# Web 服务认证①

## Web Service Authentication

冯 挺 (宁波大学 信息科学与技术学院 浙江 宁波 315211)

王 惠 (宁波中搜网络科技有限公司 浙江 宁波 315040)

**摘 要：** 随着 Web 服务的普及，Web 服务的安全性变得越来越重要。在总结了目前比较成熟的多种身份认证方法后，对其优缺点和特性进行了总结。并且，对其中比较有实际应用价值的方案进行了具体的实现。同时，还对目前网络上流行的几种 Web 服务的身份认证方法进行了分析。

**关键词：** Web 服务 身份认证 HTTP 验证 SOAP 头验证 令牌验证

Web 服务的认证，可以防止未经认证的用户随意使用公开的 Web 服务，使得用户必须要通过认证后，获得相应权限，才能使用相应的 Web 服务。目前比较成熟的 Web 服务认证方式有 HTTP 认证，自定义认证以及采用现有工具包。本文详细分析了各种方式的特点，并实现了几种比较简单实用的认证方式。

## 1 引言

W3C 组织定义 Web 服务(Web Service)为“在网络上支持机器间的互操作的一种软件系统”。通常，它是以 Web API 的形式存在于 Internet 上，通过 HTTP 协议在服务器端和客户端进行通信。利用 Web 服务可以建立面向服务的架构(SOA，Service-oriented architecture)，即不用改变应用，利用 Web 服务的消息驱动机制，让分布式系统协同工作。Web 服务平台提供了一套标准的类型系统，用于沟通不同平台、编程语言和组件模型中的不同类型系统。

随着一些关键的 Web 服务标准的纷纷制定， 越来越多的企业采用 Web 服务技术进行应用开发。和 Internet 上其它的应用一样， Web 服务也面临着安全性风险， 因为信息有可能被盗、丢失和被篡改。安全的 Web 服务是应用成功的必要保证。通常，Web 服务是面向大众公开的，所以其安全问题显的尤其重要。安全的 Web 服务需要保证以下 5 项安全性要求；“①认证:提供某个实体(人或者系统)的身份的保证；②授权:保护资源以免对其进行非法的使用和操纵；③机密性:保护信息不被泄漏或暴露给未授权的实体:④完整性:保护数据以防止未授权的改变、 删除或替代；⑤不可否认性: 防止参与某次通信交换的一方事后否认本次交换曾经发生过。”[1]

但是，在实际应用中，完整的满足上述 5 项要求对于普通的 Web 服务应用太过复杂，同时也太浪费资源。目前 Internet 上的公共 Web 服务，对于一般的安全性要求，只需满足上述 1、2 两项要求即可。对于授权，也是通过认证后的用户身份，进行相应的权限分配。因此，Web 服务的认证显得尤为重要。本文列举了目前可以比较成熟的几种认证的方法，并对其优缺点进行了讨论。同时，也对目前比较普及的应用度较广的Web服务的认证方式进行了总结。

## 2 Web服务认证方式

目前比较成熟的 Web 服务的认证方式，可分为通过 HTTP 认证、自定义方式以及采用现有开发包 3 种。

① 收稿时间:2008-12-08

### 2.1 透明认证，采用 HTTP 认证机制

对于公共的 Web 服务，可以采用和 HTTP 服务一样的集中安全方式，包括基本(Basic)、摘要(Digest)、集成(Integrated)和基于证书的(Client Certificate)的认证机制。在 IIS 上，只能用内置帐号数据(包括采用 Windows 活动目录)来认证。但是，可以实现自定义的 HTTP 认证机制以采用任意的凭据数据库。采用这些方式的一个最大的好处是简单，这些机制已经成为标准，客户端不需要学习特定 Web 服务的 API 然后采用特定的认证代码来使用 Web 服务。

#### 2.1.1 基本认证(Basic)

基本认证是服务器直接验证客户端提供的明文用户名和密码是否和保存的用户信息相匹配的，如果匹配则通过认证。其优点是算法最简单，资源占用少，认证只需要 1 个来回；缺点是不安全，密码是经过简单的 Base64 编码明文传送，如果需要安全性则需要采用安全套接层协议 HTTPS。

#### 2.1.2 摘要认证(Digest)

比基本认证稍微难一点实现，但也只需要一个来回的认证。密码不再是明文传送，而是传送密码及其他信息综合后生成的哈希值，因此这种认证方式不需要传送加密。它的优点是密码不用明文传输，比较安全；服务器端可以通过一定的哈希值生成算法在一定程度上抵制黑客攻击。它的缺点是算法比较复杂，实现起来比较麻烦。

