

## Web 服务提供方安全模型的设计、建模与分析

刘洪燕,段振华,张鹏飞

(西安电子科技大学 计算机学院,陕西 西安 710071)

**摘要:**随着 Web 服务应用的迅速发展,Web 服务提供方的安全问题已成为制约其实际应用的主要障碍之一。文中着重讨论了 Web 服务提供方面临的安全问题,引入了一种基于安全策略与实现分离的信息安全解决模型,并根据 Web 服务的实际情况进行了改进,方便了 Web 服务提供方动态地制定安全策略,灵活地实现策略指导下的各种安全措施。最后,采用投影时序逻辑(PTL, Projection Temporal Logic)形式化描述了该模型中资源访问决策部分的规范说明。

**关键词:**投影时序逻辑;Web 服务;安全策略;安全中介

**中图分类号:**TP309

**文献标识码:**A

**文章编号:**1673-629X(2006)11-0162-04

## Design, Modeling and Analysis of a WS Provider Security Model

LIU Hong-yan, DUAN Zhen-hua, ZHANG Peng-fei

(School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

**Abstract:** With the rapid development of the Web services applications, the Web services security issue has emerged as one of obstacles in business applications. In particular, the security issue of Web services providers becomes more important than other parts. Discusses the security issues faced by Web services providers, and proposes an information security solving model. With this model, the security policy and its implementation are separated. In addition, to facilitate the security designers of Web services providers to understand and analyze the model, describes the specification of the resource access decision model using projection temporal logic (PTL).

**Key words:** projection temporal logic; Web services; security policy; security intermediary

## 0 前言

Web 服务是一种新型的耦合分布式计算模式<sup>[1]</sup>,企业将其核心业务逻辑与数据资源封装成 Web 服务,通过发布服务提供给用户进行远程调用。Web 服务的用户在注册中心发现服务后,根据提供方对服务的描述(采用服务描述语言 WSDL),通过简单对象接入协议(SOAP, Simple Object Access Protocol)完成跨平台的交互与服务调用。目前,Web 服务已成为充分利用各种网络资源的一种新途径<sup>[2]</sup>。

目前,Web 服务安全的研究方法主要是将一些现有的安全技术嵌入到 Web 服务的框架中,例如数字签名(Digital signature)、XML 加密技术和标准、访问控制(Access control)技术等。对于 SOAP 消息传输的安全,WS-Security 规范<sup>[3]</sup>将基于 XML 的多种安全规范和一些其它

的规则(如 X.509 和 Kerberos 票据等的编码规则)有机地嵌入到 SOAP 扩展消息头中,改善了 SOAP 消息传输的保密性与完整性。此外,还有一些研究致力于从体系结构上改善 Web 服务的安全性,提出 Web 服务安全体系结构<sup>[4]</sup>等方法。

文中在 Web 服务的整体框架内为服务提供方的安全问题设计了一个模型(如图 1 所示),并选取其中的资源访问决策给出形式化的规范说明与上层的抽象实现。其中规范说明的描述采用投影时序逻辑(PTL, Projection Temporal Logic)。

