

# Lab 13: Web Authentication

## Details

---

Aim: To investigate the usage of OpenID and SAML for Web authentication.


## Activities

---

### L13.1 Go to:

<https://www.myopenid.com/>

and create your new OpenID account. Next find some Web sites to login with using your new account.

 Write down your OpenID account, and try to find a few sites which support OpenID and log into them.

### L13.2 Using the following code:

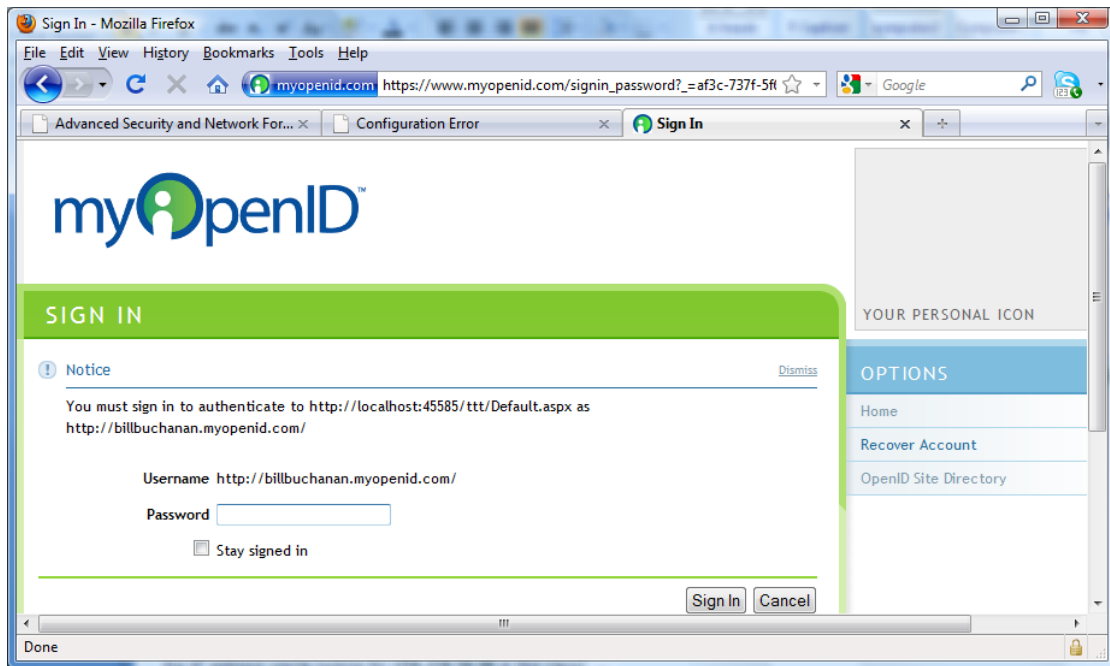
<http://buchananweb.co.uk/openid.rar>

Add a new page called success.aspx, and then add the highlighted code in the Default.aspx page:

```
protected void Page_Load(object sender, EventArgs e)
{
    OpenIdData data = OpenID.Authenticate();
    if (data.IsSuccess)
    {
        Response.Redirect("success.aspx");
    }
}

protected void Button1_Click(object sender, EventArgs e)
{
    bool success = OpenID.Login(txtOpenId.Text, "email,fullname",
    "country,language");
}
}
```

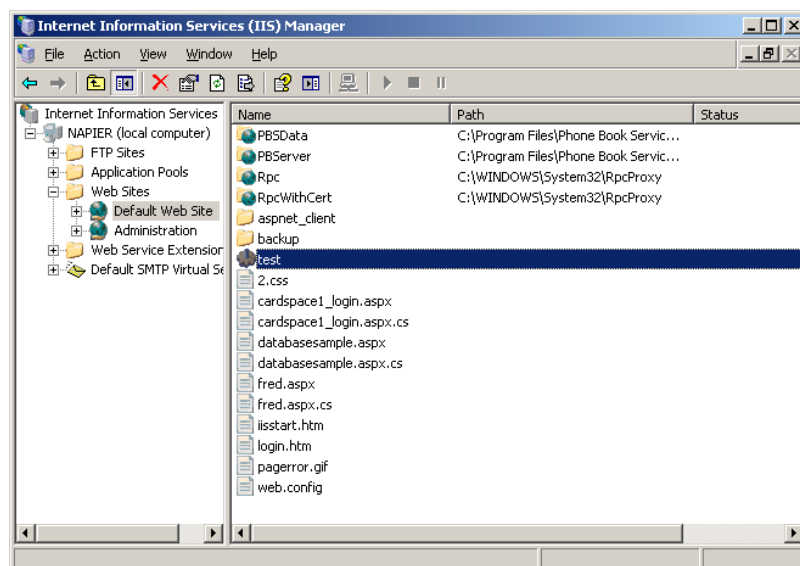
Prove that you can login with your OpenID identity.



