文章编号:1005－0523(2007)02－0113－05

用 WSE 3.0 实现 Web 服务安全中的签名与加密

胡晓红,丁振凡

(华东交通大学 信息工程学院,江西 南昌 330013)

摘要:签名和加密是实现 Web 服务安全中的常用技术.Web 服务安全开发工具包 WSE 3.0 的推出,为 Web Services 安全问题提供了新的解决方案.本文给出了用 WSE 3.0 实现部分签名和加密的具体过程,具有广泛的实用性.

关 键 词:Web 服务安全;WSE 3.0;签名;加密
中图分类号:TP393　　**文献标识码**:A

1 引言

如果没有良好的安全性,Web 服务就不能发挥它的作用.Microsoft 和 IBM 以及其他公司共同制定了一组提供 Web 服务安全性的规范,其中最重要的是 WS－Security.现在,WS－Security 规范系列在很大程度上已日臻完善.Microsoft 发布 Web 服务安全性开发工具包(Web Services Enhancements)的 3.0 版(WSE 3.0),主要目标是为开发人员提供 WS－＊ 规范的第一个完整实现,用来开发符合 WS－Security 规范的消息级安全性解决方案.它大大简化了保护 Web 服务安全的过程.

2 WSE 所支持的安全特性

WSE 要保护 Web 服务安全,主要内容就是要使用 WS－Security 等规范实现 SOAP 消息安全.可以分为以下两个方面:

1) 完整性:通过对 SOAP 消息进行数字签名以及接收方验证该签名,让 SOAP 消息的接收方可以检查 SOAP 消息在传递过程中是否被修改.WS－Security 标准要求数字签名遵循 XML 数字签名标准.该标准支持对 SOAP 消息的选定部分进行签名.默认情况下,所有 WS－Addrssing SOAP 标头、SOAP 正文,以及 Security 标头中的时间戳都要由发送方签名.

2) 保密性:通过加密 SOAP 消息,使消息发送方可以确保 SOAP 消息内容仅对预定的接收方可见.WS－Security 要求对 SOAP 消息的加密遵循 XML 加密标准,该标准支持对 XML 文档的选定元素进行加密.默认情况下,WSE 中的标准断言只对 SOAP 消息的正文加密,这是最常见的方案.

但在很多实际情况下,并不需要对整个 SOAP 消息的正文进行签名和加密,而只是想对其中某一部分内容进行签名和加密,这可以通过向 SOAP 消息添加一个自定义 SOAP 标头来实现.将要处理的内容从正文中分离出来,并在自定义 SOAP 标头中定义,这样,对 SOAP 消息部分内容的签名和加密,就转变为对自定义 SOAP 标头的签名和加密了.

收稿日期:2006－10－25
作者简介:胡晓红(1975－),女,江西进贤人,华东交通大学硕士研究生,研究方向:电子商务,Web Services.

3 WSE 3.0 策略框架

在 WSE 3.0 中,实现所支持的安全特性是基于改进的 WSE 3.0 策略框架.WSE 3.0 使用策略来创建整个管道.如图 1 所示,一个策略包含了一个有序的策略断言列表.每个策略断言定义一个对 Web 服务的要求.并在策略编译过程中,由每个策略断言生成的筛选器负责对进入和离开终结点的 SOAP 消息进行截获和处理,来执行对 Web 服务的要求.策略断言生成筛选器遵循策略中断言的顺序.而管道表示了这个输入筛选器和输出筛选器的集合.

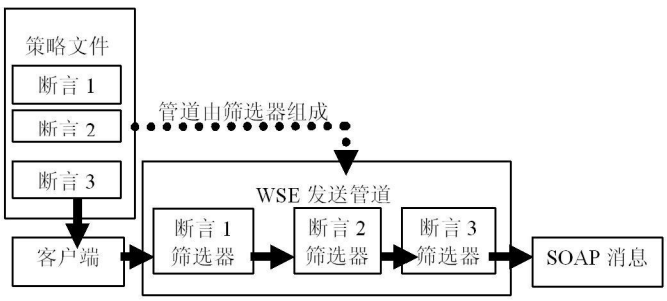

图 1 WSE 3.0 发送管道中的策略

WSE 3.0 支持一组不同的内置安全断言,这是 WSE 3.0 中最重要的新增安全功能.新的安全断言不再像 WSE 2.0 中的断言那样,将重点放在低级别完整性和保密性的模块上,实现消息级别的安全要求.相反,它们将重点放在较高级别的完备安全方案上,实现操作级别的安全要求,并代表了当今最佳做法(如相互证书身份验证).

但是,内置安全断言并不完全支持为自定义的 SOAP 标头指定安全需求,这就需要创建自定义安全断言类,创建自定义安全断言中要使用的自定义筛选器类,并实现在管道中插入自定义筛选器,在策略中使用断言类,然后将该策略应用于服务.