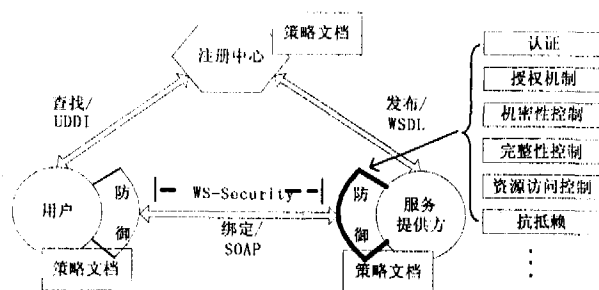


图 1 Web Service 中的安全问题

收稿日期:2006-01-15

**基金项目:**国家自然科学基金重点项目资助(60433010);国家自然科学基金面上项目资助(60373103);教育部博士点基金项目资助(20030701015)

**作者简介:**刘洪燕(1981-),男,湖南娄底人,硕士研究生,研究方向为信息安全、逻辑描述与验证;段振华,教授,博导,研究方向为逻辑描述与验证。

## 1 PTL 简介

区间时序逻辑(ITL, Interval Temporal Logic)<sup>[5-8]</sup>是

一种具有离散的时间模型的时序逻辑,可以方便地处理并行系统。PTL<sup>[8]</sup>是对区间时序逻辑的扩展,它把原来只用来表达有限状态的 ITL 扩展到无穷状态,同时增加了过去时序操作符和新的投影操作。投影操作  $(p_1, \dots, p_m) \text{ prj } q$  可以作为并行计算和投影计算的复合,避免了因使用“与”操作符在共用状态变量时可能相互影响的情况。PTL 有很好的规范描述和证明技术,在对系统的需求进行形式化描述以及系统性质验证上有很大的优势。下面简要介绍 PTL 的语法,PTL 的语义及其它相关知识请参见文献[6~8]。

设  $\Pi$  是命题的可数集合,  $V$  是有类型定义的静态变量和动态变量的可数集合。在区间内其值保持不变的变量称为静态变量,否则为动态变量。项 (terms) 和公式 (formulas) 由下列语法给出:

$$e ::= x \mid u \mid \bigcirc e \mid \bigodot e \mid \text{beg}(e) \mid \text{end}(e) \mid f(e_1, \dots, e_m)$$

$$p ::= \pi \mid e_1 = e_2 \mid P(e_1, \dots, e_m) \mid \neg p \mid p_1 \wedge p_2 \mid \exists x: p \mid \bigcirc p \mid \bigodot p \mid (p_1, \dots, p_m) \text{ prj } p$$

这里  $x$  是一个动态变量,  $u$  是一个静态变量,在  $f(e_1, \dots, e_m)$  和  $P(e_1, \dots, e_m)$  中,  $f$  是函数,  $P$  是谓词。在此假定项的类型与  $f, P$  中的参数类型是一致的。如果一个公式 (项) 不含任何时序操作符 (i.e.  $\bigcirc, \bigodot, \text{beg}(\cdot), \text{end}(\cdot), \text{prj}$ ), 就称它为状态公式 (项); 否则就称它是时序公式 (项)。

## 2 服务提供方安全问题的解决模型

图 2 是 Web 服务提供方的安全解决模型,如其左侧所示, SOAP 消息经过安全传输协议传送到服务提供方后,由 Server Stub 负责 SOAP 消息的解析,最后由 Service Implementation 来完成服务的实现。目前 Web 服务的安全研究大量地关注下层的传输安全(虚线部分所示),认为只要消息安全的传输给服务提供方后,其余工作将全部由服务提供方接手,这无疑加重了 Service Implementation 的负担。此外,由于 Web 服务的动态特性,不同的 Web 服务存在不同的安全需求,所对应的安全策略也不一样,服务提供方在开发 Web 服务时不仅要依据 WS-Security 规范,同时还要实现一套满足自身安全策略的安全措施。

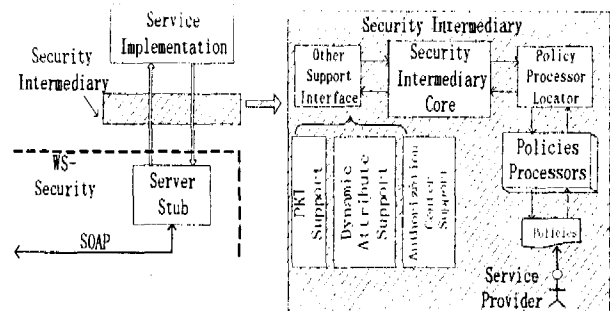


图 2 Web 服务提供方的安全解决模型

用户和服务提供方采用 SOAP 协议进行消息传递,发

送方将服务调用请求封装成 SOAP 消息后进行发送,接收方在取得 SOAP 消息后进行解析便可获得相应内容<sup>[9]</sup>。就服务提供方而言,在 Server Stub 将用户调用请求交给 Service Implementation 之前,仍有机会对不合理的请求进行阻隔。这层屏障被称为 Security Intermediary(SI)。

