

就他吧-9ta8为您提供：身份证查询、15位转16位身份证，手机号码归属地查询，IP地址查询服务，城市天气预报查询，列车时刻表简易快速查询等等查询服务，就他吧欢迎您的光临！！

清新的空气清晰的你愉快的心情陪伴你，欢迎来到梦幻Dot Net。



<	2006年3月						>
日	一	二	三	四	五	六	
26	27	28	1	2	3	4	
5	6	7	8	9	10	11	
12	13	14	15	16	17	18	
19	20	21	22	23	24	25	
26	27	28	29	30	31	1	
2	3	4	5	6	7	8	

与我联系

发短消息

搜索

常用链接

- 我的随笔
- 我的空间
- 我的短信
- 我的评论
- 更多链接

留言簿

- 给我留言
- 查看留言

梦幻Dot Net

科学地研究编程，品味开发的艺术



[博客园](#) [首页](#) [社区](#) [新随笔](#) [联系](#) [订阅](#) [XML](#) [管理](#)

随笔-103 评论-682 文章-16 trackbacks-102

使用WSE实现Web Service安全

WSE(Web Services Enhancements)是微软为了使开发者通过.NET创建出更强大，更好用的Web Services而推出功能增强插件。现在最新的版本是WSE2.0(SP2).本文描述了如何使用WSE2.0中的安全功能增强部分来实现安全的Web Services。WSE的安全功能增强实现的是WS-Security标准，此标准是WebService自己的安全协议，由IBM, BEA, Microsoft等联合制定，所以在.NET和Java系统上都可实现。

我主要是根据WSE的文档说明和自己使用体会(其实多半也是按照文档画瓢，呵呵)而写，有错误之处请指出。另外，这是用的都是2.0,它与WSE1.0相比，变化很大，尤其在安全方面。

还有一点注意，其实通过WSE实现安全有两种途径：一种就是我们下面要介绍的通过编写代码的方法；另外一种是直接编写策略文件(XML文档)，这两种方法本质上都是通过对传递的SOAP消息进行设置，如增加用户消息，加解密，签名验证等，来实现安全功能的。但本人对第二种方法不太熟悉，也没时间研究，就不写了，呵呵。

一、使用用户名和口令验证Web Services调用者身份。

原理很简单：客户端通过SOAP扩展，在SOAP消息中加入用户名和口令(明文或加密)，发送给Web Services端；服务端接到消息后，同样通过扩展从消息上下文中得到用户名和口令，再进行身份验证和其他操作。下面是实现步骤：

我参与的团队

- [Design & Pattern团队\(0/0\)](#)
- [.NET 控件与组件开发\(0/0\)](#)
- [敏捷软件开发组织\(0/0\)](#)
- [北京.NET俱乐部\(0/0\)](#)
- [Dot Net Web服务和Windows服务开发\(0/0\)](#)
- [正则表达式团队\(0/0\)](#)

随笔分类

- [.Net设计模式\(21\)](#)
- [ASP.Net\(9\)](#)
- [C#\(12\)](#)
- [C#2005\(3\)](#)
- [DotText](#)
- [JavaScript\(8\)](#)
- [Reporting Services\(7\)](#)
- [SD Channel](#)
- [SQL Server 2000\(10\)](#)
- [Visio Studio 2005 WPF\(2\)](#)
- [Web and Windows Service\(5\)](#)
- [其他\(8\)](#)
- [软件工程\(7\)](#)
- [正则表达式\(2\)](#)

随笔档案

- [2008年12月 \(1\)](#)
- [2008年10月 \(2\)](#)
- [2008年3月 \(1\)](#)
- [2008年1月 \(2\)](#)
- [2007年12月 \(1\)](#)

客户端:

1.添加Microsoft.Web.Services2和客户端要访问的Web Services引用，没有什么好说的。当然，这两个引用是必须的，你可能还需要客户端需要添加其他引用。

