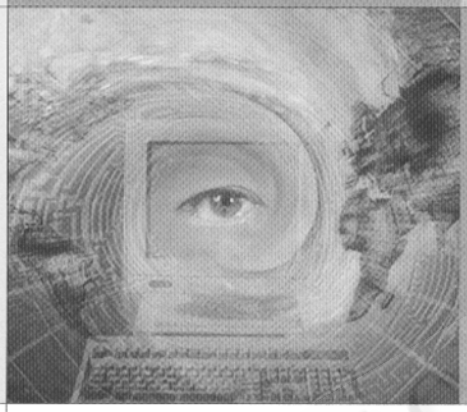
吾守尔·斯拉木 孙延鹏 傅 蓉 岳晓婷
（乌鲁木齐新疆大学信息工程学院计算机系 830046）

Web Service 模型现存问题的研究

Research on the Problems in Web Service Model



摘　要：本文通过研究 Web Service 的相关技术，对现存的模型和技术提出了新的问题和看法，并且在实践基础上提出了新的 Web Service 安全应用模型，并将此模型应用于信息服务的共享、查询、安全等各个方面，尤其在 Web Service 的安全方面提出了自己的设想，通过三层认证方式，对于 Web Service 模型中现存的不安全因素进行了相应的改进，并取得了一定成果。

关键词：Web Service 安全 认证

1 引言

本文对于Web Service模型中现存的问题进行了一些改进，如开发工具间的不兼容问题信息查询的快捷性问题等，尤其在急待解决的安全性问题方面，由于Web Service所基于的SOAP协议没有为服务的安全性提供任何的安全规范。因而需要服务开发者自己选择一种新的安全策略，本文对此提出了一些新的看法。

2 新的 Web Service 模型的设计

在进行Web Service模型设计之前，应该有一个界面友好、功能丰富的基础平台让人们更好的了解信息服务，这个平台由许多业务功能模块构成，如用户注册模块、网上交流模块、系统数据管理模块等，在此不再一一表述。下面主要介绍在这个平台上 Web Service模块的设计，分别介绍基于 Web Service的共享、查询及安全性问题。

2.1 系统的体系结构

为了更好的进行实验，探索服务的实现，我们搭建了一个系统服务平台，该系统采用三层体系结构，引入面向对象的设计思想，并针对不同的客户提供不同的服务，而且结构灵活，可方便的实现系统互联。

体系框架大体分为客户层、业务层、数据层。客户端在客户层中运行，并同业务层交互静态的XML页面或由程序动态生成的XML页面。业务层通过数据层检索数据并将数据以XML格式传回客户层。由此将XML技术和WEB服务技术集成起来，使客户端与服务端之间通过XML技术连接在一起，从而实现了跨数据平台的构想。软件的体系结构如图1所示：

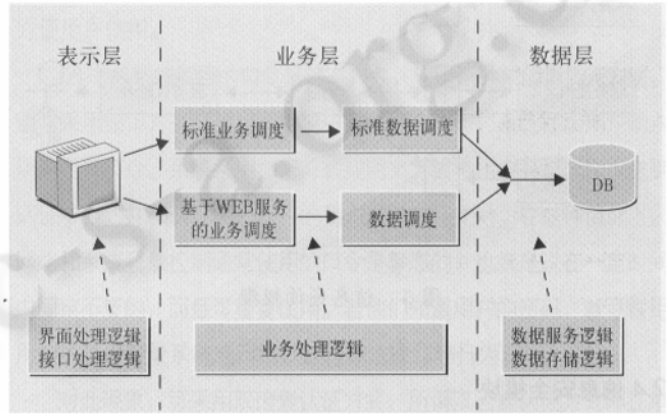


图 1　服务平台的体系结构

2.2 信息共享模块

此模块主要解决不同用户对同一Web Service提供的信息共享的问题，当不同站点同时共享同一数据库的信息时，利用Web Service注册成为Web Service的代理者，即可利用同一数据库资源共同开展网上相关业务。

信息共享使整个分销网络可分享信息、提高效率、减少风险浪费、保持数据实行一致性，为用户提供更迅速、更优质的服务。在此模型下，用户只需理解一种通用组件接口Web Service，而无需考虑内

