面向 Web 服务资源的两层访问控制方法

霍远国 马殿富 刘 建 李竹青

(北京航空航天大学计算机学院 北京 100191)

摘 要 Web服务资源具有静态的Web服务接口和动态的有状态资源两个组件。针对这两个组件的不同特征为它提出一种基于属性的两层访问控制方法(Two Level Attribute-Based Access Control,2L-ABAC)。2L-ABAC扩展基于属性的访问控制模型(Attribute-Based Access Control,ABAC),对这两个组件分别进行访问控制。ABAC系统的访问决定依赖于用户提供的主体属性,所以2L-ABAC采用策略发布机制告知用户所需的属性,并根据各层特征分别采用WSDL附件和元数据交换两种发布方式。除了分层设计带来的灵活性,2L-ABAC还继承了ABAC模型的特性,能够对来自其他安全域的用户进行访问控制。另外,它基于相关国际规范实现,如XACML和SAML,故具有通用性。

关键词 Web服务资源,基于属性的访问控制,WSDL,XACML,SAML

中图法分类号 TP393.08 **文献标识码** A

Attribute-based Two Level Access Control for Web Service Resources

HUO Yuan-guo MA Dian-fu LIU Jian LI Zhu-qing

(School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

Abstract Web Services Resource (WS-Resource) consists of static Web service interface and dynamic stateful resource. According to the different characteristics of the two components, we proposed an Attribute-Based Two Level Access Control (2L-ABAC) on for WS-Resources. Attribute retrieval is essential for ABAC systems because they are based on their decisions on attributes of users, so 2L-ABAC employs access control policies publishing mechanism to inform users of the needed attributes. Access control policies of Web Services are static and those of resources are dynamic, correspondently two publishing methods, WSDL attachment and metadata exchanging, are adopted for each level respectively. 2L-ABAC inherits from the ABAC model the capability of authorizing unknown users from other security domains, besides its flexibility due to the hierarchy design model. Moreover, this architecture can be implemented by extending the standard specifications such as XACML and SAML, so it has broad applicability for WS-Resource based systems.

Keywords WS-Resource, ABAC, WSDL, XACML SAML

1 引言

Web服务技术可以在组织间共享程序模块,但它是无状态的。Web服务资源框架(Web Services Resource Framework,WSRF)在Web服务框架中对有状态资源进行建模,将其与Web服务结合起来形成Web服务资源(WS-Resource),实现了有状态的Web服务[1,2]。这种分布式程序组件给访问控制带来了新的问题:共享的对象包含数据和程序模块,要求一种细粒度的、环境相关的访问控制来保证程序的正确执行;Web服务资源在组织间共享,所以必须能够对来自不同安全域的未知用户进行访问控制[3,4];更重要的是,Web服务接口和有状态资源都是被保护的对象,前者是静态的,和应用系统具有相同的生命周期,而后者是被动态创建和消除的,生命周期由其所有者控制,访问控制机制须兼顾这两者,并且和它们的特点及生命周期相一致。

现有的访问控制机制不能很好地解决这些问题。基于身份的访问控制(Identity-Based Access Control,IBAC)和基于角色的访问控制(Role-Based Access Control,RBAC)不适合跨安全域的情景,因为它们是针对身份明确的用户。基于属性的访问控制(Attribute-Based Access Control,ABAC)的访问判定是基于请求者和其他实体(如对象、环境、动作)的主体属性的,而非身份或标识。属性即实体和访问控制相关的特征或信息,包含在属性证书(Attribute Certificate)中。属性证书是属性权威(Attribute Authority)颁发的(经过数字签名,有效性可被验证),断言用户具有某种属性,可用于陌生人之间传递信息,所以ABAC适用于开放环境的跨安全域情景[5,6],但它们目前只应用于无状态的Web服务。研究者针对特定的系统提出了有状态Web服务的访问控制[7],但它不具通用性。