2.修改从Web Services引用生成的本地proxy类，这个类的代码在引用生成Reference文件中。从.NET开发环境右边的解决方案资源管理器里打开你要操作的Web Services引用文件夹，打开Reference.map节点，就可以看到Reference.cs或Reference.vb文件。如果你没有看到这些文件，可能是没有显示所有文件，你需要在解决方案资源管理器或命令菜单“项目”里设置“显示所有文件”。找到Reference文件后，打开它，在proxy类的声明处将类的继承父类改成Microsoft.Web.Services2.WebServicesClientProtocol.因为只有这样，proxy类才能访问WSE提供的SOAP扩展。注意，如果更新了Web服务的引用，则需要重新把继承类修改。

3.通过UsernameToken类的实例添加用户名与口令。UsernameToken属于Microsoft.Web.Services2.Security.Tokens命名空间。假设Web Services的本地proxy类名称为WebServer.WebService,用户名为Username,用户口令为Userpwd,则代码可以如下所示(vb.net，下同):

'生成本地proxy类实例

Dim mywebserv as New WebServer.WebService

'生成UsernameToken类实例，将用户名，用户口令和口令发送方式写在实例中

Dim untoken As UsernameToken = New UsernameToken(Username, Userpwd, PasswordOption.SendPlainText)

'设置SOAP消息有效期，以确减少消息即使被其他用户截获后重新使用的可能性为，这里设置为30秒，但要注意不同系统时钟同步的问题。

mywebserv.RequestSoapContext.Security.Timestamp.TtlInSeconds = 30

'将UsernameToken实例加在SOAP消息上下文中

mywebserv.RequestSoapContext.Security.Tokens.Add(untoken)

'调用Web服务的方法WebMethod

mywebserv.WebMethod

这里需要说明口令发送方式的设置。口令发送方式为枚举型数

2007年11月 (1)
2007年10月 (4)
2007年8月 (1)
2007年7月 (1)
2007年5月 (1)
2007年4月 (1)
2007年3月 (1)
2007年1月 (2)
2006年12月 (1)
2006年10月 (1)
2006年9月 (2)
2006年8月 (1)
2006年7月 (1)
2006年4月 (1)
2006年3月 (4)
2006年2月 (1)
2006年1月 (1)
2005年12月 (1)
2005年11月 (1)
2005年10月 (5)
2005年9月 (11)
2005年8月 (36)
2005年7月 (10)

文章分类

ASP.NET(2) 
安全与加密(1) 
管理(1) 
生活(1) 
数据库(2) 

文章档案

据：**SendNone**，**SendHashed**，**SendPlainText**.分别为不发送口令，发送口令哈希值和发送口令明文，上面的例子是使用发送明文。如果选择**SendNone**，表示服务端不要验证口令，这个安全级别就很低，而且如果服务需要对传递的**SOAP**消息签名，则客户端要单独提供口令对其签名；**SendHashed**是指发送的是口令的**SHA-1**哈希值，这种情况下口令保护安全，这也是微软推荐的最好方式。但它需要服务器端通过编写代码和**config**文件设置自己来验证用户身份，具体验证方法在下面的服务器端设置会讲到；**SendPlainText**是口令以明文方式传递，如果采取这种方式，上述代码中的**UsernameToken**最好加密，否则口令很容易被截获。加密方法以后章节将详细论述；另外，如果服务器端选择让**WSE**自动根据**Windows**活动目录域的用户进行身份验证，那这里必须选择发送明文。

服务端：

1.首先在**Web Services**的配置文件**Web.config**里添加配置**WSE**的元素，这也是**.NET**开发的系统使用**WSE**最基本的一步。**WSE**文档上说如果服务的调用端是**ASP.NET**系统时才需要这一步的配置，如果是普通的**Client**程序则不需要加这个标识。可我测试发现即使调用端是**Client**程序，还是需要加这个配置。下面是完整的配置文档。

```
<configuration>

  <system.web>

    <webServices>

      <soapExtensionTypes>

        <add type="Microsoft.Web.Services2.WebServicesExtension,
Microsoft.Web.Services2,Version=2.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" priority="1" group="0"/>

      </soapExtensionTypes>

    </webServices>

  </system.web>

</configuration>
```

当然，一般情况下你只需要在**Web.config**的**<system.web>...</system.web>**之间添加**<webServices>...</webServices>**部分。另外，**add**元素的数据最好写成一行，否则可能会出错。