#### 2.1.3 集成 Windows 身份认证(Integrated)

集成认证又分为 NTLM 和 Kerberos2 种。NTLM 认证方式需要把用户的 Windows 用户名和密码传送到服务端，服务端认证用户名和密码是否和服务器的此用户的信息一致。用户名用明码传送，但是密码是经过处理后派生出的一个 8 字节的密钥加密后传送。Kerberos 认证方式只把客户端访问 IIS 的认证票(Ticket)发送到 IIS 服务器，IIS 收到这个票据就能确定客户端的身份，不需要传送用户的密码。需要 kerberos 认证的用户一定是域用户。

虽然集成认证安全性比较高，但其只适用于 Intranet 和 Windows 客户端，而且采用的用户信息是用户当前登录的 Windows 帐户信息，所以在 Internet 上实用性不高。

#### 2.1.4 客户端证书认证(Client Certificate)

对于安全要求特别高的环境，可以使用客户端证书认证。客户端证书为 Web 应用程序提供了一种出色的身份认证机制。浏览器或其他客户端应用程序必须提供有效的证书才能被授予对应用程序的访问权，从而使客户端无需再提供用户名和密码。这就使得在创建由其他客户端应用程序访问的安全 Web 服务时，客户端证书非常有用。

但是这种方式需要证书服务器的支持，而且比摘要认证复杂的多，而且十分消耗资源。虽然安全性很高，但对于一般程序，实用性不高。

### 2.2 自定义安全

自定义安全是采用自己定义的接口，来认证用户的身份。Web 服务公开认证接口，定义认证 API，客户端则按照 Web 服务的公开信息来实现认证。这种方式的优点是灵活，服务器端可以随意定义各种认证所需要的接口和算法，而不必拘泥于现有方法。例如，虽然用户凭据通常以用户名、密码的方式存在，但是很多情况下，还需要其他信息来配合验证，比如在船用公司服务系统中还需要公司编号。但这种方式使客户端使用 Web 服务的要求提高了，客户端必须首先学习所需要使用的 Web 服务的认证接口，才能使用服务。而且每个 Web 服务都有其自己的认证方式，导致通用性不高。目前自定义安全基本有 2 种方式。

#### 2.2.1 SOAP header 方式[2]

这种方式是用 SOAP 头来携带凭据信息。这种方式原理上是好的，而且 WS-Security 也是这么做的。它允许独立传送，允许加密密码或者直接传送 Hash 凭据而不需要加密整个信息。但是缺点是客户端必须阅读 Web 服务的 API 文档才知道该怎么做，以手工的方式小心的以所要求的格式构建 SOAP 头，导致开发难度增大。而且，以 SOAP header 方式必须在每次

Web 服务调用上都附加相应的用户凭据，同时服务器也得每次都进行认证，比较浪费资源。

2.2.2 登录方式

把 Web 服务看作是房子，把 Web 服务的认证看作是房子的门。房子允许很多人进入，但进入之前，必须要敲门。登录认证就是为了确保使用 Web 服务的人就是所宣称的那个人。这扇门需要用户提供凭据，然后他们会得到一个安全令牌以访问服务器。服务器返回的安全令牌可以有很多种方式，可以是保存在浏览器中的 Cookie，保存在服务器上的 Session Id 或者是一串字符。

登录方式是在使用一个 Web 服务之前，先用登录信息调用一个 Web 服务附属的登录方法，如果登录成功，则得到一个每次使用 Web 服务都要用的令牌。每次使用 Web 服务时，在 SOAP Header 或者参数中携带这个令牌。这种方式的优点是不用每次调用 Web 服务都需要传输凭据信息，而只需要认证一次。缺点则是对于每一个单独的 Web 服务，它需要 2 个来回(如果需要登出以便清理会话(Session)则需要 3 个来回)；而且 Web 服务需要实现会话模型，这比无会话模型困难。

### 2.3 现有工具包

2.3.1 Web Services Enhancements

WS-Security 是一种为了保证 Web 服务安全所定义的通信协议，其完整的实现了上文所提到 5 种安全性要求。微软的 Web Services Enhancements 工具包实现了 WS-Security 标准。但使用这种工具包，学习过程异常复杂，而且作为一个系统，即使只需要其中的一小块功能，也需要学习整个工具包的使用，导致实现过程漫长。而且，使用工具包的后果就是，会导致客户端越来越依赖工具包的黑箱实现，如果正常，一切都好；如果不正常，一个小问题就会破坏整个系统。这对于一般的对安全要求不高的 Web 服务来说，得不偿失。

