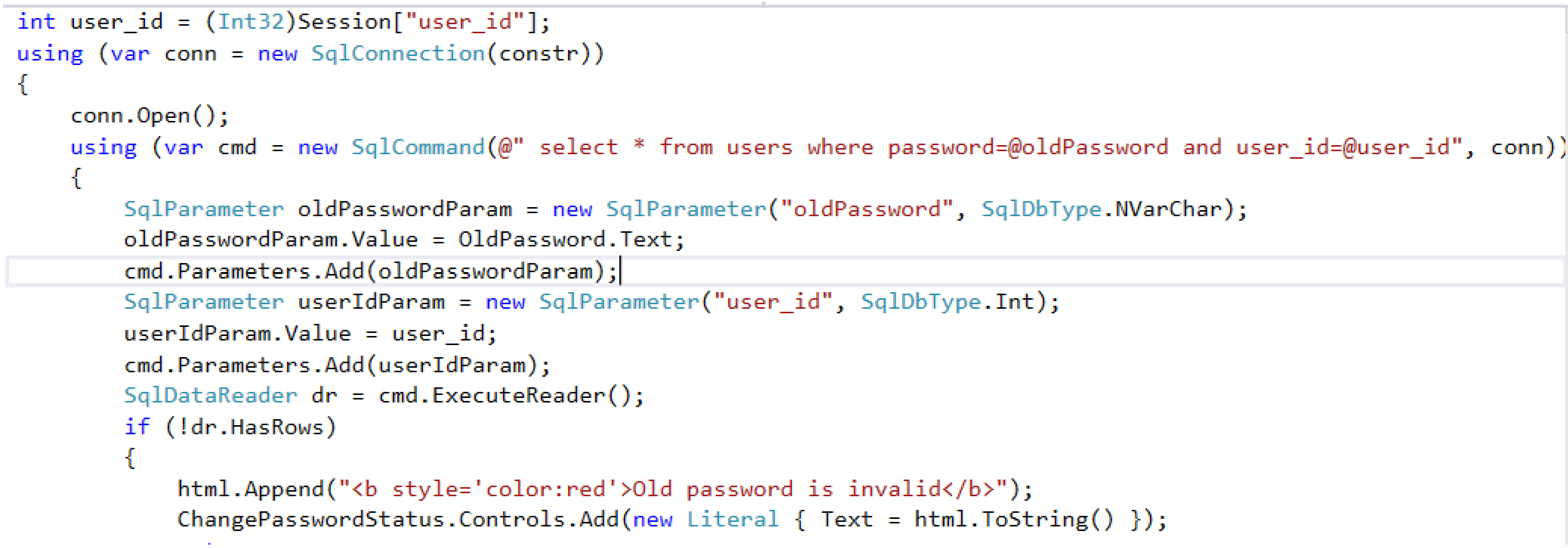
- Failed to ask or verify the Old Password



```
int user_id = (Int32)Session["user_id"];
string updSql = @"UPDATE users SET password = '" + NewPassword.Text + "' where id=" + user_id;
using (var cmd = new SqlCommand(updSql, conn))
{
    if (cmd.ExecuteNonQuery() > 0)
    {
        html.Append("<b style='color:green'>Updated</b>");
    }
    else
    {
        html.Append("<b style='color:red'>No changes made</b>");
    }
}
```

Verify the Old Password





```
int user_id = (Int32)Session["user_id"];
using (var conn = new SqlConnection(constr))
{
    conn.Open();
    using (var cmd = new SqlCommand(@" select * from users where password=@oldPassword and user_id=@user_id", conn))
    {
        SqlParameter oldPasswordParam = new SqlParameter("oldPassword", SqlDbType.NVarChar);
        oldPasswordParam.Value = OldPassword.Text;
        cmd.Parameters.Add(oldPasswordParam);
        SqlParameter userIdParam = new SqlParameter("user_id", SqlDbType.Int);
        userIdParam.Value = user_id;
        cmd.Parameters.Add(userIdParam);
        SqlDataReader dr = cmd.ExecuteReader();
        if (!dr.HasRows)
        {
            html.Append("<b style='color:red'>Old password is invalid</b>");
            ChangePasswordStatus.Controls.Add(new Literal { Text = html.ToString() });
```

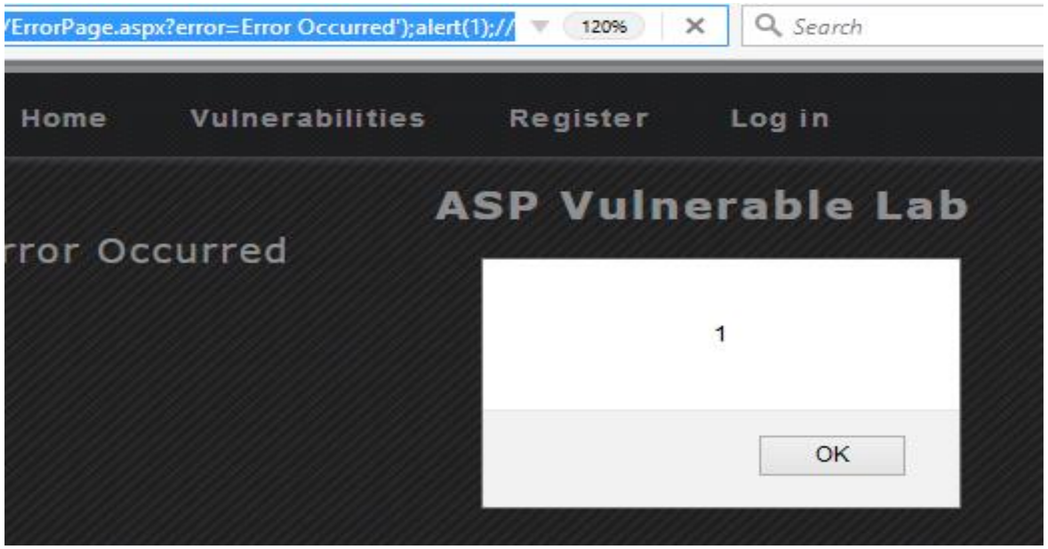
A3 - XSS

Example Vulnerable Code

```
protected void DisplayError()
{
    StringBuilder html = new StringBuilder();
    String msg = Request.QueryString["error"];
    html.Append("<script>document.write('" + msg + "');</script>");
    ErrorMessage.Controls.Add(new Literal { Text = html.ToString() });
}
```

XSS POC:

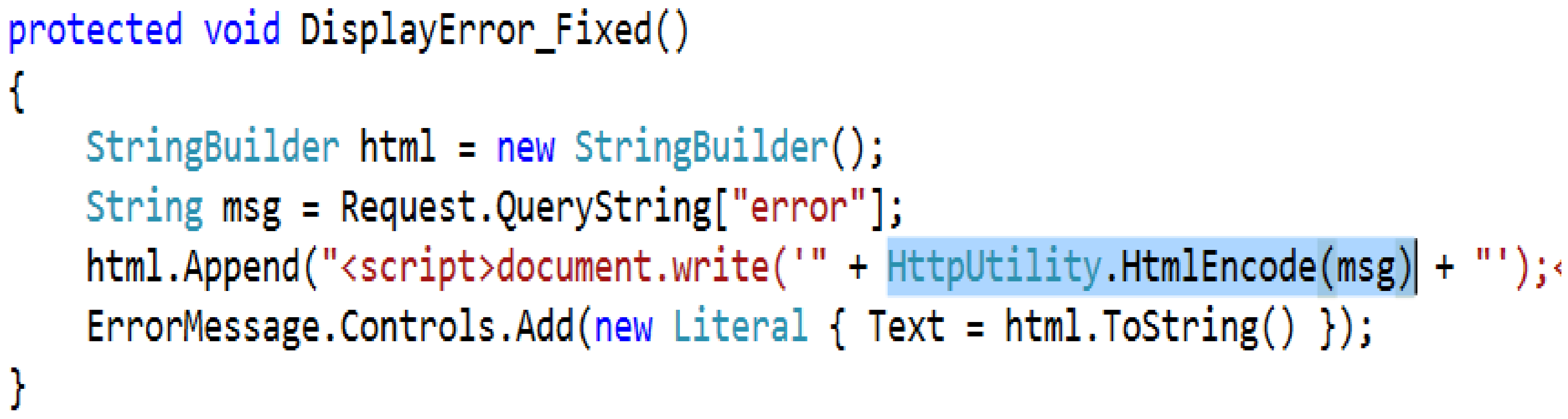
- http://localhost/ErrorPage.aspx?error=Error Occurred');alert(1);//



Prevention - Convert Special characters to HTML entities

Character	Entity
<	&lt;
>	&gt;
&	&amp;
“	&quot;
‘	&#x27;
/	&#x2F;

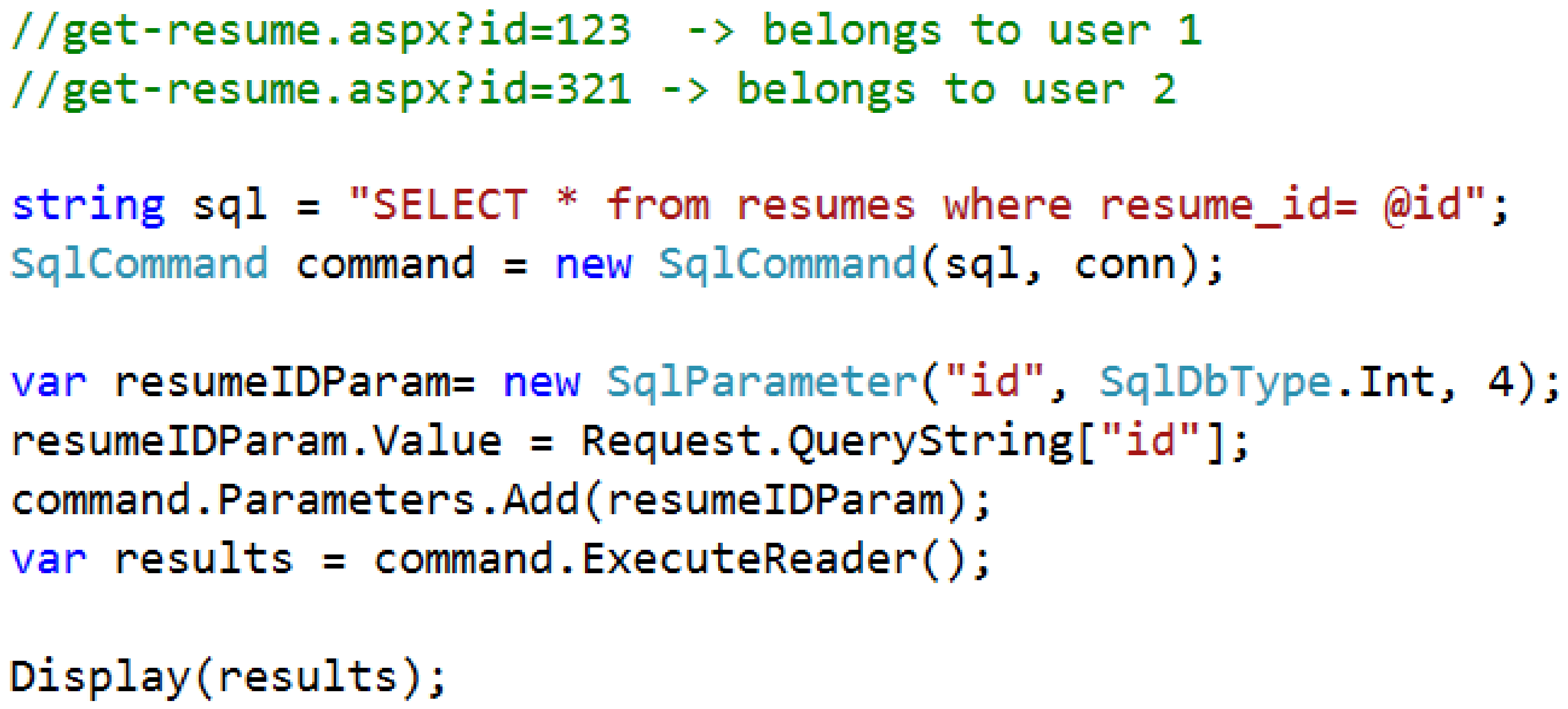
Prevention - HttpUtility.HtmlEncode



```
protected void DisplayError_Fixed()
{
    StringBuilder html = new StringBuilder();
    String msg = Request.QueryString["error"];
    html.Append("<script>document.write('" + HttpUtility.HtmlEncode(msg) + "');<
    ErrorMessage.Controls.Add(new Literal { Text = html.ToString() });
}
```