[2007年12月 \(1\)](#)

[2005年10月 \(1\)](#)

[2005年9月 \(1\)](#)

[2005年8月 \(4\)](#)

相册

[宝贝动物](#)

[精彩贴图](#)

[生活餐具](#)

[.NET技术网站](#)

[ASP.NET开发实践系列课程](#)

[ASP酷技术网](#)

[CodeProject](#)

[KnowDotNet](#)

[Visual Studio.net专栏](#)

[博客堂](#)

[发赛特技术网](#)

[酷网博客](#)

[孟宪会.NET开发者园地](#)

[太平洋开发编程技术与实例](#)

[天新网DotNet](#)

[新一代技术网](#)

[站长中国](#)

[正则表达式](#)

[中国C#技术网站](#)

[中文C#站](#)

其他技术网

[ASP.NET Forums](#) 官方中文网站

[MSDN中文站](#)

[动态网站制作指南](#)

2.配置服务端的调用身份验证行为，这一步是可选的。

如前所述，调用方发过来的**SOAP**消息中已加入了用户信息，服务端要做的工作是将这些信息解析出来，再根据一定规则对比验证，并返回结果。

上述处理过程也有两种选择：一是让**WSE**自动验证，我们自己的代码和配置文件再也不用做什么了。这种情况下，**WSE**在服务端收到调用方的**SOAP**消息后，从里面的**UsernameToken**中取出用户名和口令，这也是为什么前面提到过的自动验证下用户口令必须明文发送的原因，取出用户名和口令后，**WSE**是基于系统所在**Windows**域的用户进行判别和验证的。也就是说，**WSE**从活动目录里的用户列表遍历，寻找是否存在和所接收到的用户名/口令匹配的有效用户用户帐号，如果未找到匹配用户，则返回调用者未被授权的错误。由此可见，这种方法下应用系统及用户需要和**Windows**域紧密捆绑，缺乏灵活性并且不适合与已有业务系统对接。因此在实际应用中更多的应该用下面第二种方法。

这种方法的核心是**Web Services**通过继承**UsernameTokenManager**类，并重载**AuthenticateToken**方法实现的。

a.首先声明一个从**UsernameTokenManager**继承下来的类。

Public Class CustomUsernameTokenManager

Inherits UsernameTokenManager


这里面有一个访问权限问题，为了使经过授权的程序集才能访问这个类，你还需要给它在声明时加上一些访问限定。因为能够访问非托管代码(**UnmanagedCode**)的程序集信任级别都是比较高的，所以可以要求能访问此类的程序集都是可以访问**UnmanagedCode**的。这样上面的声明就变成如下形式：

```
<SecurityPermission(SecurityAction.Demand,Flags:= SecurityPermissionFlag.UnmanagedCode)> Public Class CustomUsernameTokenManager
```

Inherits UsernameTokenManager

当然你也可以配置成其他权限要求，只要更改**Flags**的**SecurityPermissionFlag**枚举值即可。

b.在代码中重载**AuthenticateToken**方法。服务端接收到含有**UsernameToken**实例的**SOAP**消息后，**WSE**将**UsernameToken**反序列化，并调用**VerifyToken**方法，而**VerifyToken**方法在执行过程中又会调用**AuthenticateToken**方法，这个方法会返回一个口令值，**WSE**会拿它与**UsernameToken**中的口令进行对比。如果发送的口令为明文(**UsernameToken.PasswordOption=SendPlainText**),则直接对比；如果发送口令为哈希值(**UsernameToken.PasswordOption=SendHashed**)，则**WSE**会对这个返回值做一个**SHA-1**哈希运算，再将结果

