# [WCF的用户名密码认证](http://www.cnblogs.com/xiaozhuang/archive/2008/04/30/1177399.html)

以前我们用WebService做分布式系统的时候，认证是个麻烦的问题，通常的做法是继承一个SoapHeader，把用户名和密码放到里面，每调用一个方法都要把用户名和密码传递给服务器端来验证 ，效率相当低，代码编写相当的麻烦，而且还不安全！

WCF支持多种认证技术，例如Windowns认证、X509证书、Issued Tokens、用户名密码认证等，在跨Windows域分布的系统中，用户名密码认证还是比较常用的，要实现用户名密码认证，就必须需要X509证书，为什么呢？因为我们需要X509证书这种非对称密钥技术来实现WCF在Message传递过程中的加密和解密，要不然用户名和密码就得在网络上明文传递！详细说明就是客户端把用户名和密码用公钥加密后传递给服务器端，服务器端再用自己的私钥来解密，然后传递给相应的验证程序来实现身份验证。

当然，做个测试程序就没有必要去申请一个X509数字签名证书了，微软提供了一个makecert.exe的命令专门用来生成测试使用的X509证书的，那我们就来建立一个测试用的证书，在cmd下输入以下命令：

makecert.exe -sr LocalMachine -ss My -a sha1 -n CN=MyServerCert -sky exchange –pe

这个命令的意思就是创建一个测试的X509证书，这个证书放在存储位置为'Localmachine'的'My'这个文件夹下，证书主题名字叫'MyServerCert'，需要更多关于makecert命令的信息请参考MSDN。

证书建立好了，我们就可以编写代码了，在VS2008下建立一个解决方案并在上面建立两个Web项目，一个是'Asp.net Web 应用程序'（客户端），一个是'WCF服务应用程序'（服务器端），我们先来编写服务器端代码，首先我们要编写自己的用户名密码认证逻辑，先要在WCF项目上添加引用'System.IdentityModel'然后我们建立一个新的类文件并继承自'System.IdentityModel.Selectors.UserNamePasswordValidator'，然后我们重写里面的'Validate'方法来实现用户名密码认证逻辑。代码如下；

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IdentityModel.Selectors;

using System.IdentityModel.Tokens;

namespace ServerWcfService.CustomValidators

{

public class MyCustomValidator : UserNamePasswordValidator

{

/// <summary>

/// Validates the user name and password combination.

/// </summary>

/// <param name="userName">The user name.</param>

/// <param name="password">The password.</param>

public override void Validate(string userName, string password)

{

// validate arguments

if (string.IsNullOrEmpty(userName))

throw new ArgumentNullException("userName");

if (string.IsNullOrEmpty(password))

throw new ArgumentNullException("password");

// check if the user is not xiaozhuang

if (userName != "xiaozhuang" || password != "123456")

throw new SecurityTokenException("用户名或者密码错误！");

}

}

}

上面只是一个简单的验证，实际应用中用户名和密码一般都保存在数据库中，如果验证不通过就抛出一个'SecurityTokenException'类型的异常；下一步我们需要配置一下服务端的webConfig文件，我的WebConfig文件Servicemodel配置节如下：

<system.serviceModel>

**<bindings>**

**<wsHttpBinding>**

**<binding** **name="mySecureBinding">**

**<security** **mode="Message">**

**<message** **clientCredentialType="UserName"/>**

**</security>**

**</binding>**

**</wsHttpBinding>**

**</bindings>**

        <services>

            <service behaviorConfiguration="ServerWcfService.Services.MySimpleServiceBehavior"name="ServerWcfService.Services.MySimpleService">

                <endpoint address="" binding="wsHttpBinding"contract="ServerWcfService.ServiceContracts.IMySimpleService" **bindingConfiguration="mySecureBinding"**>

                    <identity>

**<dns** **value="MyServerCert"/>**

                    </identity>

                </endpoint>

                <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>

            </service>

        </services>

        <behaviors>

            <serviceBehaviors>

                <behavior name="ServerWcfService.Services.MySimpleServiceBehavior">

                    <serviceMetadata httpGetEnabled="true"/>

                    <serviceDebug includeExceptionDetailInFaults="false"/>

**<serviceCredentials>**

**<serviceCertificate** **findValue="MyServerCert"** **x509FindType="FindBySubjectName"storeLocation="LocalMachine"** **storeName="My"/>**

