Membership and Role Provider

.NET Framework 4.5 Este tópico ainda não foi avaliado como

The Membership and Role Provider sample demonstrates how a service can use the ASP.NET membership and role providers to authenticate and authorize clients.

In this sample, the client is a console application (.exe) and the service is hosted by Internet Information Services (IIS).

Note:
The setup procedure and build instructions for this sample are located at the end of this topic.

The sample demonstrates how:

- A client can authenticate by using the username-password combination.
- The server can validate the client credentials against the ASP.NET membership provider.
- The server can be authenticated by using the server's X.509 certificate.
- The server can map the authenticated client to a role by using the ASP.NET role provider.
- The server can use the `PrincipalPermissionAttribute` to control access to certain methods that are exposed by the service.

The membership and role providers are configured to use a store backed by SQL Server. A connection string and various options are specified in the service configuration file. The membership provider is given the name `SqlMembershipProvider` while the role provider is given the name `SqlRoleProvider`.

```
<!-- Set the connection string for SQL Server -->
<connectionStrings>
  <add name="SqlConn"
       connectionString="Data Source=localhost;Integrated Security=SSPI;Initial Cat
</connectionStrings>

<system.web>
  <!-- Configure the Sql Membership Provider -->
  <membership defaultProvider="SqlMembershipProvider" userIsOnlineTimeWindow="15">
    <providers>
      <clear />
      <add
        name="SqlMembershipProvider"
        type="System.Web.Security.SqlMembershipProvider"
        connectionStringName="SqlConn"
        applicationName="MembershipAndRoleProviderSample"
        enablePasswordRetrieval="false"
        enablePasswordReset="false"
```

```
        requiresQuestionAndAnswer="false"
        requiresUniqueEmail="true"
        passwordFormat="Hashed" />
    </providers>
  </membership>

  <!-- Configure the Sql Role Provider -->
  <roleManager enabled ="true"
               defaultProvider ="SqlRoleProvider" >
    <providers>
      <add name ="SqlRoleProvider"
           type="System.Web.Security.SqlRoleProvider"
           connectionStringName="SqlConn"
           applicationName="MembershipAndRoleProviderSample"/>
    </providers>
  </roleManager>
</system.web>
```

The service exposes a single endpoint for communicating with the service, which is defined by using the Web.config configuration file. The endpoint consists of an address, a binding, and a contract. The binding is configured with a standard **wsHttpBinding**, which defaults to using Windows authentication. This sample sets the standard **wsHttpBinding** to use username authentication. The behavior specifies that the server certificate is to be used for service authentication. The server certificate must contain the same value for the **SubjectName** as the **findValue** attribute in the serviceCertificate element of serviceCredentials configuration element. In addition the behavior specifies that authentication of username-password pairs is performed by the ASP.NET membership provider and role mapping is performed by the ASP.NET role provider by specifying the names defined for the two providers.

```
<system.serviceModel>

  <protocolMapping>
    <add scheme="http" binding="wsHttpBinding" />
  </protocolMapping>


  <bindings>
    <wsHttpBinding>
      <!-- Set up a binding that uses Username as the client credential type -->
      <binding>
        <security mode ="Message">
          <message clientCredentialType ="UserName"/>
        </security>
      </binding>
    </wsHttpBinding>
  </bindings>

  <behaviors>
    <serviceBehaviors>
      <behavior>
        <!-- Configure role based authorization to use the Role Provider -->
        <serviceAuthorization principalPermissionMode ="UseAspNetRoles"
                              roleProviderName ="SqlRoleProvider" />
        <serviceCredentials>
```

```
            <!-- Configure user name authentication to use the Membership Provider --
            <userNameAuthentication userNamePasswordValidationMode ="MembershipProvid
                                    membershipProviderName ="SqlMembershipProvider"/>
            <!-- Configure the service certificate -->
            <serviceCertificate storeLocation ="LocalMachine"
                                storeName ="My"
                                x509FindType ="FindBySubjectName"
                                findValue ="localhost" />
          </serviceCredentials>
          <!--For debugging purposes set the includeExceptionDetailInFaults attribute
          <serviceDebug includeExceptionDetailInFaults="false" />
          <serviceMetadata httpGetEnabled="true"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
```

When you run the sample, the client calls the various service operations under three different user accounts: Alice, Bob, and Charlie. The operation requests and responses are displayed in the client console window. All four calls made as user "Alice" should succeed. User "Bob" should get an access denied error when trying to call the Divide method. User "Charlie" should get an access denied error when trying to call the Multiply method. Press ENTER in the client window to shut down the client.