由于不同的服务提供方对安全的要求不一样,所制定的安全策略自然也会不一样。如果根据不同的策略采用不同的实现模型,可能会使 SI 不具备通用性,从而使得每一个服务提供方都需要对 SI 进行设计与编码,这实际上退化成了将 SI 的安全功能交由 Service Implementation 来完成,于是加重了服务提供方的实现难度。为了既能帮助不同的服务提供方选用不同的安全策略,又能将 SI 与 Service Implementation 相互分离,引入一种基于策略与安全实现分离的信息安全解决模型<sup>[4]</sup>,并考虑 Web 服务的实际情况对其进行了改进。

如图 2 右侧所示,这是一个 SI 实现模型,其中 SI 核心(SIC, SI Core)负责判断服务调用请求是否满足安全策略;策略处理器定位者(PPL, Policy Processor Locator)选出相应需要判断的策略,在此 PPL 需要策略处理器库(PP, Policies Processors)的支持,PP 又需要策略库(Policies)的支持;其它支持接口(Other Support Interface)辅助 SIC 进行判断,提供一些判断所需要的支持信息,如 PKI 支持(SIC 在进行认证鉴别时用到的一些公钥密码等将由 PKI 支持来提供)、系统动态属性支持等。

SI 根据它处理的安全策略类型来进行分类,例如,处理认证相关的策略则称为认证 SI,也可以是授权 SI、资源访问决策 SI 等,或者是几种 SI 的综合。服务提供方根据实际的需要来选择和配置这些 SI。SI 进行判断的典型情况为: SIC 接收到 Server Stub 的服务调用请求,根据服务调用请求向 PPL 取得相应的策略处理器,这些处理器将反映出相应的安全策略,随后, SIC 运算这些策略处理器,此时若需要其它的信息,将从其它支持接口获取。运算结束后,若 SIC 从 PPL 中取得的所有策略处理器都能够满足, SIC 将服务调用请求发送给 Service Implementation,否则发送请求被拒绝的回执给 Server Stub。

## 3 SI 的建模

当前,安全系统的建立普遍存在一些问题,如设计上,很多安全标准与规范都是用自然语言给出的(像 WS-Security 规范<sup>[5]</sup>等)。自然语言规范说明虽然容易阅读和理解,但其缺陷在于不严格,容易让人产生误解,使得最后的实现往往违背了设计人员的意图。此外,在安全系统实现后,安全性质的验证多采用测试方法,而测试方法缺少精确的形式语义,其验证结果常令人怀疑。

本节将应用资源访问决策(Resource Access Decision, RAD)SI,形式化描述其规范说明,规范说明选用 PTL 逻辑。其它的 SI 可采用类似的形式化描述和实现手段。若服务提供方需要多个功能的 SI 时,可以进行叠加。

### 3.1 RADSI 系统介绍

RADSI 的功能是对服务访问请求进行访问控制,将满足访问控制策略的服务调用请求提交给服务实现(Service Implementation),并阻止不满足访问策略的请求。这里假定访问控制策略已由服务提供方按一定规则编写好,并存入了策略库。

整个 RADSI 系统的组成部分如图 3 虚线框内所示。

\* 资源访问决策核心部分(RADC, Resource Access Decision Core)。Service Stub 将服务访问请求发送给 RADSI, RADSI 做出判断后将决策结果发送给 Service Implementation。

\* 策略处理器定位部分(PPL, Policy Processor Locator)。该部分负责向 RADC 提供策略处理器。

\* 动态属性服务部分(DAS, Dynamic Attribute Support)。该部分在 PP 进行运算时提供系统动态属性, DAS 读取系统中一些状态信息。