部实现机制与开发语言等细节，即对该服务的调用是通过SOAP消息机制远程调用实现，并非在代理端实现。如图2所示：

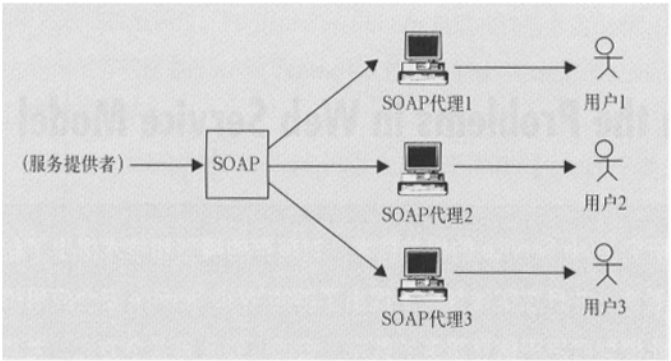


图 2　信息共享模型

2.3 信息查询模块

此模块主要用于解决对来自多个提供同一服务的Web Service的信息处理问题。我们主要采取了动态连接的方法来调用Web Service，查询来自多个服务提供者提供的同一Web Service，准确及时地获取相关信息并提供给用户。如图3所示：

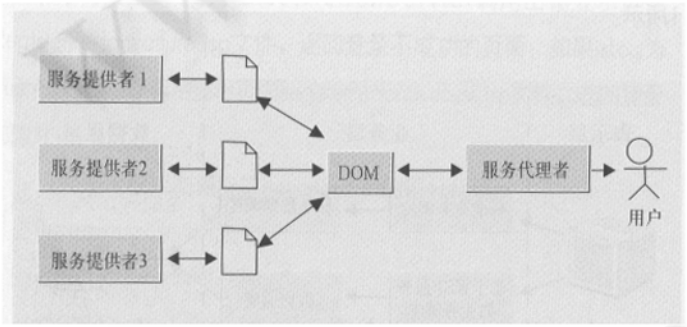


图 3　信息查询模型

2.4 信息安全模块

不论对应用程序设计者还是开发人员来说，安全性都是值得关注的重要问题。安全机制一般均采用系统安全机制与应用安全机制相结合的特点，系统安全机制的主要优点是开发者不必编写大量代码，然而采用系统安全机制时，应用也同时受到平台服务固有局限的束缚。应用本身实现的安全机制灵活性最高，但代码必须由开发者自己编写。

综合的两种方式的优点，我们可采用三层安全级别：

第一层 (Internet Information Services) IIS

第二层 ASP.NET

第三层 自定义的安全认证

这三层验证技术的采用是在认真分析了IIS与ASP.NET层的关系基础上经过不断的实验得来的，IIS与ASP.NET之间安全性关系如图4所示：

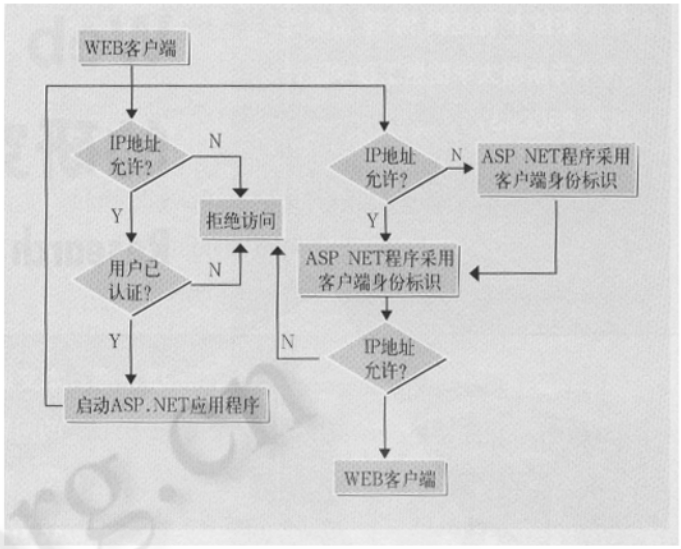


图 4　IIS 与 ASP.NET 的安全性关系

由此可见IIS与ASP.NET均提供了安全模型，以便对用户进行适当的身份验证，并在应用程序中获得正确的安全环境，两者可单独使用，也可结合使用，本文采用结合使用的办法。