2.3.2 Microsoft Windows Live ID [3]

Microsoft Windows Live ID 认证是 Microsoft Windows Live 系统(以前称作 Passport)采用的一种认证方式，微软使用它提供了一种“单点登录”服务，它允许用户使用一个账户就可以登录多个 Web 网站，包括 Hotmail、.NET Messenger Service、MSN subscriptions、其他一些与微软交往密切的公司如 RadioShack 以及其他所有采用 Live ID 认证的站点。Hotmail 或 MSN 用户自动拥有与他们的账户相对应的 Microsoft Live ID。Live ID 认证消息以电子凭据的形式传递，用于通知站点该用户已经成功登录。凭据是一小段数据，包含了登录时间、上次用户登录时间及其他一些认证过程中需要的信息。在系统内部，这些凭据以 cookies 的形式存在。这种方式在 ASP.NET 上容易简单实现，但缺点是通用性较差，是过于复杂，实现困难。而且，不具有开放性，用户信息保存在 Microsoft 的数据库中，不利于特定需求的扩展。

2.3.3 Liberty Alliance Project

Liberty 是一个商业联盟，成立于 2001 年 9 月，目标是建立一个统一的身份管理的开放标准。它也支持单点登录，允许用户在支持 Liberty 的站点上登录一次，然后就能无缝的链接到其他支持 Liberty 的站点上，而不用再次认证。其相对于 Microsoft Live ID 的优点是其是开放的。

## 3 现有Web服务

### 3.1 Google Web 服务

Google 的 Web 服务包括存取 Google 帐户信息的 Google Data APIs 和简单使用 Google 服务的 Web 服务。Google Data APIs 允许用户在自己的程序中登录 Google 帐号。一旦登录，Google 返回一个令牌，每次用户存取帐户信息的时候都需要用到这个令牌。令牌在一段时间内有效。它根据客户端类别分为单用户桌面程序和多用户 Web 程序。单用户桌面程序首先 Post 用户信息到 https://www.google.com/accounts/ClientLogin，如成功登录，得到令牌。以后每次对 API 的请求，HTTP 头上都要附加令

牌信息。格式为 Authorization: GoogleLogin auth=yourAuthToken。多用户 Web 程序可以采用 AuthSub 和 OAuth 2 种方式进行认证。OAuth 是为了保证安全 API 认证定义的一套开放标准，Google Data API 实现了这套标准。AuthSub 则是在 OAuth 出现之前的认证方式。

如果不需要存取帐户信息,只是简单使用 Google 服务的 Web 服务，例如 Google 地图，则使用序列号(API Key)方式，每个需要使用 Web 服务的客户端都申请一个序列号，每次使用时，需要携带序列号信息。例如在 Google's AJAX APIs，需要以下代码

<script type="text/javascript" src="http://www.google.com/jsapi?key=ABCDEFG"></script>

### 3.2 Microsoft MapPoint.Net

采用 HTTP 认证方式里的摘要认证。

### 3.3 Amazon Web 服务

首先需要注册帐号，得到序列号 (20 个字符的字符串)和安全序列号 (40 个字符的字符串)。此序列号和帐号相关联。在使用 Web 服务时，需要附加用这个序列号和安全序列号算出来的一个数字签名。这样就能保证使用 Web 服务的用户确实是帐号所对应的这个用户

### 3.4 豆瓣 Web 服务

与 AmazonWeb 服务的认证方式相同

### 3.5 MapABC 地图服务

与 Google 地图的认证方式相同，使用 Web 服务时需要携带序列号信息。

## 4 Web服务认证实现

### 4.1 HTTP 认证机制

IIS 的内置 Basic，Digest 等方式只能用内置帐号数据（包括 Active Directory）。为了让 HTTP 认证机制能支持独立的数据库，我们需要自己编程实现。这里用 XML 文件作为用户信息数据库(当然也可以容易的转换到 LDAP 或者是独立数据库)，以 Windows 下的托管的 .NET HTTP 模块来举例。

#### 4.1.1 基本认证(Basic)

1) 创建类 BasicAuthMode，继承自 IHttpModule。需要实现 2 个函数

a) void Init(HttpApplication context) 初始化模块，并使其为处理请求做好准备。

b) void Dispose() 处理由BasicAuthMode模块使用的资源(内存除外)。

2) 重载 HttpApplication.AuthenticateRequest 进行认证在 HttpApplication.Request. Headers["Au-thorization"]有用户凭据信息，格式为"Authoriza- tion: Basic ***" (用户名和密码以 Base64 编码))

3) 重载 HttpApplication.EndRequest

如果没通过认证，则在 Response 中添加头 AppendHeader(""WWW-Authenticate","Basic Realm="RassocBasicSample"")

安装方式如下：

1) 编译 BasicAuthMod.dll，拷贝到 Web 程序的 Bin 目录下

2) 修改 web.config 文件。修改<authentication mode=”None”/>，我们不想用 ASP.NET 的内置认证机制。添加下列语句

```
<httpModules>
<add name=” BasicAuthMode” type=” BasicAuthMode,AuthMod” />
</httpModules>
<appSettings>
<add key=” BasicAuthMode _Realm” value=”RassocBasicSample” />
<add key=” BasicAuthMode _UserFileVpath” value=”~/users.xml” />
</appSettings>
```

3) 拷贝用户信息 XML 文件到程序目录

4) 在 IIS 上关闭所有的 认证机制，只打开匿名方式.

