

MOSS 2007 and Code Access Security



Figure 4.24: *Code Access Security*

Download Sample Custom Policy File:

<http://www.codeplex.com/wsuploadservice/Project/FileDownload.aspx?DownloadId=5518>

Have you ever written a web part or a web service? If yes then you must have dealt with a security problem. Writing a web part or web service may not be a big issue but deploying them is certainly a headache. You start getting permission errors as soon as you deploy your code on the server. There are three ways to assign execution permissions to your code:

1. Increase the trust level for the entire virtual server
2. Create a custom policy file for your assemblies
3. Install your assemblies in the GAC

Safest method to install an assembly is to create a custom policy file for the assembly. Following article on MSDN contains complete details on code access security:

Microsoft Windows SharePoint Services and Code Access Security
(<http://msdn2.microsoft.com/en-us/library/ms916855.aspx>)

Written in July 2003, this is one of the most comprehensive articles written on "SharePoint and Code Access Security".

For security reasons, the assembly must be installed in the **bin** directory of the application instead of GAC but installing it in the **bin** directory requires you to assign execution permissions to the assembly. One way is to increase the trust level of the entire virtual server.

This is easy to implement but this option is least secure as it affects all assemblies used by that virtual server. Second way is to create a custom policy file and this is the recommended approach. This option is most secure but difficult to implement. In this section, we will create a custom policy file for an assembly (We will use WSUploadService discussed earlier in this book) written for MOSS 2007.

Creating a Custom Policy File

1. Go to the following location on the server:

LocalDrive:\Program Files\Common Files\Microsoft Shared\web server extensions\12\CONFIG

2. Make a copy of `wss_minimaltrust.config` and rename it `wss_customtrust.config`.
3. Open `wss_customtrust.config` file using any text editor.
4. Under the `<SecurityClasses>` element, add a reference to the **SharePointPermissions** class as follows:

```
<SecurityClass Name="SharePointPermission"
Description="Microsoft.SharePoint.Security.SharePointPermission,
Microsoft.SharePoint.Security, Version=12.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c." />
```

Listing 4.1: Change in `<SecurityClass>`

5. Search for the `<PermissionSet>` tag where the **name** attribute equals **ASP.NET**. If you couldn't find that `<PermissionSet>` tag, locate the one that has **SPRestricted** in the **name** attribute.
6. Copy the entire tag and all of its children, and paste a copy of it immediately below the one you copied.
7. Change the name of the **PermissionSet** element from **ASP.NET** (or **SPRestricted**) to **CustomTrust**.

Before:

```
<PermissionSet  
class="NamedPermissionSet"  
version="1"  
Name="SPRestricted">
```

Listing 4.2: *Permissionset***After:**

```
<PermissionSet  
class="NamedPermissionSet"  
version="1"  
Name="CustomTrust">
```

Listing 4.3: *Permissionset after modification*

8. Add the following `<IPermission>` node to the `<PermissionSet>` element where the name attribute equals **CustomTrust**:

```
<IPermission class="SharePointPermission"  
version="1"  
ObjectModel="True" />
```

Listing 4.4: *Add <IPermission>*

Therefore, the resulting customized <PermissionSet> will look as follows:

```
<PermissionSet
class="NamedPermissionSet"
version="1"
Name="CustomTrust">

<IPermission
class="AspNetHostingPermission"
version="1" Level="Minimal"
/>

<IPermission
class="SecurityPermission"
version="1" Flags="Execution"
/> <IPermission class="WebPartPermission"
version="1"
Connections="True"
/>

<IPermission class="SharePointPermission"
version="1"
ObjectModel="True" />
</PermissionSet>
```

Listing 4.5: *Customized permissionset*

9. Once you define the customized element, you must create a code group to specify when the CLR should apply the permission set. Locate <CodeGroup> tag where the **class** attribute equals **FirstMatchCodeGroup** and copy following **CodeGroup** immediately below it:

```

<CodeGroup class="UnionCodeGroup"
version="1"
PermissionSetName="CustomTrust">
<IMembershipCondition class="UrlMembershipCondition"
version="1"
Url="$AppDirUrl$/bin/*" />
</CodeGroup>