<div>正则表达式网站 (Eng)</div> <div>中国BS网</div> <div>中国Office Online</div> <div>中国WEB开发者网络</div>	
数据库技术网站	
<div>SQL Server 提示与技巧</div> <div>数据仓库之路</div>	
友情连接	
<div>就他吧服务</div> <div>就他吧为您提供软件开发, 网站设计, 网站开发维护, 各种办公软件、管理软件开发等等, 详情请进入http://www.9ta8.com</div> <div>幽默笑话</div> <div>笑话大全、智慧快餐、暴笑网文、搞笑图片、恐怖故事、幽默短信、相声小品尽在其中, 就他吧幽默站开心您每一天</div>	
积分与排名	
<div>积分 - 324947</div> <div>排名 - 137</div>	
最新评论 	
<div>1. Re:正则表达式基础知识</div> <div>特价机票</div> <div>--特价机票</div>	
<div>2. Re:.NET反射、委托技术与设计模式</div> <div>[code=csharp] [/code]</div> <div>--黎定文</div>	
<div>3. Re:.NET反射、委托技术与设计模式</div>	

与UsernameToken中的口令进行对比。如果不一致, 则返回用户未被授权的错误。上述过程全部是自动完成的, 因此我们要做的工作就是重载AuthenticateToken方法, 在里面实现返回正确的用户口令, 用于对比和验证。这其实就相当于WSE1.0里面的PasswordProvider.

重载AuthenticateToken方法实现的逻辑由系统根据具体要求来定, 比如根据用户名去数据库里寻找相应的用户口令, 或者从LDAP中等等。这里就照找了WSE文档上的例子, 返回的口令和提交的用户名相同。

```
Protected Overrides Function AuthenticateToken(ByVal userName As UsernameToken) As String
```

```
' Ensure that the SOAP message sender passed a UsernameToken.
```

```
    If userName Is Nothing Then
```

```
        Throw New ArgumentNullException
```

```
    End If
```

```
    ' This is a very simple provider.
```

```
    ' In most production systems the following code
```

```
    ' typically consults an external database of (userName,hash)
```

```
    ' pairs. For this example, it is the UTF-8
```

```
    ' encoding of the user name.
```

```
    Dim encodedUsername As Byte() = _
```

```
        System.Text.Encoding.UTF8.GetBytes(userName.Username)
```

```
        Return System.Text.Encoding.UTF8.GetString(encodedUsername)
```

```
    End Function
```

c. 配置web.config文件, 告知Web Services使用哪个类来验证用户身份。

首先在文件中加入标明使用WSE2的元素:

```
<configuration>
```


好文章！不错

--阿宝QQ

4. [Re:Dot NET设计模式—反射工厂](#)
做人要厚道 转载应注明

--passer

5. [Re:Visual C#2005中的匿名委托的实现](#)

文章写的不错的。我的站支持一下
谢 网站名称： IT资源网 网站地址：
网站描述： IT资源网内容有： 计算
机图书,电脑图书,计算机书籍,电脑
书籍,下载: ...

--dwf

阅读排行榜

- 1. [全面剖析C#正则表达式\(30162\)](#)
- 2. [正则表达式基础知识\(28226\)](#)
- 3. [C#反射实例讲解\(19380\)](#)
- 4. [.NET反射、委托技术与设计模式\(13107\)](#)
- 5. [.Net调用Java的WebService之亲身体验\(12663\)](#)

评论排行榜

- 1. [DotNet Webservice和WindowsService团队开通\(130\)](#)
- 2. [如何掌握并在实践中自如运用设计模式\(58\)](#)
- 3. [C#写的一个代码生成器\(36\)](#)
- 4. [用.Net开发Windows服务初探\(27\)](#)
- 5. [.NET反射、委托技术与设计模式\(24\)](#)

<configSections>

<section name="microsoft.web.services2"

type="Microsoft.Web.Services2.Configuration.WebServicesConfiguration,
Microsoft.Web.Services2, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />

</configSections>

</configuration>

这里要注意的是一定要把<configSections>...</configSections>写到整个<configuration>...</configuration>里的最前面，否则会出现“响应内容类型为"text/html;charset=utf-8",但应该是"text/xml"”的错误。

其次配置使用UsernameTokenManager子类

<configuration>

<microsoft.web.services2>

<security>

<securityTokenManager xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"qname="wsse:UsernameToken"
type="MyNamespace.CustomUsernameTokenProvider,MyAssemblyName,Version=2.0.0.0,
Culture=neutral,PublicKeyToken=81f0828a1c0bb867" />