本文基于ABAC模型和可扩展访问控制标记语言规

到稿日期:2009-08-18 返修日期:2009-11-09 本文受863国家重点基金项目"高可信软件生产工具及集成环境"(2007AA010301)资助。

霍远国(1985—),男,硕士,主要研究方向为Web服务中间件技术,E-mail:yuanguo.h_001@yahoo.com;马殿富(1960—),男,教授,博士生导师,CCF常务理事,主要研究方向为网络环境的系统软件核心技术、基于先进网络的下一代因特网的视频会议系统;刘 建(1982—),男,博士,主要研究方向为Web服务中间件技术等;李竹青(1985—),男,博士,主要研究方向为Web服务中间件技术等。

范[8]（eXtensible Access Control Markup Language，XACML），针对 Web 服务资源的两个组件提出基于属性的两层访问控制方法（Two Level Attribute-Based Access Control，2L-ABAC）。2L-ABAC 包含 Web 服务和有状态资源两个层次的访问控制，每个层次的访问控制和该组件的特点及生命周期相一致。ABAC 依赖于用户提供的主体属性[4,9]，所以 2L-ABAC 采用发布访问控制需求的方式告知请求者所需的主体属性，请求者据此在 SOAP 请求消息头中附带它们。根据各层特点采用不同的发布方式：Web 服务是静态的，其访问控制需求在服务发布时已经明确，所以采用 WSDL 附件的发布方式，即在 WSDL 中附带 Web 服务策略[10]（WS-Policy），其中描述访问控制需求；资源的访问控制伴随着资源被动态创建和消除，其需求无法附属于 WSDL，所以采用元数据交换[11]的发布方式。由于此类方法不针对特定系统，并且基于相关国际规范实现，因此对基于 Web 服务资源的系统具有通用性。

本文第 2 节讨论相关研究工作；第 3 节详细描述 Web 服务资源的两层访问控制框架；第 4 节讨论基于相关国际规范的 2L-ABAC 的实现；第 5 节测试 XACML 实现的性能并与 SunXACML[18]做对比分析；最后总结本文工作并展望未来目标。

2 相关工作

访问控制是安全领域的研究热点之一。SOA 被广泛用于跨组织的系统集成，要求能够对来自其他安全域的用户进行访问控制。IBAC 和 RBAC 只针对身份明确的用户，不能很好地适应这种开放环境。

基于角色的信任管理框架[12]扩展 RBAC，提出基于属性的用户到角色的自动映射机制，根据映射规则，把未知用户分配到已知角色，从而实现对未知用户的控制。但这种机制缺乏伸缩性，例如访问控制系统需考虑 m 种主体属性 A_i（$i=1,2,\cdots,m$），A_i 具有 $N(A_i)$ 种赋值，则需要建立并维护规模为 $\prod_{i=1}^{m} N(A_i)$ 的角色库。ABAC 适用于来自开放环境的未知用户[3,4,6,9]，其访问条件由基于各实体（主体、对象、环境及动作）属性的访问控制策略来描述，不但细粒度，而且具有伸缩性。Web 服务的 ABAC[6]描述了 ABAC 策略方程，并和传统的访问控制（IBAC，RBAC 等）对比，总结了 ABAC 对于 SOA 环境的优势。WSABAC[4]和带有信任协商的 ABAC[9]在 ABAC 模型的基础上引入信任协商机制，以获取主体属性。资源提供者和用户通过谨慎的、轮流的信任状暴露来逐步建立信任关系。协商过程由双方暴露非敏感的信任状开始，随着信任状的交换，建立起高级别的信任。这一过程可集成已有实现来完成，如 TrustBuilder[13]。然而协商过程耗时，并且对于失败的协商性能会更坏，因为往往需要经过多轮信任状交换才能发现协商不成功。另外，用户和资源提供者必须具有兼容的协商模块才能协商，增加了双方的相互依赖性。带有属性暴露限制的访问控制[3]扩展 ABAC，并允许用户制定属性暴露策略，规定向特定的服务暴露特定的属性。这要求用户事先明确未来要调用的所有服务，以对它们制定属性暴露策略，使得服务提供者和用户紧耦合。面向服务的访问控制[14]指出 Web 服务访问控制的 5 点需求，并提出一种基于逻辑的访问控制模型。它采用策略发布的方式向用户传递访问控制需求（告知所需的主体属性），但它和信任管理框架[12]一样，采用角色映射机制，缺乏伸缩性。