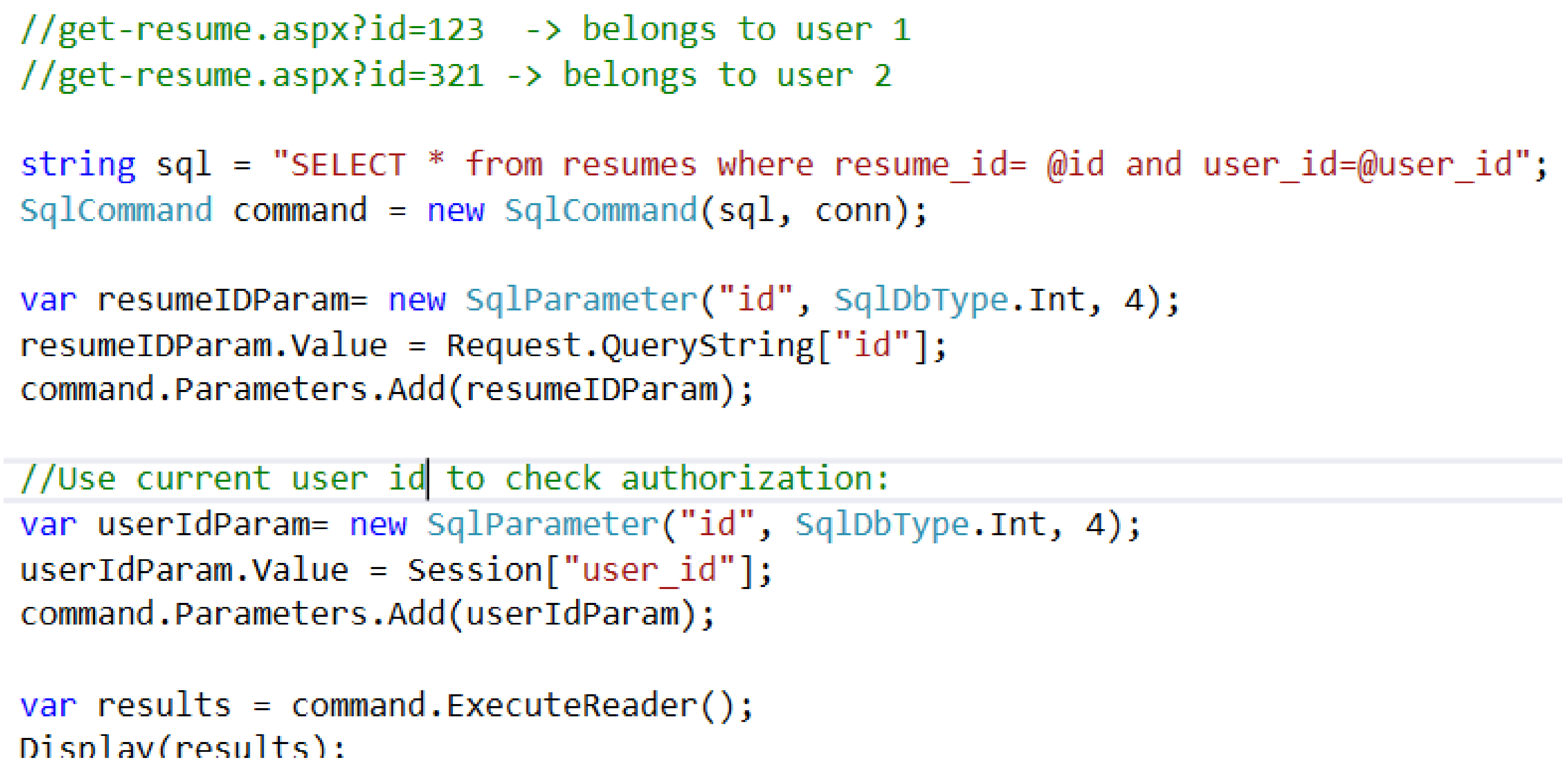
A4 – Insecure Direct Object Reference

Example Vulnerable Code



```
//get-resume.aspx?id=123  -> belongs to user 1
//get-resume.aspx?id=321 -> belongs to user 2

string sql = "SELECT * from resumes where resume_id= @id";
SqlCommand command = new SqlCommand(sql, conn);

var resumeIDParam= new SqlParameter("id", SqlDbType.Int, 4);
resumeIDParam.Value = Request.QueryString["id"];
command.Parameters.Add(resumeIDParam);
var results = command.ExecuteReader();

Display(results);
```

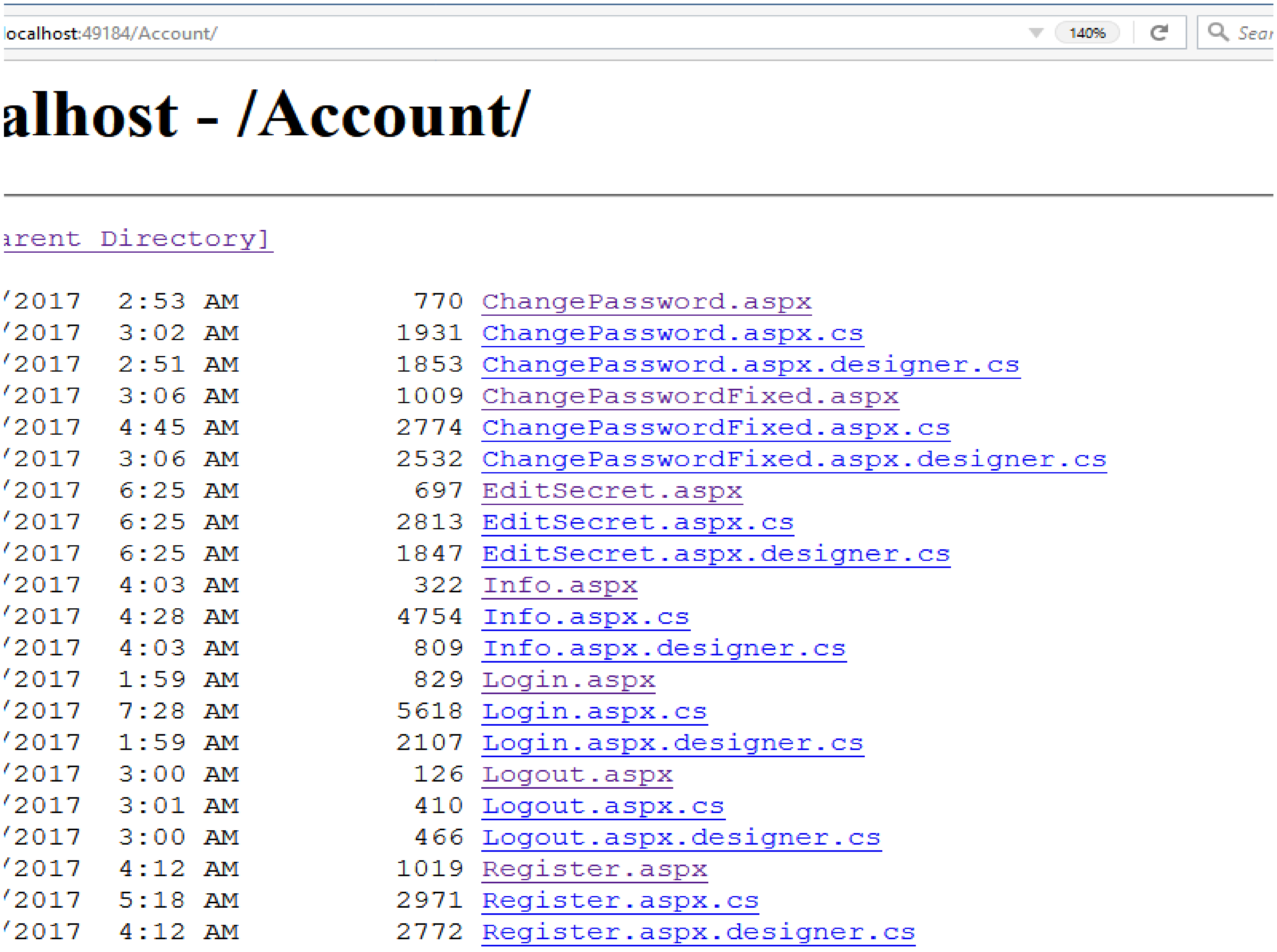
Example Secure Code



```
//get-resume.aspx?id=123  -> belongs to user 1
//get-resume.aspx?id=321 -> belongs to user 2

string sql = "SELECT * from resumes where resume_id= @id and user_id=@user_id";
SqlCommand command = new SqlCommand(sql, conn);

var resumeIDParam= new SqlParameter("id", SqlDbType.Int, 4);
resumeIDParam.Value = Request.QueryString["id"];
command.Parameters.Add(resumeIDParam);

//Use current user id to check authorization:
var userIdParam= new SqlParameter("id", SqlDbType.Int, 4);
userIdParam.Value = Session["user_id"];
command.Parameters.Add(userIdParam);

var results = command.ExecuteReader();
Display(results);
```