</security>

</microsoft.web.services2>

</configuration>

在这里面type属性的值要保持在一行，"MyNamespace.CustomUsernameTokenProvider"必须是派生类有效的类层次名，MyAssemblyName为程序集名称。另外Version，Culture和PublicKeyToken等属性可以省略。

这样，我们就完成了在Web Service调用最基本的基于用户名/口令的验证。以后各项安全功能的实现都以它为基础。

二、使用用户名和口令对SOAP消息签名

上述过程虽然实现了Web Services用户身份确认。但它不能保证Web Services接收到的SOAP消息就是所声明用户所发送的，因此在实际的应用中是还需要调用方对SOAP消息做一次签名，并将签名连同消息一起发送；服务端接收到消息后，除了验证用户身份外，还需要对其中的签名做一次认证。以确定消息在传输过程中没有被更改过，而验证的用户就是对消息签名的用户。

客户端：

只需要在原有客户端SOAP消息中添加签名即可，如下所示,黑体为新加代码：

```
mywebserv.RequestSoapContext.Security.Tokens.Add(untoken)
```

```
mywebserv.RequestSoapContext.Security.Elements.Add(New _  
MessageSignature(untoken))
```

```
mywebserv.WebMethod
```

上述代码就是客户端根据UsernameToken的实例生成一个签名，然后把签名加在SOAP消息里，具体是WS-Security扩展的SOAP消息头。

服务端：

其实服务端无需改动什么。只要客户端发送的SOAP消息里面包含了签名，服务端就会自动验证签名的有效性。服务端首先验证用户名/口令，然后使用客户端传递的用户名和自己获得的口令(WSE自动从Windows活动目录中获得，或者通过重载的AuthenticateToken的方法)对签名进行验证。如果验证失败，说明消息在传递过程中被更改过或者不是当前调用用户所签，返回相应错误。

WSE的帮助文档上这一块加了一个判断SOAP里是否包含签名的函数，通过它可以获得签名的一些相关信息，并不是必须的。有兴趣的朋友可以自己去查看。

三、使用证书验证身份并对SOAP消息签名

使用用户名/口令存在固有缺点，因此在某些安全要求较严格的系统中我们还需要使用证书来完成对用户身份的验证。关于PKI/CA的基础知识这里就不介绍了，大家可以去看相关资料。

首先我们需要配置服务端保证WSE可以访问证书及其私钥。（至于证书的申请，管理和使用属于基础知识，在这就不讲了）这里主要是在Web Services端设置，以保证服务端可以自动完成某些功能，还无需用户干预。在Web.config文件添加如下<x509>元素(黑体部分)：

```
<configuration>

  <microsoft.web.services2>

    <security>

      <x509 storeLocation="CurrentUser" />

    </security>

  </microsoft.web.services2>

</configuration>
```

storeLocation属性为WSE可访问的证书存储。它可选的值有两项：“LocalMachine”和“CurrentUser”，分别表示本机证书存储和当前用户证书存储。这里用的是CurrentUser，而storeLocation缺省值是LocalMachine.另外几个属性的配置如下：
verifyTrust,是否验证所用证书的证书链有效性，缺省为true；allowTestRoot：验证证书链过程中是否认为测试根CA所签发证书有效，这个参数在verifyTrust为true时才有效，缺省为false；allowRevocationUrlRetrieval，是否在线验证证书是否被吊销，在verifyTrust为true时才有效，缺省为true；allowUrlRetrieval,是否在线验证证书链有效性，allowRevocationUrlRetrieval，是否在线验证证书吊销状态。

客户端：

第1、2步与第一节使用用户名/口令的步骤基本相同。但这里也要多加一个引用：Microsoft.Web.Services2.Security.X509

3. 编写代码取得要使用的证书。我这里用了一个listview控件显示当前用户证书存储里的个人证书，以供用户选择。你当然也可以象WSE文档上那样利用证书的属性去定位需要使用的证书。Listview控件名称为lv_certlist，它有两列：证书持有者主体和颁发者名称。

```
Dim cert As X509Certificate

Dim lvItem As ListViewItem
```