另外，上述方法都只针对无状态的 Web 服务。研究者[7]为有状态 Web 服务的访问控制提出了一种解决方案，但它只针对特定系统（高压电子显微镜数据网，HVEM）。国际标准化组织（如 W3C，OASIS）制定了一些与访问控制相关的规范[8,15,16]，并且其中的一些已被很多开源组织实现[17]，但它们只考虑了 Web 服务的一般特征。

3 Web 服务资源的两层访问控制方法

3.1 两层访问控制的原因

Web 服务资源由静态的 Web 服务接口和动态的有状态资源两个组件组成，两者都是访问控制保护的对象。基于以下原因，本文采用两层访问控制（服务层和资源层）：①两组件具有不同的生命周期，Web 服务接口和应用系统生命周期相同，而有状态资源被用户动态地创建和销毁。每个组件的访问控制须和该组件的生命周期一致。②同一 Web 服务可以和不同的资源结合，形成不同的 Web 服务资源。分层设计可以在服务层进行粗粒度的保护，在资源层进行细粒度的保护，管理方便。③分布式环境下，服务和资源往往属于不同的组织或个人，两层访问控制分别代表两方的安全需求。如博客系统，运行组织可以限制哪些用户注册或发表文章，而用户可以限制哪些用户阅读自己的文章。

3.2 两层访问控制框架

基于上述分析，本节描述基于属性的两层访问控制框架（2L-ABAC），如图 1 所示。

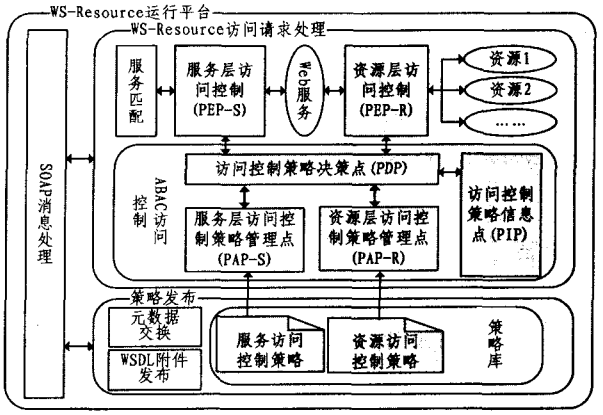

图 1 Web 服务资源的两层访问控制框架

2L-ABAC 包含服务层访问控制和资源层访问控制。前者保护 Web 服务接口免受非授权用户的访问，后者防止非授权用户读取或更改资源的状态。这两层访问控制都采用 ABAC 模型。ABAC 模型包含访问控制策略执行点（Policy Enforcement Point，PEP）、访问控制策略决策点（Policy Decision Point，PDP）、访问控制策略管理点（Policy Administration Point，PAP）及访问控制策略信息点（Policy Information Point，PIP）4 个组件以及访问控制策略（描述允许、拒绝规则）。PEP 根据访问请求构造决定权请求，发送给 PDP，然后接收并执行 PDP 作出的决定（允许或拒绝）；PDP 针对 PEP 的决定请求，评估访问控制策略决定允许还是拒绝访问；PAP 管理访问控制策略；PIP 管理环境属性及对象属性，并可能通过其他途径获取缺失的主体属性，如查询 LDAP、进一步协商

等。PIP 支持属性查询操作(lookup)。如图 1 所示,两层访问控制使用公共的 PDP,但有不同的访问控制策略(它们生命周期不同,并可能描述不同组织的安全需求),由不同的 PAP 管理。用户的 SOAP 请求消息在 SOAP 处理模块和服务匹配模块处理后,要经过两层访问控制,才能最终访问资源。访问控制策略由不同机制(WSDL 附件和元数据交换)发布。图 1 主要描述访问控制模块(阴影部分)而弱化了资源平台的其他功能。