A5 – Security Misconfiguration

Directory Listing

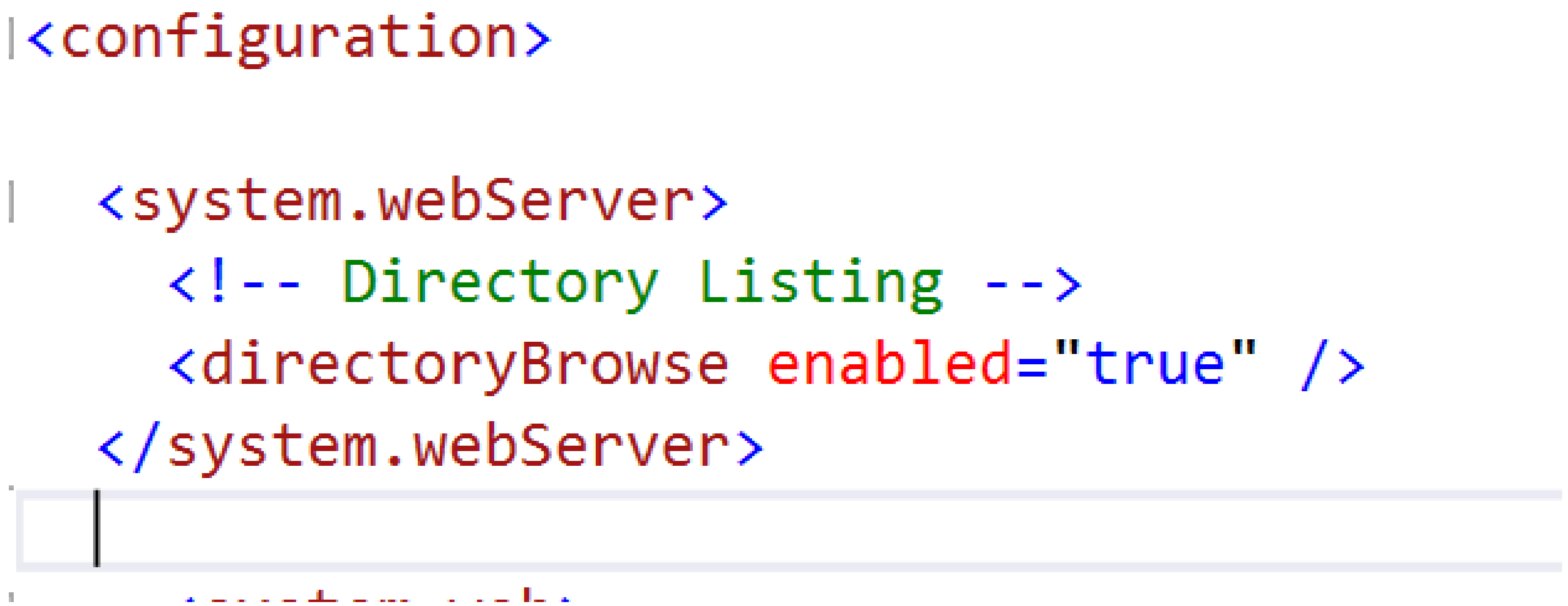


```
localhost:49184/Account/
```

alhost - /Account/

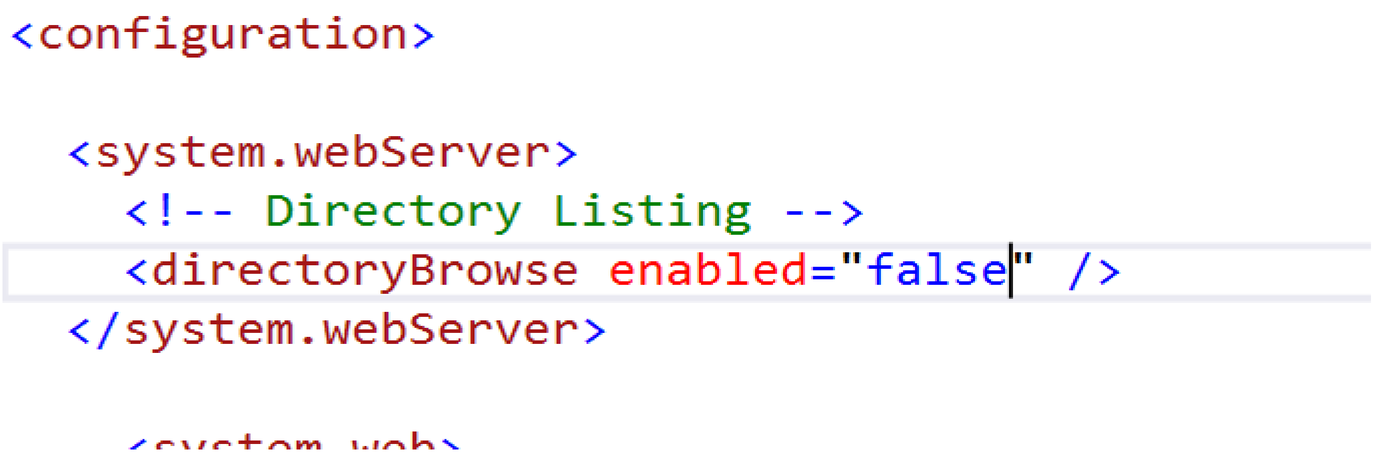
```
arent Directory]

'2017   2:53 AM          770 ChangePassword.aspx
'2017   3:02 AM         1931 ChangePassword.aspx.cs
'2017   2:51 AM         1853 ChangePassword.aspx.designer.cs
'2017   3:06 AM         1009 ChangePasswordFixed.aspx
'2017   4:45 AM         2774 ChangePasswordFixed.aspx.cs
'2017   3:06 AM         2532 ChangePasswordFixed.aspx.designer.cs
'2017   6:25 AM          697 EditSecret.aspx
'2017   6:25 AM         2813 EditSecret.aspx.cs
'2017   6:25 AM         1847 EditSecret.aspx.designer.cs
'2017   4:03 AM          322 Info.aspx
'2017   4:28 AM         4754 Info.aspx.cs
'2017   4:03 AM          809 Info.aspx.designer.cs
'2017   1:59 AM          829 Login.aspx
'2017   7:28 AM         5618 Login.aspx.cs
'2017   1:59 AM         2107 Login.aspx.designer.cs
'2017   3:00 AM          126 Logout.aspx
'2017   3:01 AM          410 Logout.aspx.cs
'2017   3:00 AM          466 Logout.aspx.designer.cs
'2017   4:12 AM         1019 Register.aspx
'2017   5:18 AM         2971 Register.aspx.cs
'2017   4:12 AM         2772 Register.aspx.designer.cs
```

Vulnerable Configuration

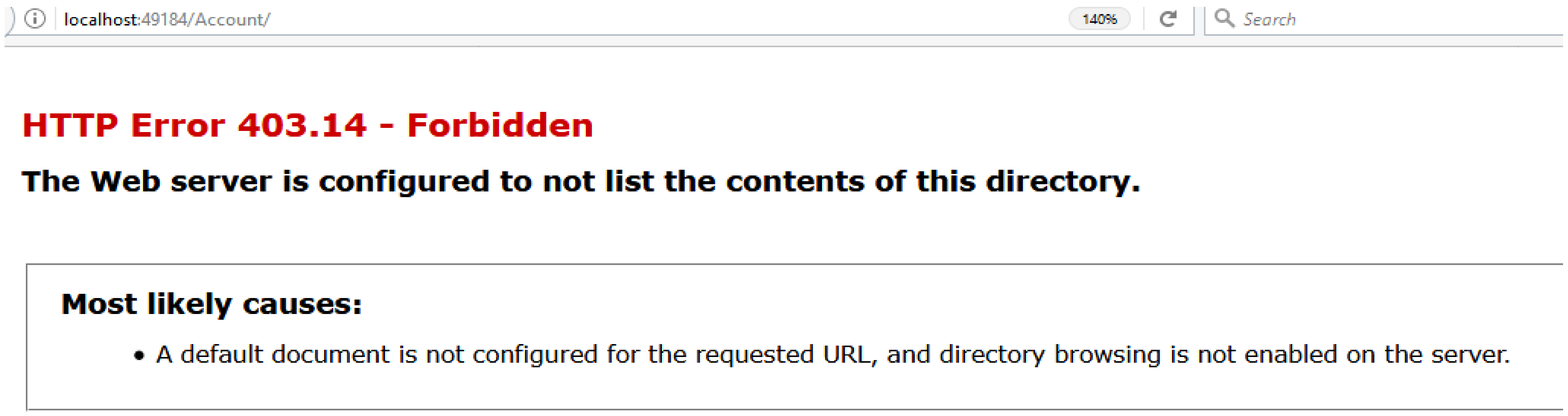


```
<configuration>