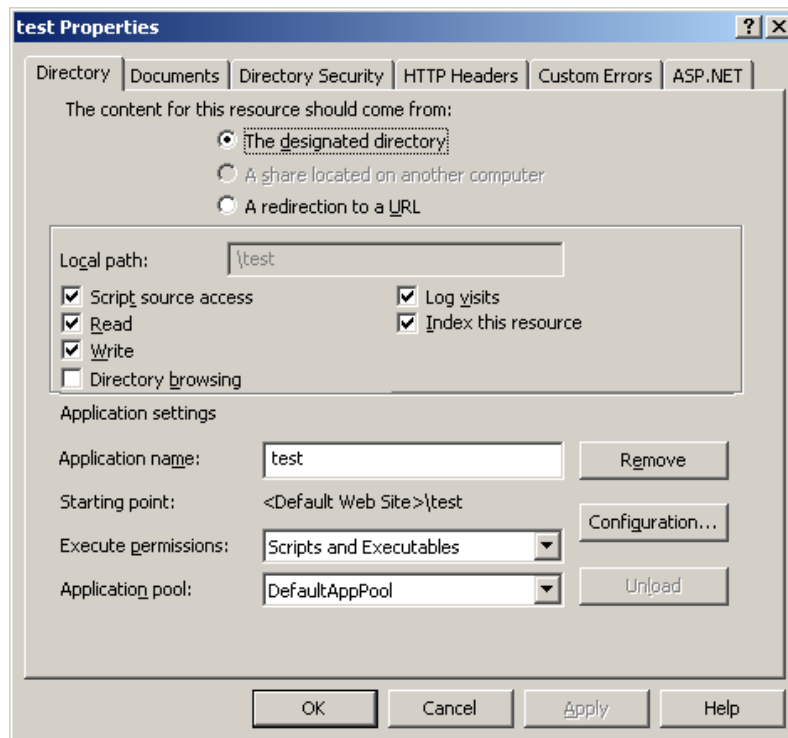
**L13.3** Run the WINDOWS2003 VM image and Download the following to the `c:\inetpub\wwwroot\test` folder:

<http://buchananweb.co.uk/wwwroot.zip>

**L13.4** Go into the IIS Manager and right click on the test folder (Figure L13.1), and set it up with an Application Name (Figure L13.2).



**Figure L13.1:** IIS Manager



**Figure L13.2:** test Properties

**L13.5** Next run Visual Studio 2008, and select Open Web site and navigate to c:\inetput\wwwroot\test.

**L13.6** Next select sample1.htm, and add the following code:

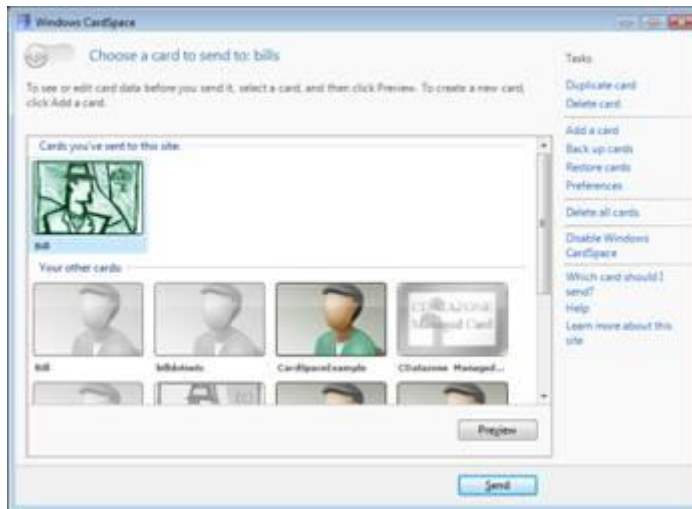
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>Sample 1</title>
</head>
<body>
<form id="form1" method="post" action="cardspace1_login.aspx">
<div>
<button type="submit">Click here to sign in with your Information Card</button>
<object type="application/x-informationcard" name="xmlToken">
<param name="tokenType" value="urn:oasis:names:tc:SAML:1.0:assertion" />
<param name="requiredClaims"
value="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier" />
</object>
</div>
</form>