在 WSE 3.0 中,正是策略框架驱动着管道.这在下面阐述的对 SOAP 消息部分签名和部分加密的实现中得到了充分体现.

4 具体实现

以下结合笔者使用 WSE 3.0 开发的商品销售系统中的应用,介绍 Web 服务安全实现的思路及过程.在实现顾客订购商品业务中,顾客通过网上平台(由商家提供 Web 服务实现业务)获得商品基本信息和商品价格后,才会决定是否购买该商品.一旦选择购买,在形成订单并进行款项统计时,网上平台需要确保此时商家所提供的商品价格具有不可抵赖性.这就产生了 Web 服务的安全需求,即需要商家对商品价格进行数字签名.为此,将商品的价格用一个自定义 SOAP 标头包含,由商家对该自定义 SOAP 标头进行签名和加密.具体实现过程如下:

4.1 WEB 服务端设置

在 Web services 的服务端,为了实现对自定义 SOAP 标头的操作.首先需要创建一个从 SoapHeader 类的派生类,并通过添加公共属性,来表示自定义 SOAP 标头中的内容.要对自定义 SOAP 标头进行数字签名,这个标头必须要有一个 Id 属性,并且 Id 属性值必须是唯一标识符.这要通过应用 System.Xml.Serialization.XmlAttributeAttribute 来指定,而其值则通过类的构造函数来获得,并保证其唯一性.类的简要定义如下:

```
public class ProductPriceInfo:SoapHeader {
  private string id;private string productid;private string productprice;
  public ProductPriceInfo(string ProductId) {
    this.ProductId = ProductId;id ="Id—"+Guid.NewGuid().ToString();}
    //设置 AttributeName 属性和 Namespace 属性
```

```
[XmlAttribute("Id", Namespace ="http://docs.oasis-open.org/…utility-1.0.xsd")]
public string Id { get { return id;} set { id = value;} }
…}
```

4.2 对 Web 服务方法的设置

在 Web 服务代理类中声明一个自定义 SOAP 标头类的成员变量，使代理类中包含该标头类信息. 并在 Web 服务方法被调用前，设置该成员变量的 SoapHeader 特性中的 MemberName 属性，将该类成员信息形成所发送消息的标头，并在请求和响应消息中都存在(默认只是请求时存在)，使 Web 服务方法被调用时能响应处理该标头信息.

```
public Product PriceInfo myHeaderMember;
[WebMethod]
  [SoapHeader("myHeaderMember", Direction = SoapHeaderDirection.InOut)]
  public Product GetProductInfoById(string Id) { … }
```

4.3 客户端方面的设置

在创建服务代理类对象后，再创建 SOAP 标头类对象，并将该 SOAP 标头对象加到服务代理类对象中的 SOAP 请求消息标头中，这样，在调用服务代理类的方法后，就会在发出的请求消息中附加该 SOAP 标头信息. 示例如下：

```
proxy.ProductPriceInfoValue = mySoapHeader;
```

4.4 对自定义断言类的设置

创建一个自定义策略断言类，并在该类对应的服务端输出 SOAP 过滤器类中，重载 SecureMessage 方法实现对自定义 SOAP 标头的数字签名和加密. 在方法中，首先需要分别创建一个用于签名和加密 SOAP 消息的安全令牌的新对象，并将用于签名的安全令牌添加到消息头 security 区中，然后用该令牌创建一个消息签名类对象. 接着，在消息头中根据元素名和命名空间值查找每一个元素，找到自定义标头元素后，获得其 ID 值，用它为自定义 SOAP 标头创建一个签名引用，再加到消息签名元素中. 最后，将修改好的消息签名元素加入 security 区中. 这就实现对该自定义标头的签名. 若这时需要对该自定义标头进行加密，则可以用加密令牌通过获得的 ID 值针对该自定义标头进行，然后将加密信息添加到消息头 security 区中. 这就实现了对该自定义标头的加密. 该方法的简化示例如下：

```
public override void SecureMessage(SoapEnvelope envelope, Security security) {
  …//获取 token 令牌用来签名，tokenb 令牌用来加密
  security.Tokens.Add(token);
  MessageSignature sig = new MessageSignature(token);
  //创建一个为 ProductPriceInfo 的限定名对象
  XmlQualifiedName header = new XmlQualifiedName("ProductPriceInfo", "http://tempuri.org/");
  for(int index = 0;index < envelope.Header.ChildNodes.Count;index++) {
    XmlElement child = envelope.Header.ChildNodes[index] as XmlElement;
    if (child != null && child.LocalName == header.Name && child.NamespaceURI == header.Names-
pace) {
      //若找到该自定义标头，则通过创建一个签名引用来实现对其的签名
      SignatureReference soapRef = new SignatureReference("#" + child.Attributes[0].Value);
      soapRef.AddTransform(new Microsoft.Web.Services3.Security.Xml.XmlDsigExcC14NTransform());
      sig.AddReference(soapRef);
      //用 tokenb 令牌对该自定义标头进行加密
      security.Elements.Add(new EncryptedData(tokenb, "#"+ child.Attributes[0].Value));
      }
  }
```

```
    security.Elements.Add(sig);
}
```