  <system.webServer>
    <!-- Directory Listing -->
    <directoryBrowse enabled="true" />
  </system.webServer>
```

Disabling Directory Listing



```
<configuration>

  <system.webServer>
    <!-- Directory Listing -->
    <directoryBrowse enabled="false" />
  </system.webServer>
```



localhost:49184/Account/

HTTP Error 403.14 - Forbidden

The Web server is configured to not list the contents of this directory.

Most likely causes:

- A default document is not configured for the requested URL, and directory browsing is not enabled on the server.

A6 Sensitive Data Exposure - Broken Cryptography

Prevention – Bcrypt Hashing(with salt)

C# Implementation of Bcrypt Library can be found here:

https://bcrypt.codeplex.com/

//Storing password: register.aspx

```
hashedPassword = Bcrypt.HashPassword(password);
storeUserDetailsInDB(username, hashedPassword );
```

//Verifying Password : login.aspx

```
hash= getHashedPasswordFromDB(username);
if(Bcrypt.Verify(password,hash))
{
    //login success page
}
```

A7 – Missing Function Level Access Control

Example Vulnerable Code

```
//menu.aspx
If(isAuthenticated())
{
    Response.Write("<a href="/Application/EditTitle.aspx">Edit Title</a>");
    …
    …
}
```

But in, "**/Application/EditTitle.aspx**" failed to do authorization Check

```
EditTitleAction(){
    //Sql query to update
}
```