\* 策略处理器库为 PPL 提供支持,同时策略处理器需要策略库/表的支持。

\* 策略库/表由服务实现方按一定规则进行编写,它将反映出服务实现方所要求的安全政策。

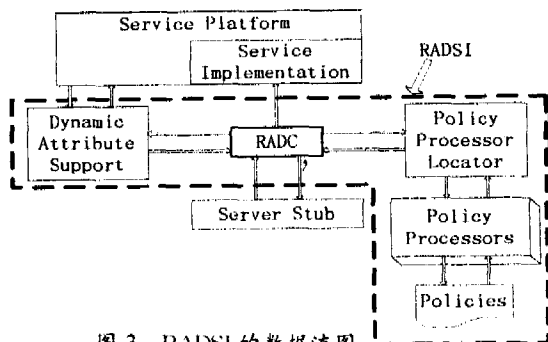


图 3 RADSI 的数据流图

### 3.2 RADSI 系统各组成部分的接口说明

在这里主要考虑 RADC, PPL 和 DAS 三个部分。

\* RADC 部分为整个 RADSI 系统的控制中心。它和 RADSI 系统的外部 and 内部都存在调用接口,能够发送与接收消息。

外部:提供一个服务访问请求递交接口供 Server Stub 使用;向 Service Implementation 提交访问请求;向 Server Stub 送回抛弃标志;

内部:向 PPL 发出取得策略处理器的请求;向 DAS 发出取得当前系统动态属性的请求。

\* PPL 部分向 RADC 提供策略处理器,它提供接口供 RADC 发来取得策略处理器的请求,并向 RADC 发回与请求相对应的策略处理器。

\* DAS 部分为 RADC 提供实时的系统动态属性,以供策略处理器在运算时做参考。它提供接口供 RADC 取得系统动态属性。

### 3.3 RADSI 系统的形式化规范说明

#### 3.3.1 RADC 的运行与闲置

RADC 发现 Server Stub 递交服务访问请求 Find-req,

当 Find-req 成立时 RADSI 系统开始运行。将请求提交服务实现 Send-acc-req 或发回抛弃标志 Send-den-dec 后, RADSI 系统进入闲置,并约定在一个请求没有处理完毕时, RADSI 暂不响应其它请求。RADC 系统定义如下:

$$RADC\_sys \stackrel{df}{=} ((halt(Find\_req) \wedge RADC\_idle); RADC\_run)^+$$

RADC\\_idle 为 RADC 系统处于闲置状态, RADC\\_run 为运行状态。

RADC 的闲置状态须保证不对 PPL 和 DAS 进行调用。定义为:

$$RADC\_idle \stackrel{df}{=} keep(\neg Ask\_pp) \wedge keep(Ask\_da)$$

Ask\\_pp 和 Ask\\_da 分别为向 PPL 发送取得策略处理器请求和向 DAS 发送取得系统动态属性请求。

RADC 的运行状态则又由等待策略处理器状态 Wait\\_pp, 策略处理器运算状态 Cal\\_pp, 处理器结果综合状态 Com\\_res 和发送决策结果组成,但若运行过程中发生任何错误将导致发回抛弃标志。定义为:

$$RADC\_run \stackrel{df}{=} trap(any\_error); (Wait\_pp; Cal\_pp; Com\_res); ((dec\_acc \rightarrow Send\_acc\_req) \wedge ((dec\_den \vee any\_error) \rightarrow Send\_den\_dec))$$

trap(c):p 表示 p 执行时,若 c 为真,则 p 停止执行。

$$trap(c):p \stackrel{df}{=} (halt(c) \wedge prefix(p) \vee (prefix(halt(c)) \wedge p))$$

其中 dec\\_acc 表示 RADC 得出允许访问的决策, dec\\_den 表示决策结果为拒绝访问服务。any\\_error 表示 RADSI 可能发生的任何错误。