3.3 访问控制模型

2L-ABAC 的每一层都采用 ABAC,ABAC 不是基于静态的访问权限定义,而是基于对各种实体(主体、对象、环境和动作)的属性的判定。ABAC 并不关心请求者是谁,而只关心他有哪些属性。2L-ABAC 继承了这一特征,所以适用于 SOA 环境。访问控制策略是关于实体属性的规则,描述"特定主体(即用户或其代理)在特定环境下对特定对象执行特定动作"的权限。下面具体描述 2L-ABAC 的访问控制模型。第 4 节介绍此模型的基于 XACML 规范[8]的实现。

(1)属性变量集

属性(Attribute):即实体的和访问控制相关的特征或信息。有 4 类实体属性:①主体属性(Subject Attribute,ATTR-S):主体是访问请求的发起者,即用户或其代理。主体属性指与访问控制相关的主体信息,如用户的组成员关系、职位、所属部门等;②对象属性(Object Attribute,ATTR-O):对象是被主体访问的实体,2L-ABAC 中,Web 服务和资源分别是两层访问控制中的对象。对象属性是与访问控制相关的对象信息,如服务的端口、资源标识及名字空间等;③环境属性(Environment Attribute,ATTR-E),指与安全相关的环境因素,如服务器系统时间、负载等;④动作属性(Action Attribute,ATTR-A),指主体请求对对象的访问动作,如读(r)、写(w)、读-写(r-w)、执行(exec)。

属性变量($Attr\text{-}V$):每个实体属性用一个属性变量表示。属性变量 $Attr\text{-}V$ 有一个定义域,记为 Domain($Attr\text{-}V$),表示该属性的取值范围。$Attr\text{-}V$ 取值 $value$ 记为 $Attr\text{-}V=value$,其中 $value\in$Domain($Attr\text{-}V$)。

属性变量集(Attribute Variable Set,简称 AVS):$AVS=\{Attr\text{-}V_i \mid i\in[1,n]\}$,为所有 4 类实体的属性变量组成的集合,$n=|\text{ATTR}-\text{S}|+|\text{ATTR}-\text{O}|+|\text{ATTR}-\text{E}|+|\text{ATTR}-\text{A}|$。

(2)访问控制策略

条件(Condition):Condition:$=\langle Attr\text{-}V\rangle\langle OP\rangle\langle value\rangle$,其中 $Attr\text{-}V\in AVS$ 为属性变量,$value\in$Domain($Attr\text{-}V$)为常量,OP 表示 Domain($Attr\text{-}V$)上的关系运算,如整数有">","<","="等。可见 Condition 为关于实体属性的谓词,若条件满足,则结果为 TRUE,否则为 FALSE。Condition.$Attr\text{-}V$ 表示 Condition 的属性变量为 $Attr\text{-}V$,其中点号表示所属关系,下文也按此约定。$Attr\text{-}V=value$ 时,Condition 的结果记为 Condition($value$)。

访问控制规则(Access Control Rule,ACR)为 ACR:Effect←$\text{Condition}_1\wedge\text{Condition}_2\wedge\cdots\wedge\text{Condition}_m$,$m\geqslant 1$。$\text{Condition}_i$ 为条件,$i\in[1,m]$;各条件的合取(即规则的右部)为规则前件。Effect 为规则后件,表示前件满足时达到的结果。Effect∈{Permit,Deny},Permit 表示授权,Deny 表示拒绝。当实体属性的值未知或不满足前件,规则的结果为 Indeterminate,这时 PEP 可以和用户进一步交互,获取必要的属性,也可直接拒绝用户访问。

例 1 访问控制规则"销售部(sales)的经理(Manager)在工作时间(9:00 至 17:00)有权执行(exec)'创建销售计划服务'(create_sales_plan)",可表示为 r_1:

Permit←$Sub\_Department=$'sales'$\wedge Sub\_Role=$'Manager'$\wedge Obj\_ServiceName=$'create_sales_plan'$\wedge Time>9{:}00\wedge Time<17{:}00\wedge Act\_ID=$'exec'

式中,$Sub\_Department$ 和 Sub_Role 为主体属性变量,前者表示主体所属部门,后者表示主体职位;$Object\_ServiceName$ 为对象属性变量,表示服务名字;$Time$ 为环境属性变量,表示系统时间;Act_ID 为动作属性变量,表示操作类型。

访问控制策略(Access Control Policy,ACP):ACP=$\{$ComAlg,Rule-Set$\}$。Rule-Set 为规则集合,ComAlg 为规则合并算法。规则合并算法把各个规则的运算结果合并起来,形成最终的策略结果。ComAlg 有允许/拒绝覆盖算法(一个规则运算结果为 Permit/Deny,则策略的结果为 Permit/Deny)等。

策略相关属性变量集(Policy Attribute Variable Set,PAVS):PAVS(p)表示访问控制策略 p 的所有规则所涉及到的属性变量集合。

例 2 只包含例 1 中 r_1 这条规则的策略 p 的相关属性变量集可表示为:PAVS(p)=$\{Sub\_Department,Sub\_Role,Obj\_ServiceName,Time,Act\_ID\}$。

(3)决定请求

决定请求(Decision Request,DR):$DR=\{Attr\text{-}V_i=value_i \mid i\in[1,t]\}$,表示一组实体属性变量的赋值,描述"特定主体在特定环境下对特定对象请求执行特定操作",是策略评估的输入。策略决定该请求是否允许。

请求相关属性变量集(Request Attribute Variable Set,RAVS):决定请求 r(包含 t 个属性赋值)的相关属性变量集,记为 $RAVS(r)=\{Attr\text{-}V_i \mid Attr\text{-}V_i=r.Attr\text{-}V_i,i\in[1,t]\}$。

(4)策略评估

策略评估是根据决定请求提供的属性赋值列表计算适用的访问控制策略来做出决定。首先逐个评估访问控制策略中的规则,然后调用合并算法将各个规则的结果进行合并,得到策略结果。亦可根据合并算法进行优化。如对于拒绝覆盖算法,遇到一个结果为 Deny 的规则,则策略的结果为 Deny,而无需继续计算。规则的评估算法如图 2 所示。首先从访问控制信息点 PIP 中查询缺失的属性赋值,若存在缺失属性 Attr 的值无法从 PIP 中查到,则返回 Indeterminate;然后逐个计算规则的各个条件的结果,若存在条件 Condition 的结果为 FALSE,表示规则的前件不满足,则返回 Indeterminate;若所有条件都满足,即满足规则的前件,则返回规则的后件,即 Effect。

例 3 $DR=\{Sub\_Department=$"sales",$Sub\_Role=$"Manager",$Obj\_ServiceName=$"create_sales_plan",$Act\_ID=$"exec"$\}$在 9:00 至 17:00 之间对于例 1 中规则 r_1 的评估结果为 Permit。

```
Alg1: ACREvaluation(dr, acr)
输入: 决定请求dr; 访问控制规则acr
输出: 规则的结果, Permit表示允许, Deny表示
      拒绝, Indeterminate表示实体属性不充分
      或不满足策略的前件。
1.  miss_attrs ← PAVS(acr) - RAVS(dr);
2.  if miss_attrs ≠ Ø then
3.    for each Attr-V in miss_attrs
4.          value = PIP.lookup(Attr-V);
5.          if value = null then
6.                return Indeterminate;
7.          endif
8.          dr = dR ∪ {Attr-V=value};
9.    endfor
10. endif
11. for each Condition in acr
12. for each Attr-V_i in dr
13.   if Attr-V_i = Condition.Attr-V then
14.         result ← Condition(value_i);
15.         break;
16.   endif
17. endfor
18. if result = FALSE then
19.   return Indeterminate;
20. endif
21. endfor
22. return acr.Effect;
```