Example Fix

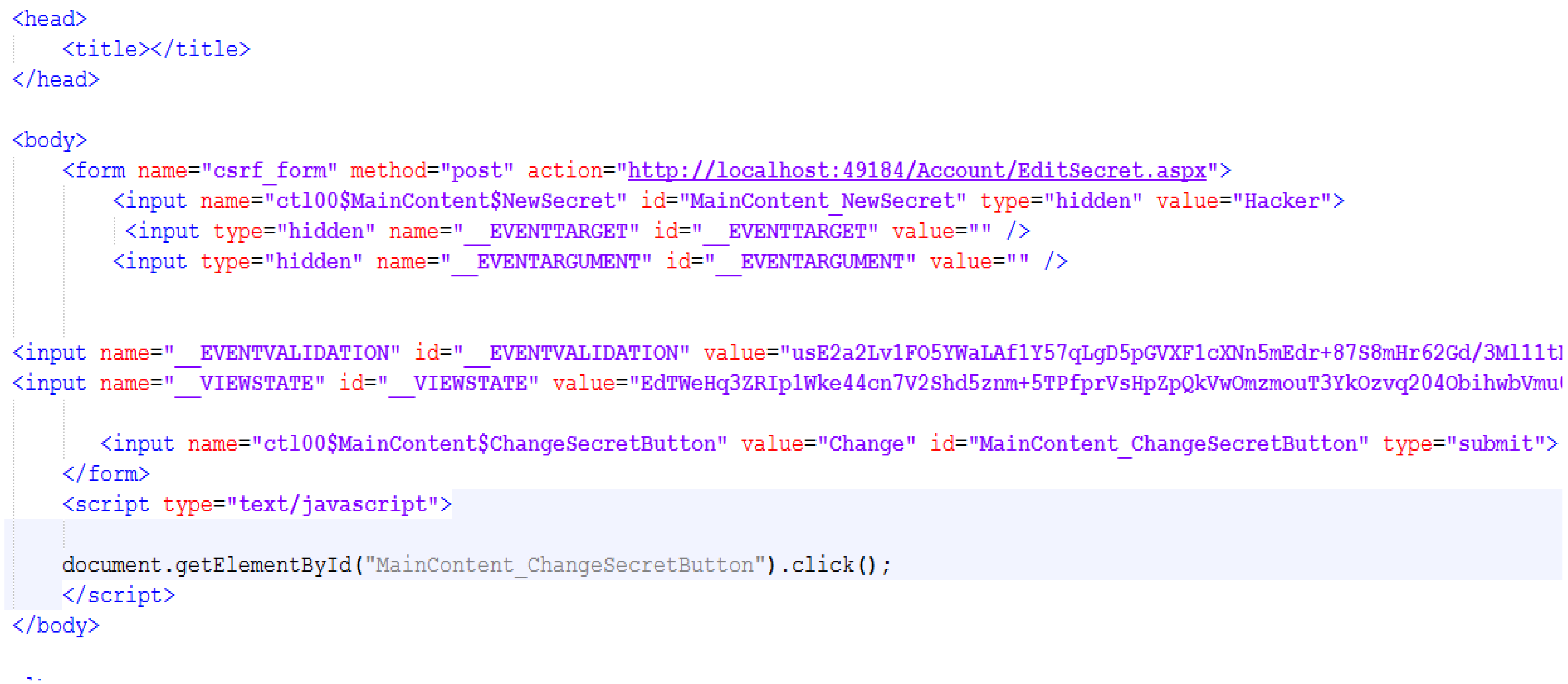
```
//menu.aspx
If(isAuthenticated())
{
    Response.Write("<a href="/Application/EditTitle.aspx">Edit Title</a>");
    …
    …
}
```

Check Authorization in every pages:

```
EditTitleAction()
{
      If(isAuthenticated())
      {   //Sql query to update
      }
}
```

A8 - CSRF

CSRF Attack POC



```
<head>
    <title></title>
</head>

<body>
    <form name="csrf_form" method="post" action="http://localhost:49184/Account/EditSecret.aspx">
        <input name="ctl00$MainContent$NewSecret" id="MainContent_NewSecret" type="hidden" value="Hacker">
         <input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="" />
        <input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="" />


<input name="__EVENTVALIDATION" id="__EVENTVALIDATION" value="usE2a2Lv1FO5YWaLAf1Y57qLgD5pGVXF1cXNn5mEdr+87S8mHr62Gd/3M111t
<input name="__VIEWSTATE" id="__VIEWSTATE" value="EdTWeHq3ZRIp1Wke44cn7V2Shd5znm+5TPfprVsHpZpQkVwOmzmouT3YkOzvq204ObihwbVmu

       <input name="ctl00$MainContent$ChangeSecretButton" value="Change" id="MainContent_ChangeSecretButton" type="submit">
    </form>
    <script type="text/javascript">

    document.getElementById("MainContent_ChangeSecretButton").click();
    </script>
</body>
```

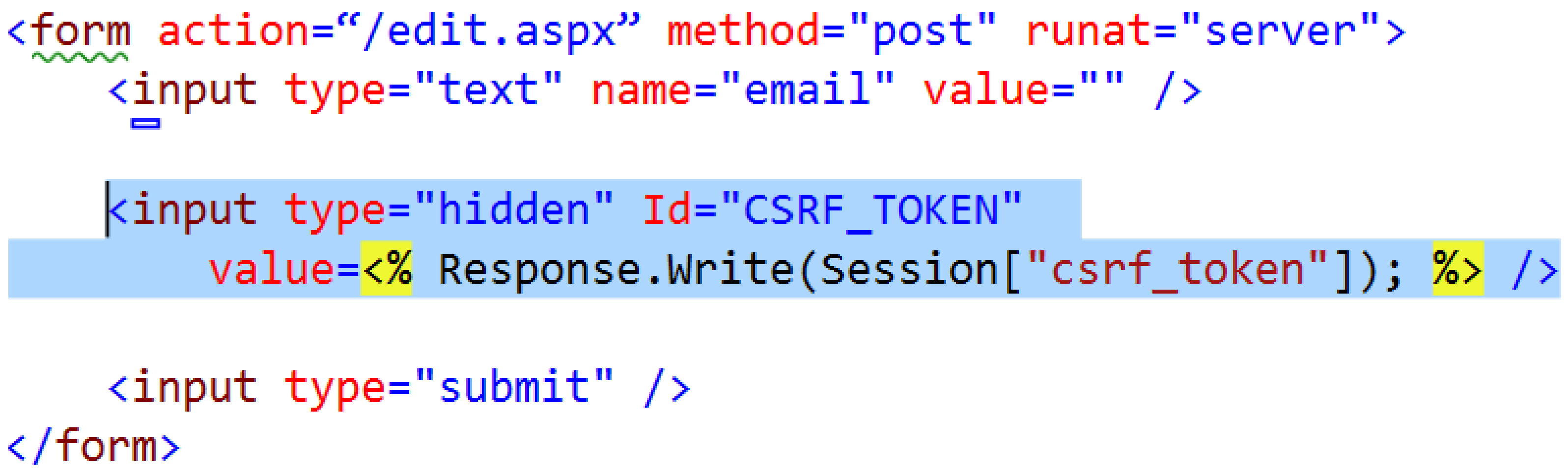
Example 1- Implementation of CSRF Token

Login.aspx – Generate Token:

```
If(validUser())
{
    Session[“csrf_token”] = SHA256(GetRandomNumber());
    ..
    …
}
```