</body>
</html>
```

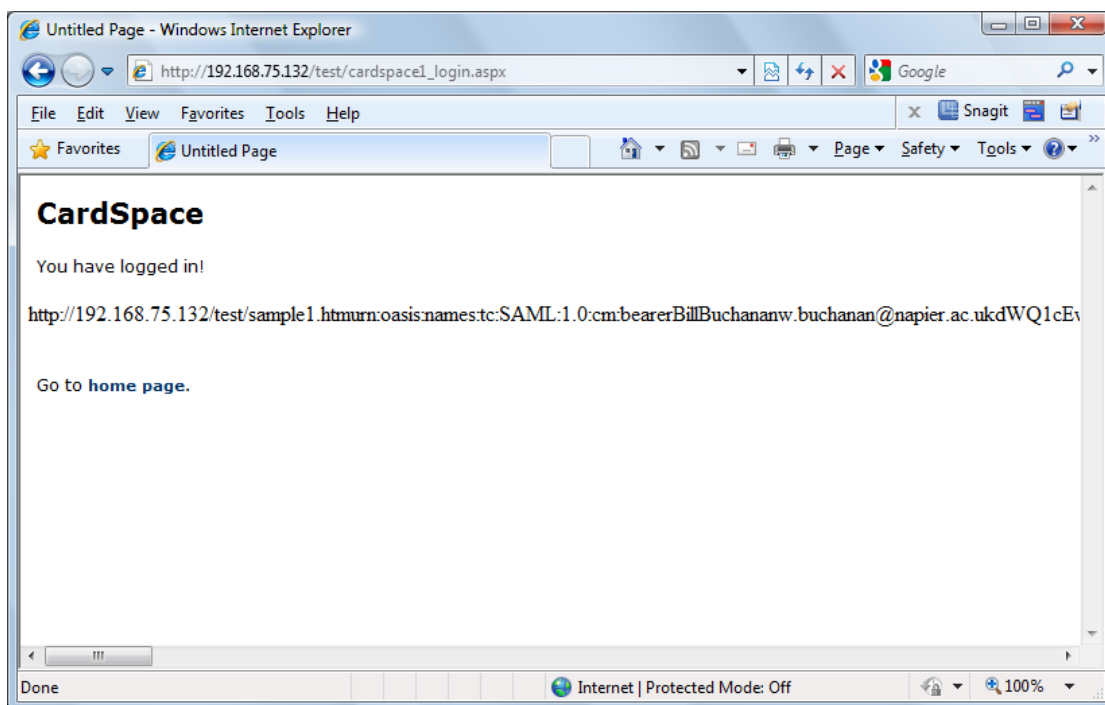
**L13.7** Next select cardspace1\_login.aspx.cs, and add the highlighted code:

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = Request.Params["xmlToken"];
}
```

**L13.8** Next load <https://localhost>, and select the first example (sample1.htm). Select your card (or create one), and login, such as:



**L13.9** Next login remotely from your desktop into the virtual image, such as with:



**L13.10** Next select sample2.htm, and add the following code:.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>Authenticate</title>
<object type="application/x-informationcard" name="_xmlToken">
<param name="tokenType" value="urn:oasis:names:tc:SAML:1.0:assertion" />
<param name="requiredClaims"
value="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier" />
```