'从当前用户证书存储中打开个人证书库

```
certstore = X509CertificateStore.CurrentUserStore(X509CertificateStore.MyStore)
```

```
certstore.Open()
```

```
lv_certlist.Items.Clear()
```

'遍历证书，写到listview控件里

```
For Each cert In certstore.Certificates
```

```
    lvitem = lv_certlist.Items.Add(cert.GetName)
```

```
    lvitem.SubItems.Add(cert.GetIssuerName)
```

```
    lvitem.Tag = lvitem.Index
```

```
Next cert
```

4. 选择所选择的证书生成X509SecurityToken的实例，它和前面的UsernameToken一样是SecurityToken的子类。

```
Dim cert As X509Certificate
```

```
Dim certToken As X509SecurityToken
```

```
Dim result As String
```

```
cert = certstore.Certificates(lv_certlist.SelectedIndices(0))
```

'判断所选择证书是否支持签名，并且私钥是否存在

```
If cert.SupportsDigitalSignature And Not (cert.Key Is Nothing) Then
```

'遍历证书，写到listview控件里

```
certToken = New X509SecurityToken(cert)
```

```
Dim mywebserv As New WebServer.WebService
```

```
mywebserv.RequestSoapContext.Security.Timestamp.TtlInSeconds = 30
```

'添加包含证书信息的X509SecurityToken

```
mywebserv.RequestSoapContext.Security.Tokens.Add(certToken)
```

'使用证书对SOAP消息签名，并将结果写在消息中

```
mywebserv.RequestSoapContext.Security.Elements.Add(New _  
MessageSignature(certToken))
```

'调用Web服务的方法

```
mywebserv.WebMethod  
End If
```

服务端：

和验证使用用户名/口令签名的消息类似，服务端也同样自动验证签名有效性。同时，服务端会根据web.config文件中<x509>元素里的属性设置来对证书的有效性进行验证。如果验证失败，则返回相应错误。

四、使用用户名和口令对SOAP消息加密

前面第一节讲过，如果UsernameToken实例中的口令是明文，那最好将UsernameToken加密发送。在第一节客户端代码中做如下改动,黑体代码为新加：

```
mywebserv.RequestSoapContext.Security.Tokens.Add(untoken)
```

```
mywebserv.RequestSoapContext.Security.Elements.Add(New _  
Microsoft.Web.Services2.Security.EncryptedData (untoken))
```

```
mywebserv.WebMethod
```

上述代码就是客户端根据UsernameToken的对SOAP消息加密，然后把密文加在SOAP消息里，具体是WS-Security扩展的SOAP消息头

服务端自动对消息解密。服务端首先验证用户名/口令，然后使用客户端传递的用户名和自己获得的口令(WSE自动从Windows活动目录中获得，或者通过重载的AuthenticateToken的方法)对解密数据。如果失败，返回相应错误。

WSE的帮助文档上这一块同样加了一个判断SOAP是否加密的函数。

五、使用证书对SOAP消息加解密

这个略微复杂一些。根据PKI非对称加解密原理，对消息加密的客户端需要事先得到作为接收方的服务端的证书公钥，而服务端必须保证可以访问其证书对应的私钥。因此我们首先要在服务端选择设置好它所使用的证书，原则就是使服务端能够随时并自动访问其证书对应的私钥。

对于基于ASP.NET开发的Web Services，需要给它运行时的缺省用户赋予访问证书私钥的权限。运行在IIS 6.0上的Web Services缺省用户是Network Service，其他情况下用户账户由machine.config文件里<processModel>元素的userName属性决定，缺省情况下是“machine”，它等同于ASPNET账户。添加步骤如下：运行WSE 2.0自带的X.509 Certificate tool（开始---程序---Microsoft WSE 2.0---X.509 Certificate tool），选择并打开WSE要访问的证书。打开后，“Private Key File Name”文本框里会显示证书所对应私钥文件的名称，Private Key File Folder”文本框里会显示私钥文件的路径。单击“Private Key File Properties”打开文件属性对话框，可以在安全属性页里添加被授权访问的Network Service或ASPNET用户账户。

如果打开证书后发现私钥文件不存在或不可访问，说明此证书有问题或者私钥受保护，比如有口令保护或者存储在安全外设中，不能被Web Services随时自动访问，因此也就不适用。