IIS身份验证和ASP.NET提供的程序之间的最可能的结合如图5所示：

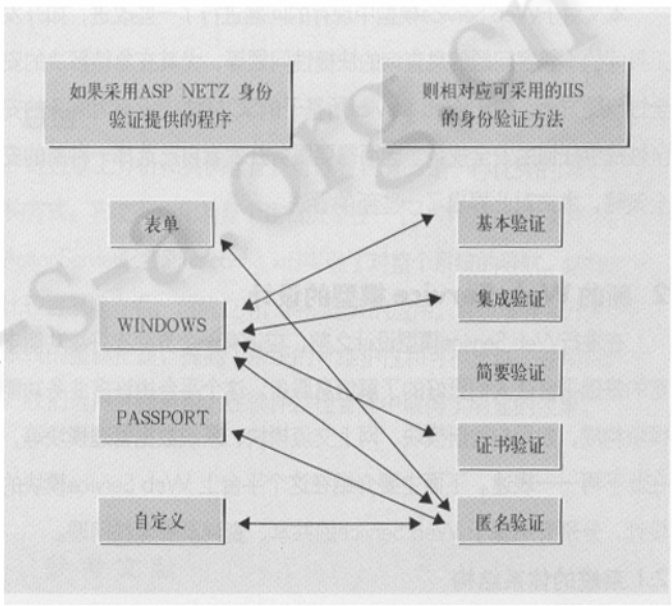


图 5　IIS 与 ASP.NET 组合设置

由图可知，PASSPORT身份验证需要支付许可费用，WINDOWS身份验证又有自身局限性，因此决定自行组合，采用三层安全验证方式，在一层采用匿名认证的方式，第二层采用基于窗体的验证即表单验证的方式，并且在实践的基础上对表单验证技术提出两处新的改进。第三层采用应用安全机制即自定义安全认证，增加认证的灵活性强，此方法是基于数字签名技术与用户信任凭证双因素动态验证组合

的方法，可以有效地解决Web Service的安全性问题。

3 Web Service 的技术实现与改进

3.1 基于 Web Service 的信息共享

在数据库中，游标提供了一种对表中检索出的数据进行操作的灵活手段，而存储过程将查询语句封装到一起，使执行速度加快，减少了网络通信量，大大增强了效率和灵活性。因此我们在设计过程中充分利用游标与存储过程这两大优点，自定义了存储过程提取数据，并在存储过程中使用事务和游标，将所需的数据信息一次从数据库中取出，这样在加快提取速度的同时实现了数据的安全存取，而一旦发生错误，可利用事务来安全回滚，具体编程不再表述。

3.2 基于 Web Service 的信息查询

随着Web Service的数量越来越多，相关的多个Web Service的信息处理也存在相当多的问题，在此将主要解决的问题罗列如下：

(1) Web Service在不同数据库、不同编程语言下的实现技术应针对不同数据库分别采用Oledb\Sql连接方式。

(2) 由于信息的来源不同，有结构化的信息，也有半结构化的信息，因此可采用统一的XML数据模型，即采用XMLSCHEMA描述数组模式，采用XSL显示数据。

(3) 在研究过程中发现，Delphi数据集（clientdataset）和.NET 中的数据集(dataset)存在不兼容问题,为了解决这个问题,采用了XML作为中介,两者均向XML做转化。

在Delphi中为了保证XML和Clientdataset数据格式一致,必须使用XML映象器做两个格式转换文件(.xtr)即先产生XML tO Datapacket 的转换文件xtoc.xtr文件，再利用TXMLTransFormProvider组件通过xtoc.xtr文件将原始XML转换为ClientDataSet ,现在可以像使用数据库中数据那样使用XML文件中的数据了。

当对数据作修改时，还需要XML映象器产生Datapacket to XML的转换文件ctox.xtr,将TXMLTransFormProvider的TransformationWrite属性的子属性TransformationFile设为ctox.xtr文件，然后可调用ClientDataSet.ApplyUpdates方法将改动的数据存回XML文件，最后在主窗体的OnClose时间加上存盘指令，程序退出时会将所做的数据修改存入XML文件。

(4) 为了解决来自不同Web Service的不同格式的数据插入、合并、分页、排序等问题，可采用DOM技术。DOM是基于树结构的一种接口，DOM模型需要对整个XML文档进行扫描，然后解析生成一个对象树，XML文档中所有的标签和属性都是用对象来表示。当创建了DOM对象时，既可对树接点进行查找、编辑、新建等操作，从而实现数据的合并。