```

</object>
<script language="javascript">
function GoGetIt()
{
var xmltkn=document.getElementById("_xmltoken");
var thetextarea = document.getElementById("xmltoken");
thetextarea.value = xmltkn.value ;
}
</script>
</head>
<body>
<form id="form1" method="post" action="cardspace2_login.aspx">
<div>
<button name="go" id="go" onclick="javascript:GoGetIt();">Click here to get the to-
ken.</button>
<button type="submit">Click here to send the card to the server</button>
<textarea cols=100 rows=20 id="xmltoken" name="xmlToken" ></textarea>
</div>
</form>
</body>
</html>

```

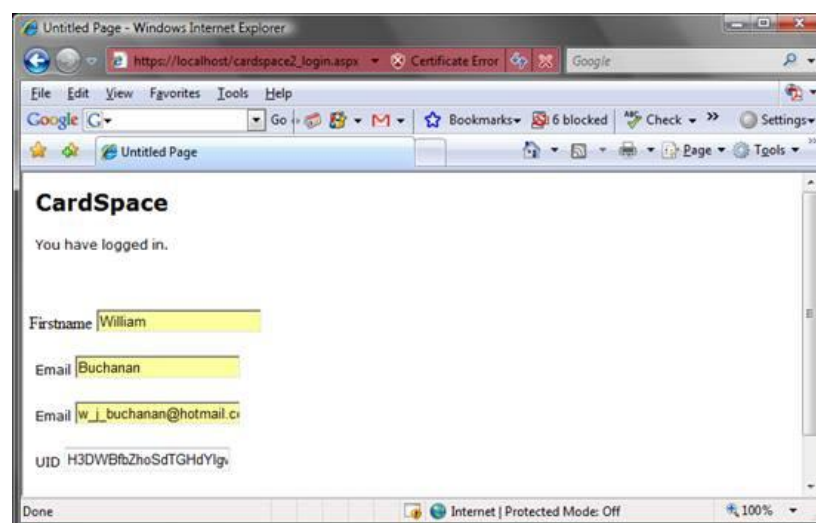
**L13.11** Next select cardspace2\_login.aspx.cs, and add the highlighted code:

```

protected void Page_Load(object sender, EventArgs e)
{
    string xmlToken;
    xmlToken = Request.Params["xmlToken"];
    if (xmlToken == null || xmlToken.Equals(""))
    {
        // ShowError("Token presented was null");
    }
    else
    {
        Token token = new Token(xmlToken);
        firstname.Text = token.Claims[ClaimTypes.GivenName];
        surname.Text = token.Claims[ClaimTypes.Surname];
        email.Text = token.Claims[ClaimTypes.Email];
        uid.Text = token.UniqueID;
    }
}

```

Next show that the Web site now displays the details from the card, such as:



**L13.12** Export the card you have created, and view its contents. Now import it into WINDOWS2003.

## Toolkit 7 (URL cache)

🔗 On-line demo:  
[http://buchananweb.co.uk/adv\\_security\\_and\\_network\\_forensics/toolkit07/toolkit07.htm](http://buchananweb.co.uk/adv_security_and_network_forensics/toolkit07/toolkit07.htm)

The objective of this series of labs is to build an integrated toolkit. Open up:

<http://buchananweb.co.uk/toolkit.zip>

and extract to a local folder. Next open up toolkit.sln, and double click on client.cs (Refer to <http://buchananweb.co.uk/dotnetclientserver.zip> for a completed version).

### 7.1 Add a new tab named [OS], and add another tab into this tab (see Figure 7.3).

Next add two DateTimePickers (dtStart and dtEnd), two buttons, and two datagridviews (dgURLCache and dgFileCache). Add the following code on the Show History button:

```
Showhistory();
```

And the method:

```
using UrlHistoryLibrary;

public void Showhistory()
{
    this.dgURLCache.Rows.Clear();
    this.dgFileCache.Rows.Clear();
    urlHistory = new UrlHistoryWrapperClass();
    enumerator = urlHistory.GetEnumerator();
    list = new ArrayList();

    GetHistoryItems();

    list.Reverse();

    if (textBoxFilter.Text != "")
    {
        enumerator.SetFilter(textBoxFilter.Text,
STATURLFLAGS.STATURLFLAG_ISTOPLEVEL);
    }
    foreach (STATURL u in list)
    {
        string[] url = new string[2];

        url[0] = Convert.ToString(u.LastVisited);
        url[1] = u.URL;
        STATURL u1 = (STATURL)list[0];

        if (u.LastVisited >= dtStart.Value && u.LastVisited <= dtEnd.Value)
        {
            u1 = (STATURL)list[list.Count - 1];

            if (url[1].StartsWith("http")) this.dgURLCache.Rows.Add(url);
            else if (url[1].StartsWith("file"))
this.dgFileCache.Rows.Add(url);
        }
    }

    GC.Collect();
}
```

**7.2** Test that the program can view the URL history. Next add the following code to the Clear URL History button:

```

        DialogResult rtn=MessageBox.Show("Are you sure you want to delete all your
URL history?", "URL History", MessageBoxButtons.YesNo);
        if (rtn == DialogResult.Yes)
            urlHistory.ClearHistory();
    
```

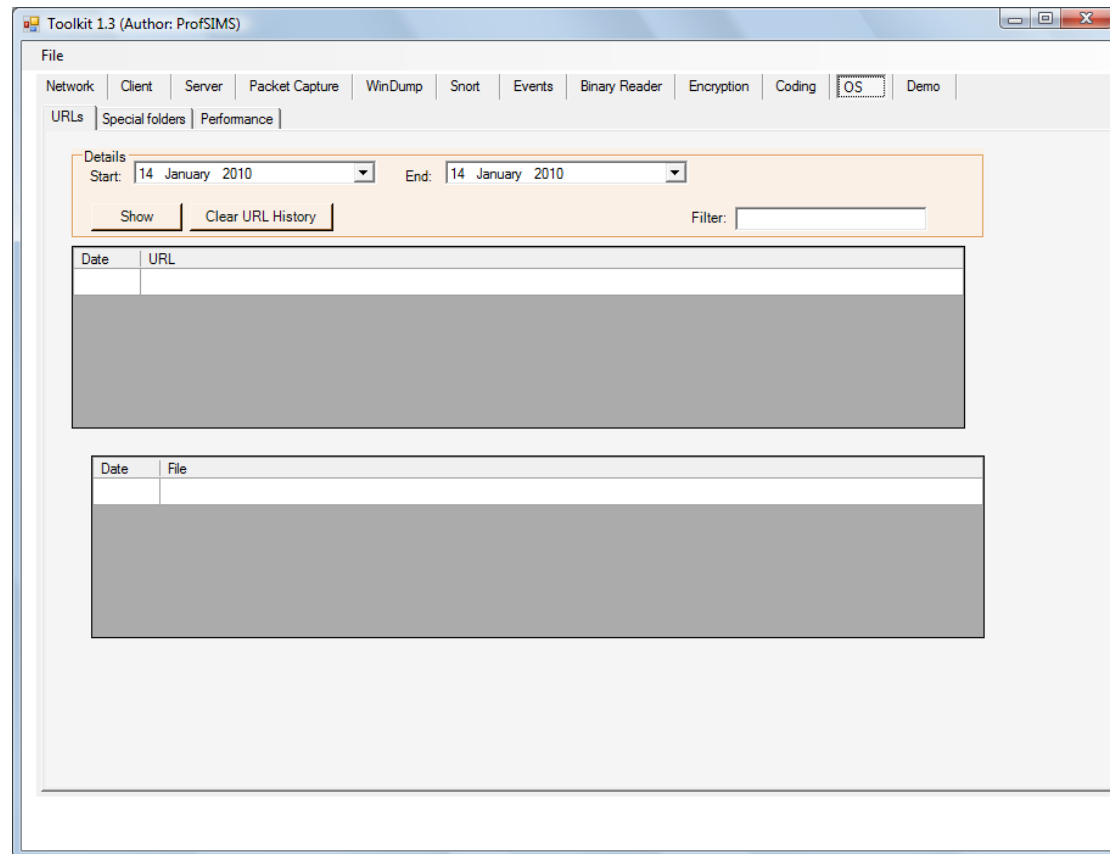


Figure 7.3

**7.3** Test the program for its operation.

**7.4** If you have time, investigate the “Special Folders” tab (see Figure 7.3), such as with the following code:

```

        DirectoryInfo d = new
DirectoryInfo(System.Environment.GetFolderPath(Environment.SpecialFolder.Recent));
        ShowFiles(dgFilesRecent, d.FullName);

    public void ShowFiles(DataGridView dg, string folder)
    {
        try
        {
            dg.Rows.Clear();
            string[] files = Directory.GetFiles(folder);
            CreateMessageForStatus(tbFiles, folder);
            foreach (string s in files)
            {
                string filename = s;
                FileInfo f = new FileInfo(filename);
            }
        }
        catch { }
    }
    
```

```
        string[] s1 = new string[2];  
        s1[0] = Convert.ToString(f.LastAccessTime);  
        s1[1] = s;  
        CreateMessageForStatusAppend(dg, s1);  
    }  
}  
catch (Exception ex)  
{  
}  
}
```

This code allows the user to view the “Recent” special folder. If you get this code to work, try and view the other “Special folders”.