Web服务的安全性研究

祝伟华 周 颖 杨 丹
(重庆大学软件学院 重庆400044)

摘 要 WS-Security是一个受到广泛支持的保证Web服务安全的建议规范,本文首先介绍了Web服务的安全性需求,然后简要描述了WS-Security规范,进而根据此规范阐述了安全性需求的实现,最后概述了Web服务安全性的应用实现,这些可以作为今后相关研究工作的框架。

关键词 Web服务,安全性,加密,签名,验证,WS-Security

Security Study on Web Service

ZHU Wei-Hua ZHOU Ying YANG Dan
(College of Softwaer,Chongqing University,Chongqing 400044)

Abstract WS-Security is a wide-supported specification on ensuring Web service security,this article introduces the security requirement on Web service and the actual WS-Security specification,thus it explains the implementation of the specification and the applications of the specification; it provides a framework for further application and research.

Keywords Web service,Security,Encryption,Signature,Authentication,WS_Security

1 引言

当今应用Web服务技术、实施供应链管理、电子商务的企业在试图把它们的流程和其商业伙伴、客户和供应商的流程集成的时候,面临着各种各样的风险。尽管Web服务技术有助于简化它们的内部业务流程,例如,把业务流程、供应链以及客户关系集成在一起,并使其自动化和标准化,但Web服务技术并不直接提供保护企业资产所必需的安全性机制。

Web服务要成为各种业务处理、系统和产品的异类互联的基本结构,安全性是至关重要的。如果参与者不能确定消息交换的完成,很多业务问题就不可能得以解决。如果没有一种Web服务标准来提供可靠的消息传送,则各种应用程序只能根据其业务逻辑来实现必需的功能,这种要求给负责业务逻辑的开发人员带来了沉重的负担,并且妨碍了互操作性。

本文中涉及的Web服务的安全性的三个需求包括:验证(authentication)、数据完整性(data integrity)和数据机密性(data confidentiality),以下将简要地概述一下这些功能:

1)验证用来确保业务事务中的各方确实是他们所声称的;可以通过提交用户ID和密码来获得身份证明,还可以使用由信任的认证机构(Certificate Authority)(比如Verisign)签发的X.509证书获得身份证明。X.509证书中包含身份凭证,并且具有一对与之相关联的私钥和公钥。由一方提交的身份证明包括证书本身和一条使用证书的私钥签名的信息,通过确认与证书相关联的公钥签名的信息,接收方可以验证发送方是证书的所有者,从而确认他们的身份。在双方彼此验证时,称作相互验证(mutual authentication),这种验证通常是在Web服务客户和Web服务提供者之间完成的。

2)为了确认业务信息的数据完整性,可以使用安全性密钥对数据进行数字签名。如使用发送方的X.509证书的私钥对Web服务请求的SOAP Body进行数字签名,也可以签署请求中的SOAP Head代码块,以确保在业务上下文范围以外的事务中交换的信息的完整性。

3)安全性的三个需求的第三个是机密性。可以利用加密技术来使Web服务请求和响应中交换的信息不可读,确保访问传送中的、内存中的或已经持久化的数据的任何人都将需要适当的算法和安全性密钥解密数据才能访问实际的信息。

2 Web服务安全规范(WS-Security)概述

WS-Security[1]是一个受到广泛支持的保证Web服务安全的建议规范,现在它已被结构化信息标准促进组织(OASIS)批准为正式标准。WS-Security规范提出了一套在创建安全Web服务时用于实现消息内容完整性和机密性的简单对象访问协议扩展,解决的是如何在多点消息路径中维护一个安全的环境,它适合多种安全模型和加密技术。

WS-Security定义了一个用于携带安全性相关数据的SOAP标头元素[2]。如果使用XML签名,此标头可以包含由XML签名定义的信息,其中包括消息的签名方法、使用的密钥以及得出的签名值。同样,如果消息中的某个元素被加密,则WS-Security标头中还可以包含加密信息(例如由XML加密定义的加密信息)。WS-Security并不指定签名或加密的格式,而是指定如何在SOAP消息中嵌入由其他规范定义的安全性信息。WS-Security是一个基于XML的安全性元数据容器的规范。

除了利用现有的消息身份验证、完整性和加密外,WS-Security指定了一个通过UsernameToken元素传输简单用户凭据的机制。此外,为了发送用于加密或签名消息的二进制令牌,还定义了BinarySecurityToken元素,在此元素中可以存储关于调用方消息的签名方法和加密方法的信息。WS-Security将所有安全信息保存在消息SOAP部分中,从而为Web服务安全性提供了端到端的解决方案。