3.3 Web Service 的信息安全问题

采用三层安全机制来解决Web Service的安全问题时，在第二、三层分别提出了新的问题并作了相应的改进：

(1) 在第二层安全机制采用了基于窗体的验证（表单验证），对此方法作了两处改进：

① 通常的表单验证方法中，数据库中存放的是用户密码的原文，验证时把用户的帐户名和密码同数据库中的数据比较，如一致则通过验证。这样，一旦有人进入数据库所在服务器，用户信息便一览无余。改进的方法是，将数据库独立于Web Service器，且在IIS中设置为直接接收来自服务器IP地址的访问；数据库存放的是注册时经加密算法计算过的用户密码，在选择算法时，采用具有不可逆性的算法，即使有人闯入数据库也无法还原用户密码；用户登录时，输入已申请的帐号、密码，由登录模块对密码加密，判断是否同数据库中密码一致，一致方可进入。通过这样的步骤，系统在并不知道用户密码的明码情况下就可以确定用户登录系统的合法性。这样不但可以避免用户的密码被具有管理员权限的用户知道，而且还在一定程度上增加了密码被破解的难度。

② 同时，可设定用户的有效在线时间，一旦超时，则自动下线，重新登录方可进入。这样，即使有人离开时忘了注销，系统也会在短时间内帮助用户注销；系统还可在用户个人工具箱中设置注销链接，方便用户使用。

(2) 一般系统所用的口令是静态口令，它是一种单因素认证方法，通常采用如下方式：当用户需要访问系统资源时，系统提示用户输入用户名和口令。系统采用加密方式或明文方式将用户名和口令传送到认证中心。并和认证中心保存的用户信息进行比对。在这种情况下，网络和系统登录控制通常使用的口令是静态的，也就是说在一定时间内固定不变的，而且可重复使用。若他们知道用户的密码，就可冒用用户的身份登录系统或网络，进行非法操作等行为。

对此现象，可采用双因素认证方式，所谓双因素认证方式即在单因素（固定口令）认证基础上结合第二个物理认证因素，以使认证的确定性按指数递增。在此，提出了第二种认证因素信任凭证。此信任凭证根据用户的网卡的地址、登录时间等生成，且在一定时间后自动更新。在信任凭证中采用了GUID,GUID是根据机器上网卡的地址加上时间、机器重启次数、随机数等因素按一定的算法生成，在一定时间的范围内，全世界唯一，此方法比较新颖，国外一些刊物上已提出，我们也是处于初步研究阶段。

如aaaaaaaa-bbbb-bbbb-cccc-dd-dd-dd-dd-dd-dd-dd-dd

其中aa...代表32位长的随机数，bb代表时间戳，cc与机器重起次数有关，dd是一个6字节长的字符串，一般是网卡地址，没有网卡的

是一个常数。

(3) Web Service是需要实现一个模式已确定请求客户的信任凭证。理论上在Web Service中可以传递用户帐户和密码给每一个方法，但是使用此方法有一个弊病，即对同一Web服务的后继调用需要用户重新登录；针对此情况，将面向对象的方法用于设计之中，即创建一个继承System.Web.Services.Protocols.SoapHeader的类，这个类将成为后来的SOAP Header，它将被传递到Web Service，其中包含用来验证用户的所有信息，而且利用SOAP头发送验证信息能够确保Web服务代码简洁，能够允许客户程序一次设置SOAP头信息之后反复使用。

(4) 对于加密算法采用了非对称加密。因为对称加密虽然速度快，但对称加密本身有一些弊病，例如对称加密容易受到中途拦截窃听的攻击，因而不适用于数字签名技术，而且密钥的个数大约是以参与者数目的平方的速度增长，因而很难将它的使用扩展到大范围的人群中；而非对称加密因为不必发送密钥到接收者，所以它不用担心密钥被中途拦截的问题，因而支持数字签名技术。而且要分发的密钥的数目与参与者的数目一样，这样，在与参与者数目很大的情况下，非对称密码技术仍然会工作得很好。