Example 1- Implementation of CSRF Token

Edit.aspx – Including in Form:



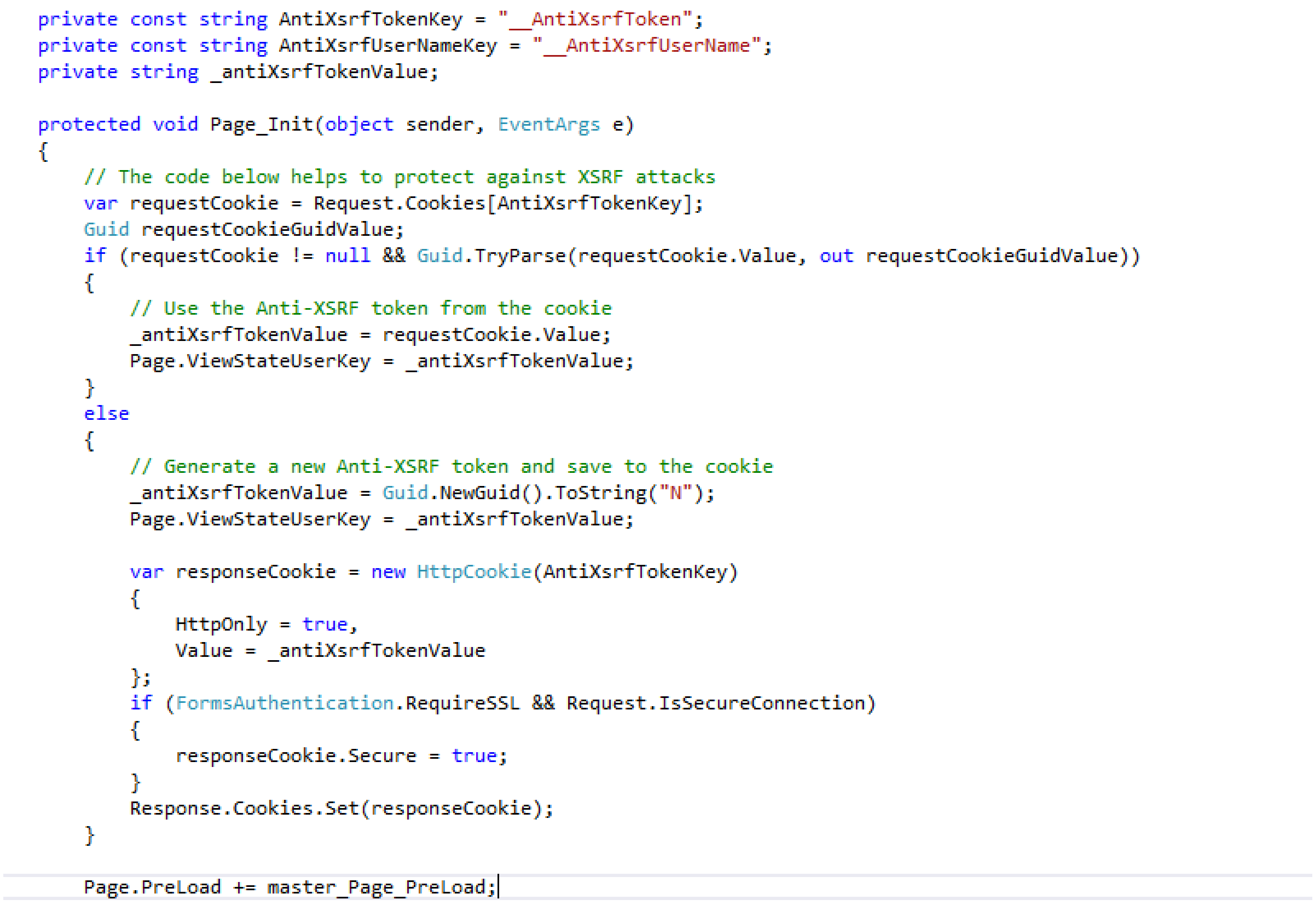
```
<form action="/edit.aspx" method="post" runat="server">
    <input type="text" name="email" value="" />

    <input type="hidden" Id="CSRF_TOKEN"
        value=<% Response.Write(Session["csrf_token"]); %> />

    <input type="submit" />
</form>
```

Example 1- Implementation of CSRF Token

Edit.aspx – Verifying before doing action:

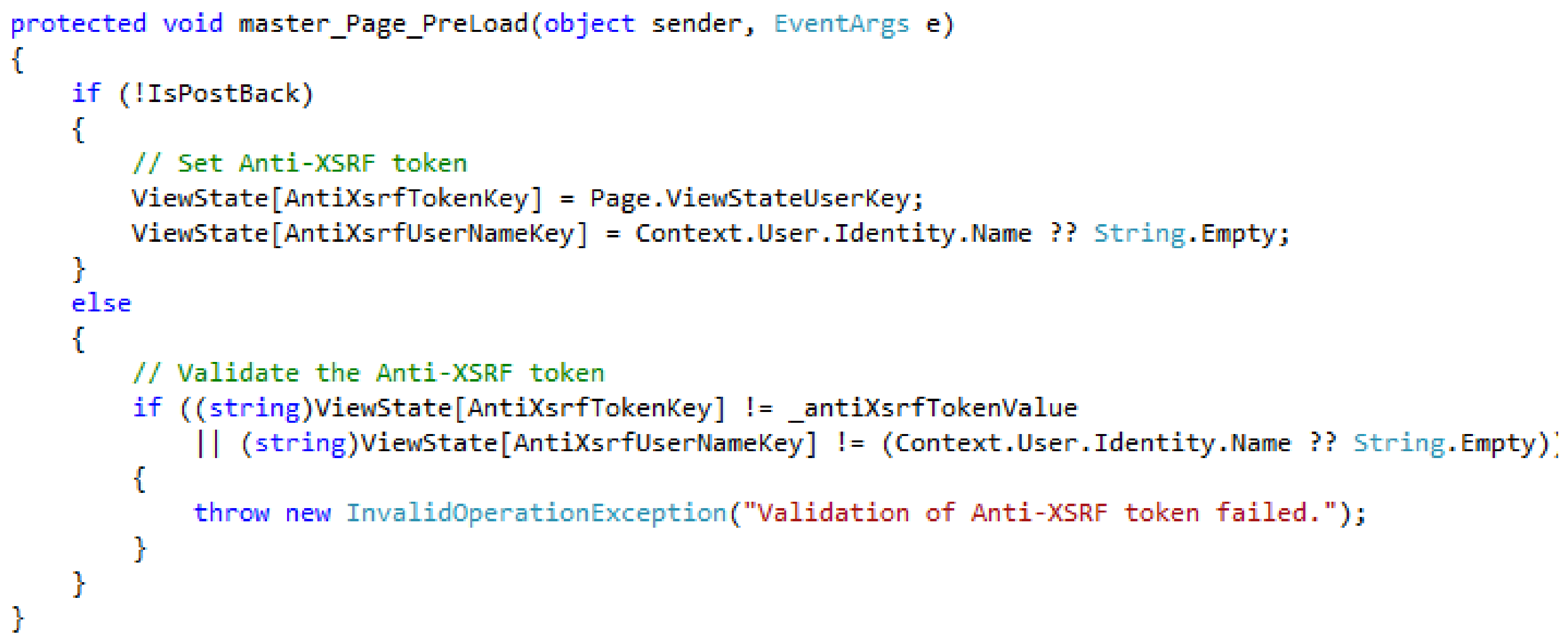
```
If(CSRF_TOKEN.Text == Session[“csrf_token”])
{
    //Do the action
}
else
{
    //deny the request & display Incorrect CSRF token
}
```