但是,WS-Security只是一种构件,并不提供完整的安全性解决方案,它必须与其它Web服务扩展和更高级的特定于应用程序的协议联合使用,以适应多种安全性模型和加密技术。

在本文中,我们将了解如何使用WS-Security和其他配合工具在SOAP消息中嵌入安全机制,以解决WS-Security

所涉及的三个方面问题，即身份验证、签名、加密。

3 安全性需求的实现方案

通过使用 Web 服务安全性技术，我们可以有选择地实现安全性的三个需求中的每个需求。需要另一个应用程序的服务的应用程序在本文中被称为消费应用程序(Consuming Application)。本身提供服务的应用程序称为服务提供者(Service Provider)。图1说明了这种关系，

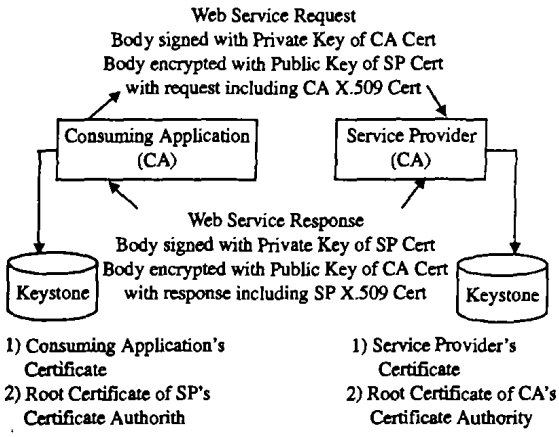

图1 系统环境

如图1所示，在调用服务提供者的服务之前，服务提供者的 X.509证书的公钥必须同客户端交换，并配置在客户端的运行时中，签发消费应用程序和服务提供者双方证书的认证机构(Certificate Authority)(比如 Verisign)的根证书需要导入到本地的密钥存储容器中。消费应用程序的密钥存储容器将包含服务提供者的证书的根证书。同样地，服务提供者的密钥存储容器将需要消费应用程序的证书的根证书。上述要求是强制性的，使之可以对单个证书的数字签名(作为 SOAP 消息中的二进制令牌传送)进行确认。

这三个安全性需求的完整实现将包括：

1) Web 服务的发送方进行请求或响应，包括：A)使用它们的 X.509证书的私钥来签署消息；B)使用接收方的 X.509证书的公钥来加密消息，以便确保只有它们才能访问消息内容。

2) Web 服务的接收方进行请求或响应，包括：A)使用发送方的公钥来检验消息的签名，以便验证发送方并确认消息的完整性；B)使用它们的 X.509证书的私钥来解密消息。

为了不必显示全部 XML 命名空间声明，使用了以下 XML 命名空间[3]说明 Web 服务安全性规范(WS-Security) XML 加密的细节：

命名空间	说明	命名空间 URI
Xs	XML 架构	http://www.w3.org/2001/XMLSchema
Wsse	WS-Security	http://schemas.xmlsoap.org/ws/2002/07/secext
Wsu	实用程序元素	http://www.w3.org/2001/XMLSchema
Soap	SOAP 元素	http://schemas.xmlsoap.org/soap/envelope/

图2 XML 命名空间

3.1 验证

验证用户身份时，用于传递调用方凭据的最常见方法之一是使用用户名和密码，这是一项用于 HTTP 基本身份验证和简要身份验证的技术。UsernameToken 元素里包含用户名和密码。如果希望以更安全的方式发送密码，可以发送它的简要散列。密码简要散列是随机内容、创建时间与密码的组合，随机内容的长度是16字节，以 base64编码值的形式传递，其工作原理是客户端使用这些信息加上密码来创建密码散列。接收方通过获取其中的密码并再次创建散列来验证此数据，如果结果一致，则表明密码正确。但这种保护方式不能防范重复攻击。如果使用这种保护方式，应该包括一个 wsu:Timestamp 标头，其中包含一个足够小的时间，用以提供创建时间值和过期时间值；然后，对消息中的 wsu:Timestamp 元素进行签名，以便可以检测到对时间戳的任何篡改。否则，攻击者可能使用完整的 UsernameToken 攻击 Web 服务。为防范重复攻击，还需要包含一个机制，用于跟踪传入消息的某个唯一特性。这种机制需要将此特性保存到缓存中，并且至少持续到消息超时。