**<userNameAuthentication** **userNamePasswordValidationMode="Custom"customUserNamePasswordValidatorType="ServerWcfService.CustomValidators.MyCustomValidator,ServerWcfService"/>**

**</serviceCredentials>**

                </behavior>

            </serviceBehaviors>

        </behaviors>

    </system.serviceModel>

加粗的那些是我加上去的或者在默认上修改了的。Bindings节指定了客户端提供的认证类型为'username'并在endpoint节中指定bianding配置。在dns节中修改原来的localmachine为MyServerCert，当然你也可以修改为别的，这取决于你的证书主题名称是什么。也就是上面命令中的CN=MyServerCert，接下来我们加入在serviceCredentials配置节，并在里面配置两个小节，ServiceCertificate节中指定了我们的X509证书的位置，以用来加解密message，usernameAuthentication节中指定了我们自己的用户名密码验证逻辑。

Sorry，忘了一件事情，就是写一个测试的服务契约并实现，写法上和无认证的写法一样，如下ServerWcfService.ServiceContracts.IMySimpleService：

[OperationContract]

string PrintMessage(string message);

这样，服务端的代码编写和配置就完成了，生成项目测试一下，页面显示服务已生成成功。

接下来我们开始编写客户端代码，先在客户端引用刚才生成的WCF服务，然后编写客户端代码如下：

protected void btnPrint\_Click(object sender, EventArgs e)

{

TestWCFService.MySimpleServiceClient client = new ClientWeb.TestWCFService.MySimpleServiceClient();

client.ClientCredentials.UserName.UserName = "xiaozhuang";

client.ClientCredentials.UserName.Password = "123456";

lbMessage.Text = client.PrintMessage(txtMessage.Text);

}

如果你有一个真正的X509证书，那么现在的代码就可以正常运行了。但是很不幸，我们的证书是测试用的，我们运行的时候出错：'X.509 certificate CN=MyServerCert 链生成失败。所使用的证书具有无法验证的信任链。请替换该证书或更改 certificateValidationMode。已处理证书链，但是在不受信任提供程序信任的根证书中终止'，WCF无法验证测试证书的信任链，那我们要做的就是绕过这个信任验证，具体做法如下：

先要在Asp.net Web应用程序项目上添加引用'System.IdentityModel'然后我们建立一个新的类文件并继承自'System.IdentityModel.Selectors.X509CertificateValidator'，然后我们重写里面的'Validate'方法来实现我们自己的X509认证逻辑，代码如下：

using System;

using System.Configuration;

using System.IdentityModel.Selectors;

using System.IdentityModel.Tokens;

using System.Security.Cryptography.X509Certificates;

namespace ClientWeb.CustomX509Validator

{

    /// <summary>

    /// Implements the validator for X509 certificates.

    /// </summary>

    public class MyX509Validator: X509CertificateValidator

    {

        /// <summary>

        /// Validates a certificate.

        /// </summary>

        /// <param name="certificate">The certificate the validate.</param>

        public override void Validate(X509Certificate2 certificate)

        {

            // validate argument

            if (certificate == null)

                throw new ArgumentNullException("X509认证证书为空！");

            // check if the name of the certifcate matches

            if (certificate.SubjectName.Name != ConfigurationManager.AppSettings["CertName"])

                throw new SecurityTokenValidationException("Certificated was not issued by thrusted issuer");

        }

    }

}

你可以把Validate方法里面留空让所有的认证都通过，也可以自己定义认证逻辑，如果认证失败，就抛出'SecurityTokenValidationException'的异常，然后我们配置一下客户端的webconfig让它使用我们自己的X509认证，增加以下的配置节，并在'endpoint'节中指定behaviorConfiguration="myClientBehavior"。

<behaviors>

            <endpointBehaviors>

                <behavior name="myClientBehavior">

                    <clientCredentials>

                        <serviceCertificate>

                            <authentication certificateValidationMode="Custom"customCertificateValidatorType="ClientWeb.CustomX509Validator.MyX509Validator,ClientWeb" />

                        </serviceCertificate>

                    </clientCredentials>

                </behavior>

            </endpointBehaviors>

        </behaviors>