(5) 自定义认证的实现原理如下：

一般用户注册服务用户身份时，系统随机根据非对称加密算法产生一对密钥，公钥提供给注册服务用户，私钥系统保存，客户程序根据用户信息和消息摘要函数生成不可逆的消息摘要，然后对消息摘要生成数字签名，并利用公钥对用户信息及附加的数字签名加密，客户程序最后将加入信息发往服务器。服务器端用私钥解密信息后，拆分出用户信息和数字签名，并根据用户信息和相同的消息摘要函数生成消息摘要，将数字签名还原成消息摘要，然后判断两者是否一致，如不一致，则拒绝，这样可以有效解决阻塞问题；如一致，服务器端随机生成一个唯一的信任凭据，然后把这个信任凭据加密后发送给客户程序，同时在数据库中保存信任凭据以及它的生成时间。此后，客户程序解密信任凭据，在每一个请求中把重新加密的信任凭据通过SOAP协议的头元素发送给服务器端。对于每一个客户程序发送的请求，服务器端提取客户端传送加密的信任凭据后用私钥对其解密，然后同数据库中原来保存的相应用户的信任凭据去比较，判断客户程序提供的信任凭据是否正确，是否过期，根据检查结果判断用户是否已经通过身份验证。

在实践中已经具体实现了此认证方法：首先在Web端定义SOAP头，即创建一个由SoapHeader派生的类，加入必要的公用成员：

```
public class AuthenticationHeader :SoapHwader
{public string AuthenticationTonen
{get
{return_AuthenticationTonen;}
Set
{_AuthenticationToken=value;}
}}
```

然后，声明该标头类的实例，并在WebMethod中加入标头，设定标头的值来自客户端，且SOAP消息中的标头为必需：

```
public AuthenticationHeader AuthHeader;
[WebMethod]
[SoapHeader("AuthHeader",Direction=SoapHeaderDirection.In,
Required=true)]
```

客户程序登录

```
//客户端利用代理调用服务端的login函数，此函数在数据库中检查用户信息及其数字签名的合法性。
head=new SoapHeaderClient.localhost.SoapHeader_class();
//如果用户名字和数字签名是合法的，login方法随机生成一个唯一的信任凭据（Guid类型），并把这个凭据和当前时间利用存储过程保存到数据库，以后用它确定键是否已经过期，然后把这个凭据加密后返回客户端。
Guid authToken=Guid.NewGuid();
SqlCommand cmd=new SqlCommand("InsertAuthToken",dataConn);
cmd.CommandType=CommandType.StoredProcedure;
SqlParameter paramAuthToken=new SqlParameter("@AuthToken",
SqlDbType.int);
paramAuthToken.value=authToken;
```

客户端发送SOAP头

接下来客户端解密凭据，在每一个后继请求中用SOAP Header发送重新加密的凭据。

```
head.AuthenticationHeaderValue.Authentication token=encry
(_authToken)
```

服务器端处理SOAP头

服务器端从SOAP请求提取出SOAP头，并根据数据库中此用户的凭据来检查SOAP头中平局是否正确，是否过期。如果正确，提取服务并更新凭据的使用时间。

```
if Validate(null,null,AuthHeader.AuthenticationToken);
```

这样，客户程序就可以保留和使用代理的实例，且不必每次都重新登陆或设置SOAP头信息。利用SOAP头发送验证信息能够确保Web服务的代码简洁，能够允许客户程序一次设置SOAP头信息后反复使

下转第 24 页 >>

上接第 35 页

用，而且因为SOAP头对立于SOAP体，所以即使WEB方法升级，此认证方法仍有效。

4 总结

本文在研究Web Service技术以及相关技术的基础上，对现存的Web Service技术进行了多方面的改进，提出了新的Web Service的应用模型。在服务的共享方面，利用Web Service技术与脚本技术的结合降低了整个系统的复杂度，高效地实现了基于服务的信息共享。

参考文献

1 Peter A,bromberg,Professional ASP.NET Web Service [M][WROX] Chapter 5 SOAP FormatChapter 6 Custom SOAP techniques, SOAP Extensions。

2 Web Service Using Microsoft .NET and C# [M] Reliable Software, IncModule 1: Chapter 3 .NET Types and C#Module 3: Chapter 2 SOAP Encoding。

3 C# Web 高级编程[M]，清华大学出版社，2002.8,Chapter 11 .NET安全与密码技术,Chapter 12 作为应用程序插件的Web服务。

4 ASP.NET在数据库开发中的应用[M]，电子工业出版社，2002.3。

5 [美]邢什 公钥基础设施（PKI）实现和管理电子安全[M]， 清华大学出版社，2002.12。