验证用户身份的另一个选择是发送 X.509证书。可以使用 PKI 将此证书映射到应用程序中的现有用户。使用证书来验证自己时非常容易受到重复攻击的破坏，因此，最好强制消息发送方同时使用其私钥来签名消息。当消息发送 X.509证书时，将在一个名为 BinarySecurityToken 的 WS-Security 令牌中传递此证书的公共版本，证书本身以 base64编码数据的形式发送。使用证书的私钥创建的签名虽然可以证明客户端是该证书的合法拥有者，但是这种消息可以被重复，因此必须在 wsu:Timestamp 标头中说明消息在被忽略之前的有效时间。

Kerberos 也是很常见的验证方式，用户需要提供一组凭据(例如用户名/密码或 X.509 证书)，如果所有内容检验合格，安全系统将授予用户一个 TGT。TGT 是一个隐藏的数据，用户无法读取，但必须提供它才能访问其他资源。通常，用户需要提供 TGT 来获得服务票据(ST)。系统的工作方式如下：

1)客户端到密钥发行中心(KDC)验证身份，并被授予一个 TGT。

2)客户端获取 TGT 并使用它来访问 Ticket Granting Service(TGS)。

3)客户端为访问特定网络资源而请求一个 ST，然后 TGS 向客户端提供 ST。

4)客户端向网络资源出示此 ST，并使用 ST 指示的权限访问资源。

Kerberos 包含客户端向服务证明身份以及服务向客户端证明身份的机制。ST 只适用于访问一个网络资源，并可用于发现调用方的身份。当在消息中提供 Kerberos 票据时，需要将该数据加密复制到消息中。

3.2 签名

签名后的消息几乎无法被篡改，但不能禁止外部查看消息内容。使用签名，SOAP 消息的接收方可以知道已签名的元素在路由中未发生改变，从而可以确定：

·由 X.509证书、UsernameToken 或 Kerberos 票据标识的用户已对消息进行了签名。

·签名后的消息没有被篡改。

SOAP 消息签名[4]有几种方法，每种方法都提供了一个可用于签名消息的密钥。X.509允许发送方使用他们的私钥来签名消息；Kerberos 提供了一个由发送方创建并在票据中传输的会话密钥，只有此消息的预期接收方可以读取票据，发现会话密钥并验证签名的可靠性；最后，可以使用密码对

UsernameToken 进行签名。

签名是使用 XML 签名生成的，如果内容发生改变，签名便不再匹配。签名时中间方可以向 wsu:Timestamp 中添加一个 wsu:Received(说明特定的中间方收到消息的时间)元素，每当元素发生改变时，都需要更新签名，否则将出现异常。

3.3 加密

仅仅证明消息发送方的身份和表明消息未经更改仍然不够。如果通过签名的纯文本来发送信用卡号或银行帐号，攻击者实际上可以验证没有其他攻击者更改过消息的内容，因此他们可以十分肯定数据是有效的。为此需要加密数据，使得只有预期的接收方才能够阅读消息。

加密数据时，可以选择使用对称加密或不对称加密。对称加密需要一个共享密钥。也就是说，加密消息与解密消息使用的是同一个密钥。如果同时控制两个端点并且可以信任使用密钥的用户和应用程序，则可以使用对称加密。

如果需要使用简单的分布式密钥来发送数据，则可以使用不对称加密。X.509证书允许使用不对称加密。接收数据的端点可以公布它的证书，并允许任何人使用公钥来加密信息，只有接收方知道私钥。因此，只有接收方可以得到加密的数据并将其重新转换为可读内容。

当执行不对称加密时，消息的接收方需要知道私钥才能解密消息。公钥的交换必须提前进行。

4 Web 服务安全性的应用实现

依据以上理论，下面概述的 Web 服务安全性的实现使用的是 X.509证书，是很多客户解决方案中常见的一种实践。

4.1 消费应用程序处理 Web 服务请求

消费应用程序(Consuming Application)一般有一个服务代理(Service Proxy)，当调用 Web 服务时，代理或客户端系统上的 SOAP 运行时会在发送请求之前执行 Web 服务安全性功能。