5 实现结果

在 Web 服务端处理业务后，形成的要发送的响应消息中，还未经签名和加密处理的自定义 SOAP 标头的简要格式如下所示：

```
<soap:Header>
    <ProductPriceInfo p5:Id="Id-0d8bacd7-b24e-447a-bf20-c5bcc98760e4"…>
      <ProductId>1111</ProductId>…
      /Product PriceInfo>
</soap:Header>
```

经自定义策略类中的服务端输出 SOAP 过滤器进行部分签名和加密处理后，生成的消息头简要格式如下所示：

```
<soap:Header>
  <ProductPriceInfo p5:Id="Id-e45ddc0a-1dfe-4329-bf4a-a07961cc17be"…>
    <xenc:EncryptedData Id="Enc-9fa5f9e0-d6a9-469c-99ee-af202dbc1431"…>
    </xenc:EncryptedData>
  </ProductPriceInfo>
  <wsse:Security soap:mustUnderstand="1">
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <Reference URI="#Id-e45ddc0a-1dfe-4329-bf4a-a07961cc17be">
        <DigestValue>4JFIvAJ+AoRn5ZDhXYVZpjh0F2g=</DigestValue>
      </Reference>
    </SignedInfo>
   </Signature>
   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:ReferenceList>
      <xenc:DataReference URI="#Enc-9fa5f9e0-d6a9-469c-99ee-af202dbc1431"/>
    </xenc:ReferenceList>
   </xenc:EncryptedKey>
 </wsse:Security>
</soap:Header>
```

通过两个消息头的比较，可以看到，自定义标头中的信息已经被成功加密，而针对它的签名信息也已经存在于 Security 区的 SignedInfo 元素中.

6 结束语

WSE 3.0 作为新推出的工具，在实现 Web 服务安全方面提供了更加简便、更加全面的设计方案. 在该工具的支持下，我们通过 WSE 3.0 策略框架来设计安全架构，快速实现实际应用中的安全需求，并较好地解决了以前设计中总是将实现安全的代码与实现业务的代码一起混在 Web 方法中而造成麻烦的问题. 使安全设计人员能专心根据安全需求进行设计，实现了系统的业务逻辑与安全逻辑的分离. 当然 WSE3.0 本身也在不断发展，还需要我们进一步研究，不断完善其具体的实现功能.

参考文献：

[1]Web Services Enhancements 3.0 for Microsoft.NET[S].http://www.microsoft.com/downloads/details?aspx?familyid=018a09fd-3a74-43c5-8ec1-8d789091255d&displaylang=en,2005-12.

[2]于国良，韩文报.XML 的安全性研究[J].信息工程大学学报，2006,7(1):7-10,22.

[3]Web Services Enhancements(WSE)3.0 的新功能[EB/OL].http://msdn2.microsoft.com/en-us/library/ms977317.aspx,2005-6.

[4]Protect Your Web Services Through The Extensible Policy Framework In WSE 3.0 [EB/OL].http://msdn.microsoft.com/msdnmag/issues/06/02/WSE30/,2006-2.

[5]张维勇，程俊，王建新.基于 WS-Security 安全规范的 Web 服务设计[J].合肥工业大学学报(自然科学版)，2006,29(8):972-975.

Implementing Signature and Encryption of Web Services Security with WSE 3.0

HU Xiao-hong，DING Zhen-fan

(School of Information Engineering，East China Jiaotong University，Nanchang 330013，China)

Abstract：Signature and encryption are the technique often used in implementing Web services security.The issuance of Microsoft Web Services Enhancements(WSE)3.0 supplies a new solution to Web Services security.The paper explores the implementation ways of partial signature and partial encryption by using WSE 3.0，which has wide-spread practicability.

Key words：Web Services Security；WSE 3.0；signature；encryption

(上接第 112 页)

Realization of Embedded Secure Web Server Based on Linux

YANG Feng-ping，XING Jian，MA Shu-yan

(School of Electrical and Electronic Engineering，East China Jiaotong University，Nanchang 330013，China)

Abstract：The embedded Web Server here is based on processor ARM9 and operating system Linux.Firstly，the paper presents its whole design，then it expatiates three modules of the Web Server：double processes module，CGI module and secure mechanism module.

Key words：embedded Web Server；process；CGI；security authentication