```

Listing 4.6: *<CodeGroup>*

The membership condition for this new code group is based on URL membership and the URL points to the **bin** directory. The permissions will be applied to all the assemblies in the **bin** directory of the current application. You can also use strong name membership but then the permissions will be applied only to one assembly. For example, if I have written a web service and I wanted to assign permissions to my assembly only, I would use strong name membership. Copy following code immediately below the `<CodeGroup>` tag where the **class** attribute equals **FirstMatchCodeGroup**, if you want to use strong name membership:

```

<CodeGroup class="UnionCodeGroup"
version="1"
PermissionSetName="CustomTrust">
<IMembershipCondition class="StrongNameMembershipCondition"
version="1"
PublicKeyBlob="0x00240000048000009400000006020000002400005253413100040000010
001004"
Name="UploadService" />
</CodeGroup>

```

Listing 4.7: *<CodeGroup> modification*

Replace **PublicKeyBlob** value with your own value and change the name of the assembly in the **Name** attribute. **Name** attribute contains the name of the assembly. To retrieve the public key blob for an assembly, use the `secutil.exe` tool. Please note that **publickeyblob** is different from **publickeytoken**. `Secutil.exe` is located in the following folder:

LocalDrive:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\Bin

To retrieve the public key blob for your assembly, either copy the `secutil.exe` tool to the folder that contains your assembly else provide exact path to the assembly in the command, and run the tool as follows:

```
secutil.exe -hex -s UploadService.dll > blob.txt
```

Listing 4.8: *secutil.exe*

UploadService.dll is the name of the assembly. This command will create a text file named blob.txt. Open blob.txt and copy the public key and paste it in the **publickeyblob** attribute.

10. Save and close the file. The policy file is ready to use.

11. Open the web.config file for the virtual server where you have deployed your component and add the following <trustlevel> tag to the **SecurityPolicy** element:

```
<trustLevel name="WSS_Custom" policyFile="LocalDrive:\Program Files\Common  
Files\Microsoft Shared\Web Server Extensions\12\config\wss_customtrust.config" />
```

Listing 4.9: <TrustLevel>

Virtual Directories for web applications are located in the following folder:

LocalDrive:\Inetpub\wwwroot\wss\VirtualDirectories

Suppose I want to deploy my web service in the web application configured at port 17316. The URL of that application would be `http://localhost:17316/` and its virtual directory will be:

LocalDrive:\Inetpub\wwwroot\wss\VirtualDirectories\17315

Create a **bin** folder in this path and copy your assembly to the bin folder. The web.config for this virtual server will be located in the following folder:

LocalDrive:\Inetpub\wwwroot\wss\VirtualDirectories\17315

In the web.config file, change the <trust> tag so that it refers to the newly defined trust level.

```
<trust level="WSS_Custom" originUrl="" />
```

Listing 4.10: *WSS_Custom trust level*

12. Save and close the web.config file.

13. Restart IIS to apply the custom policy to the specified virtual server.

Download Sample Custom Policy File:

<http://www.codeplex.com/wsuploadservice/Project/FileDownload.aspx?DownloadId=5518>

Summary

This chapter was the amalgamation of different features and techniques. The chapter started with the topic that showed you how to make Excel services work in SharePoint. It talks about some important settings that are usually overlooked by the users and the services fail to work as a result. This chapter teaches you making SharePoint surveys accessible to the anonymous users. This chapter also introduced you to two very useful templates, GroupBoard Workspace and Sales Account Manager. You saw different features available out of the box in these two extremely useful templates. This chapter also teaches you a technique to hide library items after a certain amount of time. The last section is about the code access security. This is another area that is overlooked by the users but is of great importance, especially to the developers who struggle to apply code access security to their components. Developing components for SharePoint is only half work, the remaining work is to use them with proper security settings in a proper security context.