图 2　访问控制规则评估算法

3.4　访问控制策略发布

如前所述，基于属性的访问控制依赖于用户提供的主体属性，属性证书是属性权威颁发的，断言用户具有某种属性。访问不同的资源，要求用户提供不同的属性证书。访问控制策略发布的目的就是告知用户访问特定服务或资源需要哪些属性证书。服务层访问控制策略在 Web 服务接口发布时已经明确，所以采用 WSDL 附件的方式来发布。资源层访问控制策略是资源所有者为其资源制定的，伴随资源本身被动态地创建或删除；并且同一组 Web 服务接口可以操作多个资源实例，所以各个资源实例的访问控制策略无法随 Web 服务接口发布，故采用元数据交换的发布方式。

```
<definitions xmlns=http://schemas.xmlsoap.org/wsdl/
xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy"
xmlns:ws-xacml="xacml-3.0-profile-webservices-spec-v1.0-wd-5">
 <wsp:Policy Id="policy1" >
  <ws-xacml:XACMLAuthzAssertion >
   < ws-xacml:Requirements >
    <xacml:Policy RuleCombiningAlgId="identifier:rule-combining-
          algorithm:deny-overrides">
     <xacml:Target/>
     <xacml:Rule RuleId="rule1" Effect="Permit">
      <xacml:Condition>
       <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
        <xacml:Apply
            FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
         <xacml:SubjectAttributeDesignator AttributeId="Sub_Department"/>
         <xacml:AttributeValue >sales</xacml:AttributeValue>
        </xacml:Apply>
        <xacml:Apply
            FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
         <xacml:SubjectAttributeDesignator AttributeId=" Sub_Role"/>
         <xacml:AttributeValue>Manager</xacml:AttributeValue>
        </xacml:Apply>
        .....
       </xacml:Apply>
      </xacml:Condition>
     </xacml:Rule>
    </xacml:Policy>
   </ws-xacml:Requirements >
  </ws-xacml:XACMLAuthzAssertion>
 </wsp:Policy>
 ......
<portType name="...">
 <operation name="create_sales_plan">
  <wsp:PolicyReference URI="#policy1" required="true"/>
  <input>...</input>
  <output>...</output>
 </operation>
</portType>
 ......
</definitions>
```

图 3　WSDL 附带访问控制策略

访问控制策略采用 XACML 规范[8]定义的策略语言来描述，如图 3 中的〈xacml:Policy〉元素片段所示。WS-Policy 规范[10]定义了一个基于 XML 的策略语法，用于描述和传递 Web 服务的需求(Requirements)和能力(Capabilities)。WS-Policy 由一系列的断言(Assertion)组成。策略框架定义了与 SOAP 消息认证、完整性和机密性相关的断言，但不包含对访问控制的支持[14]。而 WS-XACML 规范[16]定义了 ws-xacml:XACMLAuthzAssertion 授权断言，其中的 ws-xacml:Requirements 子元素可封装访问控制策略(xacml:Policy)。2L-ABAC 使用授权断言来扩展 WS-Policy，使它具有描述访问控制策略的功能。扩展的 WS-Policy 如图 3 中的〈wsp:Policy〉元素片段所示。

WSDL 附件发布方式：这种方式就是在 WSDL 发布中附带基于授权断言(ws-xacml:XACMLAuthzAssertion)扩展的 WS-Policy，其中包含 XACML 规范[8]描述的访问控制策略。WS-Policy-Attachment[19]定义了两种策略附属机制，把 WS-Policy 附属到它所适用的策略主体(Policy Subject)中，分别是策略主体定义内部附属机制和独立于主体定义的附属机制。另外，它还特别描述了 WSDL 附带 WS-Policy 的具体附属机制。这里采用这种机制把扩展的 WS-Policy 附属到 WSDL 中。

例 4　使用 xacml:Policy 描述例 1 中的访问控制规则，然后封装成 ws-xacml:XACMLAuthzAssertion 并集成到 wsp:Policy，最后附属到 WSDL 描述的 create_sales_plan 服务，如图 3 所示。