等待策略处理器状态 Wait\\_pp 的行为是发出取得策略处理器请求后,等待 PPL 的返回,但若 T\\_ppl 时间后仍没有结果则停止等待。因此 Wait\\_pp 可描述为:向 PPL 发出取得匹配策略的请求,并等待策略的返回,但仅等待 len(T\\_ppl) 长的时间。定义如下:

$$Wait\_pp \stackrel{df}{=} Ask\_pp \wedge shorter(halt(Get\_pp), len(T\_ppl))$$

shorter(p, q) 表示 p 先执行完时,则 q 停止执行; q 先执行完时,则 p 停止执行。

$$shorter(p, q) \stackrel{df}{=} \exists E, E': \{halt(E \vee E') \wedge prefix(p \wedge halt(E) \wedge prefix(q \wedge halt(E)'))\}$$

策略处理器的运算状态 Cal\\_pp 的行为是对所取得的策略处理器进行分析,而在分析过程中有可能会参考系统的动态属性。因此 Cal\\_pp 由两个状态来描述:等待系统动态属性状态和策略处理器的分析状态。但如果发生 DAS 返回结果的错误,则停止策略处理器的运算,所有匹配中的策略都处理完毕也会结束该状态。定义如下:

$$Cal\_pp \stackrel{df}{=} trap(das\_ret\_error \vee all\_cal\_cd); ((cal\_pp\_ing; Wait\_da)^+)$$

cal\_pp\_ing 表示 pp 正在进行运算。all\_cal\_ed 表示所有取得的策略处理器均已运算完毕。Wait\_da 为等待系统动态属性,其行为类似 Wait\_pp,它向 DAS 发送取得系统动态属性的请求,得到返回信息则结束,否则等待,但仅等待 T\_das 时间。定义为:

$$\text{Wait\_da} \stackrel{\text{df}}{=} \text{Ask\_da} \wedge \text{shorter}(\text{halt}(\text{Get\_da}), \text{len}(\text{T\_das}))$$

Ask\_da 为 RADC 发出取得系统动态属性的请求,Get\_da 为 DAS 返回所请求的系统动态属性。T\_das 为 RADC 所能等待系统动态属性的最长时间。

策略处理器运算结果综合状态 Com\_res,其动作为:当所有的策略处理器的运算结果均为允许访问时,置 dec\_acc 成立,否则置 dec\_den 成立。Com\_res 的行为定义为:

$$\text{Com\_res} \stackrel{\text{df}}{=} (\text{all\_pp\_acc} \rightarrow \text{dec\_acc}) \wedge (\neg \text{all\_pp\_acc} \rightarrow \text{dec\_den})$$

all\_pp\_acc 表示所有的策略处理器的运算结果为允许访问服务。定义 RADC 所取得的策略处理器集合为 pps,所取得的策略处理器的数目为 |pps|,pp\_acc(k) 表示策略处理器 k 的运算结果为允许访问服务。all\_pp\_acc 定义如下:

$$\text{all\_pp\_acc} \stackrel{\text{df}}{=} \forall k \in \text{pps}; \text{pp\_acc}(k)$$

### 3.3.2 PPL 的运行与闲置

PPL 的功能为提供匹配的安全策略,安全策略已假定存储在策略库中,并已编写了策略处理器在 PP 中。PPL 系统的运行情况为:PPL 一直等待 RADC 的取得策略处理器请求,当有该请求时,在 T\_pp 时间内返回相应的策略处理器集合。当完成一次运行后,又进入闲置,等待下一次 RADC 发来的询问请求。其行为可以形式化定义如下:

$$\text{PPL\_sys} \stackrel{\text{df}}{=} (\text{Get\_pp} \wedge (\text{halt}(\text{Ask\_pp}) \wedge \text{PPL\_idle}; (\text{Get\_pp} \text{ within } \text{len}(\text{T\_pp})))^+$$

b within p 表示条件 b 在 p 运行的区间内成立,定义为: b within p  $\stackrel{\text{df}}{=} \neg \text{first}(p \wedge \neg \Diamond(b))$