To set up, build, and run the sample

1. To build the C# or Visual Basic .NET edition of the solution, follow the instructions in Running the Windows Communication Foundation Samples.
2. Ensure that you have configured the ASP.NET Application Services Database.

> **Note:**
>
> If you are running SQL Server Express Edition, your server name is .\SQLEXPRESS. This server should be used when configuring the ASP.NET Application Services Database as well as in the Web.config connection string.

> **Note:**
>
> The ASP.NET worker process account must have permissions on the database that is created in this step. Use the sqlcmd utility or SQL Server Management Studio to do this.

3. To run the sample in a single- or cross-computer configuration, use the following instructions.

To run the sample on the same computer

1. Make sure that the path includes the folder where Makecert.exe is located.
2. Run Setup.bat from the sample install folder in a Visual Studio command prompt run with administrator privileges. This installs the service certificates required for running the sample.
3. Launch Client.exe from \client\bin. Client activity is displayed on the client console application.
4. If the client and service are not able to communicate, see Troubleshooting Tips.

To run the sample across computers

1. Create a directory on the service computer. Create a virtual application named servicemodelsamples for this directory by using the Internet Information Services (IIS) management tool.
2. Copy the service program files from \inetpub\wwwroot\servicemodelsamples to the virtual directory on the service computer. Ensure that you copy the files in the \bin subdirectory. Also copy the Setup.bat, GetComputerName.vbs, and Cleanup.bat files to the service computer.
3. Create a directory on the client computer for the client binaries.
4. Copy the client program files to the client directory on the client computer. Also copy the Setup.bat, Cleanup.bat, and ImportServiceCert.bat files to the client.
5. On the server, open a Visual Studio command prompt with administrative privileges and run `setup.bat service`. Running `setup.bat` with the `service` argument creates a service certificate with the fully-qualified domain name of the computer and exports the service certificate to a file named Service.cer.
6. Edit Web.config to reflect the new certificate name (in the **findValue** attribute in the serviceCertificate element of serviceCredentials), which is the same as the fully-qualified domain name of the computer.
7. Copy the Service.cer file from the service directory to the client directory on the client computer.
8. In the Client.exe.config file on the client computer, change the address value of the endpoint to match the new address of your service.
9. On the client, open a Visual Studio command prompt with administrative privileges and run ImportServiceCert.bat. This imports the service certificate from the Service.cer file into the CurrentUser - TrustedPeople store.
10. On the client computer, launch Client.exe from a command prompt. If the client and service are not able to communicate, see Troubleshooting Tips.

To clean up after the sample

- Run Cleanup.bat in the samples folder after you have finished running the sample.

Note:
This script does not remove service certificates on a client when running this sample across computers. If you have run Windows Communication Foundation (WCF) samples that use certificates across computers, be sure to clear the service certificates that have been installed in the CurrentUser - TrustedPeople store. To do this, use the following command: `certmgr -del -r CurrentUser -s TrustedPeople -c -n <Fully Qualified Server Machine Name>` For example: `certmgr -del -r CurrentUser -s TrustedPeople -c -n server1.contoso.com`.

The Setup Batch File

The Setup.bat batch file included with this sample allows you to configure the server with relevant certificates to run a self-hosted application that requires server certificate-based security. This batch file must be modified to work across computers or to work in a non-hosted case.

The following provides a brief overview of the different sections of the batch files so that they can be modified to run in the appropriate configuration.

- Creating the server certificate.

  The following lines from the Setup.bat batch file create the server certificate to be used. The %SERVER_NAME% variable specifies the server name. Change this variable to specify your own server name. This batch file defaults it to localhost.

  The certificate is stored in My (Personal) store under the LocalMachine store location.

```
echo ************
echo Server cert setup starting
echo %SERVER_NAME%
echo ************
echo making server cert
echo ************
makecert.exe -sr LocalMachine -ss MY -a sha1 -n CN=%SERVER_NAME% -sky exchang
```

- Installing the server certificate into the client's trusted certificate store.

  The following lines in the Setup.bat batch file copy the server certificate into the client trusted people store. This step is required because certificates generated by Makecert.exe are not implicitly trusted by the client system. If you already have a certificate that is rooted in a client trusted root certificate —for example, a Microsoft-issued certificate—this step of populating the client certificate store with the server certificate is not required.

```
certmgr.exe -add -r LocalMachine -s My -c -n %SERVER_NAME% -r CurrentUser -s
```

Contribuições da comunidade