首先，对 SOAP 消息进行数字签名。SOAP 运行时可以访问 keystore(密钥存储容器)以检索所需的安全性密钥和证书，检索后可以仅对 SOAP 主体进行签名，也可以对 SOAP Body 内部的单个元素进行签名，另外，还可以对 SOAP 头代码块进行签名。签名是利用消费应用程序的 X.509证书的私钥来执行的。一旦消息被签名，X.509证书本身就作为二进制安全性令牌包含进 SOAP 头了。消息是使用带有共享密钥的对称算法进行加密的，而用于数据加密的密钥是使用带有与服务提供者的 X.509证书相关联的公钥的非对称算法进行加密的。考虑到服务提供者可能会使用多个证书，一旦消息和共享密钥被加密，服务提供者(请求正发送给它)的 X.509证书的引用必须包含进 SOAP 头。

4.2 服务提供者处理 Web 服务请求

当服务提供者接收到 Web 服务请求时，基于请求者的 URL(所发布的服务访问端点)，请求就定向到 SOAP 处理引擎(SOAP 运行时)。在请求中传送的消息数据和共享密钥是加密的，所以第一步就是识别 SOAP 头中引用的 X.509证书，并从 keystore 中检索它的私钥，一旦获得私钥，就可以使用非对称算法来解密共享密钥。通过公开的共享密钥，就可以使用对称算法来解密消息数据。现在，通过公开的整个消息，就可以访问在 SOAP 头中传送的 X.509证书来检索它的公钥。消息的数字签名是利用消费应用程序的公钥来执行的。服务提供者的 SOAP 运行时(runtime)不仅确认消息的完整性，而且还确保消费应用程序确实对消息进行了签名。这一过程也验证了消息的来源/发送方，因为只有拥有这个证书的发送方才可以访问私钥。一旦消息被解密且确认了签名，SOAP 运行时(runtime)就可以调用 Web 服务的实现。

4.3 服务提供者处理 Web 服务响应

执行了服务实现的业务逻辑并且获得了一个响应后，Web 服务安全性操作在 Web 服务的响应消息上进行。服务提供者的 SOAP 运行时使用它自己的 X.509证书的私钥对消息进行数字签名。证书包含在 SOAP 消息中，并且使用共享密钥来加密这个消息。用于数字加密的密钥可能是在最初请求中传送的同一密钥，也可能是另一个随机生成的密钥。共享密钥的加密是使用在请求中传送的证书的公钥来执行的；因而，只有已经访问了证书的私钥的请求发送方才可以解密这个消息。一旦消息被签署和加密，服务提供者的 SOAP 运行时就发送响应给消费应用程序。

4.4 消费应用程序处理 Web 服务响应

消费应用程序对 Web 服务响应的处理与服务提供者对请求执行的处理非常相似。消费应用程序接收到 Web 服务响应后，基于最初的 HTTP 会话，响应定向到 SOAP 处理引擎(SOAP 运行时)。在响应中传送的消息数据和共享密钥是加密的。因此，最初的步骤是检索与对应的请求相关联的证书的私钥，以便使用非对称算法来解密共享密钥。通过公开的共享密钥，就可以使用对称算法来解密消息数据。在整个消息公开以后，就可以访问 SOAP 报头中传送的 X.509证书，以便检索它的公钥。响应消息的数字签名是使用服务提供者的公钥来执行的。在签名成功确认之后，消费应用程序的 SOAP 运行时不仅确认消息的完整性，而且还确保服务提供者确实对消息进行了签名。这个过程也验证了消息的来源/发送方，因为只有拥有这个证书的发送方才可以访问用于对消息签名的密钥。一旦消息被解密并且确认了签名，SOAP 运行时就将响应发送给消费应用程序。

结束语 正如以上所论述的，Web 服务安全性非常复杂，并且有非常多的应用。希望我们目前依赖的所有公用服务，例如安全性、事务、存储、查询、路由、进程协调等等，将来都能够直接映射至 Web 服务体系结构；到那时，我们就能够真正地以无缝方式将任意两种服务耦合在一起。

参 考 文 献

1 Web Services Security (WS-Security) Version 1.0 April 5,2002 IBM,Microsoft&Verisign. http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

2 SOAP Security Extensions: Digital Signature. http://www.w3.org/TR/SOAP-dsig/

3 Namespaces in XML. http://www.w3.org/TR/1999/REC-xml-names-19990114/

4 XML-Signature Syntax and Processing. http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/

5 Basiura R,Batongbacal M. Professional Asp.net Web Services : 398～413