PPL\_idle 为 PPL 处于闲置状态,即保持不发送任何策略处理器。定义为: PPL\_idle  $\stackrel{\text{df}}{=} \text{keep}(\neg \text{Get\_pp})$

### 3.3.3 DAS 的运行与闲置

DAS 的行为类似 PPL,等待 RADC 的询问系统动态属性请求 Ask\_da,当该请求为真时,在 T\_das 时间内返回所请求的系统动态属性值。提供完 RADC 所询问的属性值后,进入闲置状态,等待下一次 RADC 发来的询问请求。定义为:

$$\text{DAS\_sys} \stackrel{\text{df}}{=} \neg \text{Get\_da} \wedge (\text{halt}(\text{Ask\_da}) \wedge \text{DAS\_idle}; (\text{Get\_da} \text{ within } \text{len}(\text{T\_das})))^+$$

DAS\_idle 为 DAS 处于闲置状态。定义如下:

$$\text{DAS\_idle} \stackrel{\text{df}}{=} \text{keep}(\neg \text{Get\_da})$$

Get\_da 为 DAS 返回系统动态属性的值。

## 4 结束语

文中所给出的 Web 服务提供方的统一安全解决模型着重解决服务调用的 SOAP 请求安全到达后,服务提供方存在的一系列安全问题。该模型的优点在于将安全策略的制定与服务调用请求是否满足安全策略的检查进行了分离,充分考虑了 Web 服务的动态特性,为不同的服务提供方制定不同的安全策略,并为今后的更新策略提供方便。此外,还给出了该模型规范的形式化描述,形式化描述的规范说明可以方便服务提供方的安全设计人员准确理解该模型。该模型同样还可适用于有安全需求的 Web 服务用户。

不过,模型在实现时仍存在一些不足:如安全策略必须按一定的格式进行编写,安全策略高效匹配算法的研究以及如何将一些复杂的安全功能整合到其它支持接口中等。下一步将开发 SI 的实现平台,该平台包括以下功能:安全策略的半自动化生成、安全策略的匹配、各种安全服务支持的调用接口、方便按规定格式编写安全策略以及各种 SI 的开发等。

## 参考文献:

- [1] Ogibuji U. The Past, Present and Future of Web Services[DB/OL]. 2001-09. <http://www.webservices.org/index.php/article/articleview/663/1/61>.
- [2] Pierce M, Fox G, Youn C, et al. Interoperable Web Services for Computational Portals[C]//Supercomputing'02. Baltimore, Maryland: [s. n.], 2002: 1-12.
- [3] Specification: Web Services Security (WS-Security)[EB/OL]. 2004. <http://www106.ibm.com/developerworks/library/ws-secure>.
- [4] Yi D, Wang J. An Approach for Modeling and Analysis of Security System Architectures[J]. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(5): 1099-1119.
- [5] Mozskowski B. Executing Temporal Logic Programs[M]. Cambridge, UK: Cambridge University Press, 1986.
- [6] Cau A, Zedan H, Coleman N, et al. Using ITL and Tempura for large-scale specification and simulation[C]//Parallel and Distributed Processing. [s. l.]: [s. n.], 1996: 493-500.
- [7] Duan Z, Koutny M. A Framed Temporal Logic Programming Language[J]. JCST, 2004, 19(3): 314-351.
- [8] Duan Z, Yang X, Koutny M. Semantics of Framed Temporal Logic Programs[C]//Proceedings of ICLP 2005, LNCS 3668. Barcelona, Spain: Springer Verlag, 2005: 256-270.
- [9] Geer D. Taking Steps to Secure Web Services[J]. Computer, 2003, 36(10): 14-16.