对于Web Services，如果使用的用户是缺省的ASPNET，那最好选择本机存储（LocalMachine Store）里的证书。因为ASPNET这个用户是安装ASP.NET时自动生成的，它的口令是没有办法访问的。如果证书包含在当前用户存储（CurrentUser Store）中，服务端很可能就访问不到私钥。

服务端设置好证书后，接下来就要使客户端得到这个证书和公钥。具体的方法和你的业务系统 and 应用需求有关系，你可以把它放在网页上让用户事先下载安装；如果应用在局域网环境，可以让客户端去LDAP或CA里取得。

客户端做如下改动：

```
mywebserv.RequestSoapContext.Security.Tokens.Add(certToken)
```

’SOAP消息中添加加密结果

```
mywebserv.RequestSoapContext.Security.Elements.Add(New _  
Microsoft.Web.Services2.Security.EncryptedData(certToken))
```

```
mywebserv.WebMethod
```

服务端接收到加密后的**SOAP**消息后，自动使用相应私钥解密。如果失败，返回相应错误。

可以看出，使用证书加解密消息的难点在于证书的选择和配置，尤其是消息接收端。

WSE除了支持上述用户名/口令，X509证书两种令牌外。还支持**Kerberos Ticket**，安全上下文令牌（**Security Context Token**）和用户自定义令牌。**Kerberos Ticket**好象是WSE自己加的内容，标准的WS-Security里没有，而且只在**Windows XP SP1/SP2**，**Windows2003**上支持。安全上下文令牌需要一个上下文令牌服务生成。而用户自定义令牌的核心是由**SecurityToken**派生一个用户自己的令牌类，完成各种安全功能。总之WSE就是使用这种基于令牌和扩展**SOAP**消息来实现**Web Services**安全的。

0

0

(请您对文章做出评价)

posted on 2006-03-23 09:57 [振河](#) 阅读(3945) [评论\(2\)](#) [编辑](#) [收藏](#) [网摘](#) 所属分类: [Web and Windows Service](#)

评论:

[#1楼](#) 2007-12-14 09:46 | [chenfeile](#)[未注册用户]

楼主你好.

看了上面的文章,写得很好,也有实践.但实践期间出了不少问题.

你能不能源码也贴上来.

[回复](#) [引用](#)

[刷新评论列表](#) [刷新页面](#) [返回页首](#)

发表评论

昵称:

[\[登录\]](#) [\[注册\]](#)

主页:

邮箱: (仅博主可见)

评论内容: [闪存](#) [个人主页](#)

[登录](#) [注册](#)

[使用Ctrl+Enter键快速提交评论]

[Windows 7交流小组](#)

[个人主页上线测试中](#)

[今天你闪了吗?](#)

[2009博客园纪念T恤](#)



[China-pub](#) 计算机图书网上专卖店! 6.5万品种 2-8折!

[China-Pub](#) 计算机绝版图书按需印刷服务

链接: [切换模板](#)

导航: [网站首页](#) [个人主页](#) [社区](#) [新闻](#) [博问](#) [闪存](#) [网摘](#) [招聘](#) [找找看](#) [Google搜索](#)

最新**IT**新闻:

[优酷网六点声明:拥护反盗版联盟支持正版](#)

[微软Windows Live新增23家合作网站](#)

[李彦宏弯道超车论引爆中美经贸论坛](#)

[微软重金招揽实习生:月收入最高6千美元](#)

[Facebook全球用户数突破3亿:现金流转正](#)

相关搜索:

[Web and Windows Service](#)

相关链接:

[Webservice的设计和模式](#)

[remoting和webservice有什么区别](#)

《[博客园精华集 - Web标准之道](#)》

热曲先听: [客房服务 Hotel Room Service](#)

海外著名Web设计师们的工作台

博客园推荐图书:《[高性能网站建设指南](#)》

修正版 [疯狂代码](#) 写给WEB2.0的站长

Powered by: [博客园](#) 模板提供: [沪江博客](#) Copyright ©2009 [振河](#)

就他吧-9ta8伴您开心每一天