CSRF Token Implementation - Example 2 (continued..)



```csharp
private const string AntiXsrfTokenKey = "__AntiXsrfToken";
private const string AntiXsrfUserNameKey = "__AntiXsrfUserName";
private string _antiXsrfTokenValue;

protected void Page_Init(object sender, EventArgs e)
{
    // The code below helps to protect against XSRF attacks
    var requestCookie = Request.Cookies[AntiXsrfTokenKey];
    Guid requestCookieGuidValue;
    if (requestCookie != null && Guid.TryParse(requestCookie.Value, out requestCookieGuidValue))
    {
        // Use the Anti-XSRF token from the cookie
        _antiXsrfTokenValue = requestCookie.Value;
        Page.ViewStateUserKey = _antiXsrfTokenValue;
    }
    else
    {
        // Generate a new Anti-XSRF token and save to the cookie
        _antiXsrfTokenValue = Guid.NewGuid().ToString("N");
        Page.ViewStateUserKey = _antiXsrfTokenValue;

        var responseCookie = new HttpCookie(AntiXsrfTokenKey)
        {
            HttpOnly = true,
            Value = _antiXsrfTokenValue
        };
        if (FormsAuthentication.RequireSSL && Request.IsSecureConnection)
        {
            responseCookie.Secure = true;
        }
        Response.Cookies.Set(responseCookie);
    }

    Page.PreLoad += master_Page_PreLoad;
```

CSRF Token Implementation - Example 2

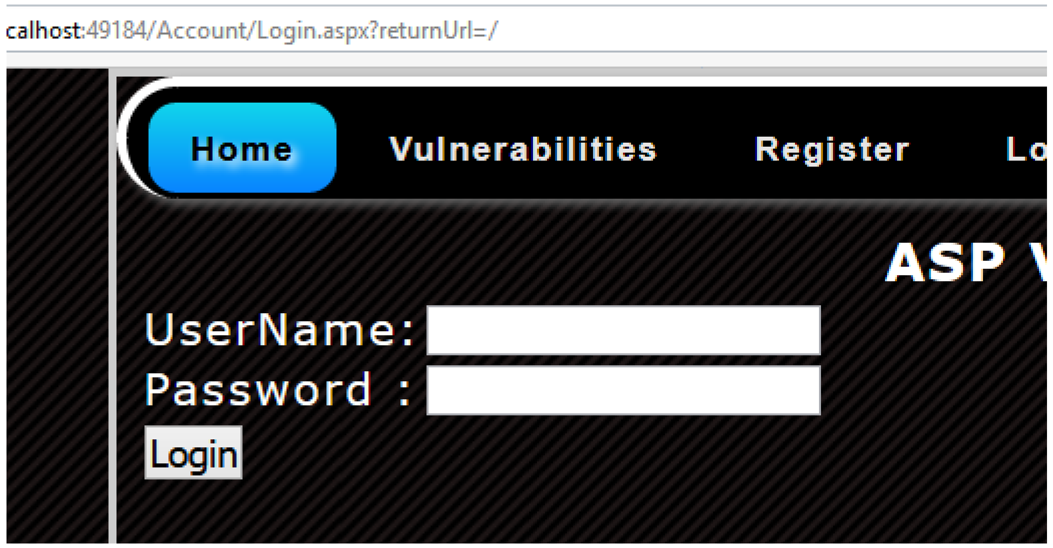


```
protected void master_Page_PreLoad(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        // Set Anti-XSRF token
        ViewState[AntiXsrfTokenKey] = Page.ViewStateUserKey;
        ViewState[AntiXsrfUserNameKey] = Context.User.Identity.Name ?? String.Empty;
    }
    else
    {
        // Validate the Anti-XSRF token
        if ((string)ViewState[AntiXsrfTokenKey] != _antiXsrfTokenValue
            || (string)ViewState[AntiXsrfUserNameKey] != (Context.User.Identity.Name ?? String.Empty))
        {
            throw new InvalidOperationException("Validation of Anti-XSRF token failed.");
        }
    }
}
```

Reference:
https://www.owasp.org/index.php/.NET_Security_Cheat_Sheet
https://www.divergent-thought.com/en/blog/cross-site-request-forgery

A10. Unvalidated Redirects and Forwards

Unvalidated Redirect:


calhost:49184/Account/Login.aspx?returnUrl=/

```
public void RedirectionAfterLogin()
{
    if (!String.IsNullOrEmpty(Request.QueryString["returnUrl"]))
    {
        Response.Redirect(Request.QueryString["returnUrl"]);
    }
    else
    {
        Response.Redirect("~/");
    }
}
```

Example Prevention

```
public void RedirectionAfterLogin_Fixed()
{
    if (!String.IsNullOrEmpty(Request.QueryString["returnUrl"]) && IsLocalUrl(Request.QueryString["returnUrl"]))
    {
        Response.Redirect(Request.QueryString["returnUrl"]);
    }
    else
    {
        Response.Redirect("~/");
    }
}

private bool IsLocalUrl(string url)
{
    /**
     * Validating URL & allowing only local redirection
     */

    // From: https://docs.microsoft.com/en-us/aspnet/mvc/overview/security/preventing-open-redirection-attacks

    if (string.IsNullOrEmpty(url))
    {
        return false;
    }
    else
    {
        return ((url[0] == '/' && (url.Length == 1 ||
                (url[1] != '/' && url[1] != '\\'))) ||   // "/" or "/foo" but not "//" or "/\"
                (url.Length > 1 &&
                 url[0] == '~' && url[1] == '/'));   // "~/" or "~/foo"
    }
}
```