4.1.2 摘要认证

摘要认证的详细标准请参见 RFC2617。实现的基本结构类似于上文所实现的基本验证，只是需要更改具体的认证方式，更改基本认证中的第二步。

**4.2 自定义认证**

4.2.1 SoapHeader 方式

1) 创建类 AuthHeader，继承自 SoapHeader。AuthHeader 中包括用户名，密码和其他需要的信息(例如域名，IP 地址)

2) 在Web服务的每个方法中,添加SoapHeader属性。

```
[SoapHeader("AuthHeader")]
[WebMethod]
public int Add(int num1,int num2)
{
if (Authenticate())// 认证
{
// do sth      // 通过认证则执行 Web 服务
}
}
```

3) 在 Authenticate()中验证 AuthHeader 中的用户信息，如果通过验证则允许用户调用 Web 方法。

4) 可以给 Web 方法添加 EnableSession 属性来保存认证的结果，以避免每次都通过用户信息来进行验证。

4.2.2 登录方式

1) 创建登录方法

```
[WebMethod]
Public string Login(string username, string password)
{
    if (Authenticate())
    {
    Return Token();   //通过认证则返回安全令牌
    }
}
```

2) 在 Web 服务的其他方法中，增加令牌参数

```
Public int Add(string token, int num1, int num2)
{
    if  (isTokenAuthenticated())// 检查令牌是否正确，是否在有效时间内等
    {
    // do sth          //通过认证则执行 Web 服务
    }
}
```

3) 以给 Web 方法添加 EnableSession 属性来保存认证的结果，以避免每次都要检查令牌有效性。

表 1 Web 服务认证方式优缺点比较

| | 方式 | 优点 | 缺点 |
|---|---|---|---|
| HTTP 认证 | 基本验证 | 简单，占用资源少 | 不安全，密码明文传输；扩展性不高 |
| | 摘要验证 | 比 Basic 安全；可扩展算法以抵御攻击 | 实现较困难；扩展性不高 |
| | 集成验证 | 安全，和 Windows 集成 | 只能用在局域网和 Windows 系统 |
| | 客户端证书 | 非常安全 | 需要证书服务器，实现非常困难；占用资源多 |
| 自定义认证 | SOAP Header | 使用较简单，易于扩展，安全性可依据要求自定义要求实现 | 每次使用 Web 服务都要携带凭证信息；各个 Web 服务的 SOAP Header 格式不一致 |
| | 登录方式 | 使用较简单，易于扩展，相对于 SOAP Header 方式较节约资源 | 即使只是使用一次 Web 服务也需要登录；需要会话模型 |
| 现有工具包 | WSE，Live ID，Liberty Alliance | 安全性高，可靠 | 需要学习工具包的使用，复杂度太高；过于依赖工具包的实现导致扩展性不高 |

## 5　各种方式比较

表 1 列举了各种认证方式的优缺点。表 2 对各种认证方式的各种性能进行了比较，5 分为最优，1 分为最差。

表 2　Web 服务认证方式各种属性比较

| | 方式 | 安全性 | 易用性 | 通用性 | 扩展性 | 流行性 |
|---|---|---|---|---|---|---|
| HTTP 认证 | 基本验证 | 1 | 5 | 4 | 1 | 3 |
| | 摘要验证 | 2 | 3 | 3 | 1 | 2 |
| | 集成验证 | 4 | 2 | 1 | 1 | 1 |
| | 客户端证书 | 5 | 1 | 2 | 1 | 2 |
| 自定义认证 | SOAP Header | 1−5* | 4 | 4 | 5 | 3 |
| | 登录方式 | 1−5* | 4 | 4 | 5 | 5 |
| 现有工具包 | WSE，Live ID，Liberty Alliance | 4 | 1 | 2 | 2 | 3 |

*：自定义安全的安全性可自由选择。

## 参考文献

1 龚豫鄂,方家骐.Web 服务安全体系结构研究.计算机工程与设计, 2006,27(13):2341－2344.

2 曾昭毅,张南平,钟珞.利用 SOAP 标头实现 Web Service 自定义安全机制.武汉理工大学学报(信息与管理工程版), 2004,26(1):74－77.

3 李匀.网络渗透测试保护网络安全的技术、工具和过程.北京:电子工业出版社, 2007:282－283.