元数据交换发布方式：即通过元数据(mex:Metadata)的形式将资源的 WS-Policy 传递给用户。WS-MetadataExchange 规范[11]定义了元数据的请求和响应模式。如前所述，采用授权断言扩展的 WS-Policy 能够描述访问控制策略。

4　实现

本节描述基于国际规范的 2L-ABAC 实现。表 1 显示了 2L-ABAC 的主要模块所支持的规范。XACML 规范[8]定义了一个基于 XML 的策略语言，支持基于属性的访问控制。XACML 策略文档的根部是一个策略元素(xacml:Policy)或策略集元素(xacm:PolicySet)。策略集是一个可以包含策略及其他策略集的容器。如 3.3 节所述，策略元素包含规则集和一个合并算法。另外，为了优化性能，XACML 策略元素还包括一个目标(Target)子元素，用于描述该策略的适用范围。实现中使用 Target 建立策略的索引(以 Target 为键，以策略为值的反向索引)，以提高策略匹配的速度。除此之外，XACML 规范还描述了访问控制模型和各组件(PEP，PDP，PAP 和 PIP 等)之间的数据流(如 3.3 节(4)所述)。如 3.4 节所述，WS-XACML 规范[16]定义了关于访问控制的授权断言，用于扩展 WS-Policy[10]，使它能够集成访问控制策略。扩展的 WS-Policy 通过策略附属机制(WS-Policy-Attachment[19])附属到 WSDL 中，随其被发布。元数据交换发布方式指通过 WS-MetadataExchange 规范[11]定义的元数据请求与响应机制来发布扩展的 WS-Policy。用户的属性证书用 SAML[15](它是 OASIS 发布的安全断言标记语言规范)实现，其中定义的 saml:AttributeStatement 可封装主体属性。属性证书被添加到 SOAP 消息头部，访问控制系统使用这些属性信息可做出授权决定。

表 1　对国际规范的支持

2L-ABAC 的模块	国际规范定义的数据结构与机制	国际规范
访问控制策略模型	xacml:Policy	XACML[8]
基于 WSDL 附件的访问控制策略发布	wsdl:definitions ws-xacml:Requirements ws-xacml:XACMLAuthzAssertion wsp:Policy wsp:PolicyReference	WSDL WS-XACML[16] WS-Policy[10] WS-Policy-Attachment[19]
基于元数据交换的访问控制策略发布	mex:Metadata	WS-MetadataExchange[11]

带有属性证书的 SOAP 请求	soap:envelop saml:AttributeStatement	SOAP SAML[15]

图 4 描述了基于 XACML 规范[8]的访问控制模型(见 3.3 节)的实现。XACML 策略文件存储在磁盘中,为了避免反复加载它们(策略文件可能很多而不能全部常驻于内存),实现中 PAP 采用缓存的机制。缓存项淘汰算法为 LRU(Least Recently Used),即当缓存空间紧张时,淘汰最近最少使用的策略。缓存中为策略建立索引(以 Target 子元素为键的反向索引)。PDP 通过 Target 匹配可快速找到适用的策略。PDP 负责评估策略并做出决定,规则评估算法如图 2 所示。其中条件(Condition)是一个布尔类型的表达式,表达式的求值依赖于数据类型注册表和函数注册表。前者注册了 XACML 规范涉及到的数据类型,如 Java 类型、XML 类型及自定义类型等,后者注册的是各种运算符函数。XACML 规范定义了算术、逻辑以及高阶运算等 210 个函数,它们由唯一的 ID 标识。函数注册表以函数的 ID 为键,对应的值为该运算符的 Java 实现(通过 Java 反射机制获取)。系统启动时,注册表自动到指定目录下加载函数,实现类中的运算符函数。这种方式也方便用户自定义运算符(把实现类放入指定目录下即可)。策略评估引擎使用这两个注册表进行计算。PIP 管理着环境及资源的属性,PDP 评估策略时,可用来查询缺失的属性。实现中还预置了一个扩展点,便于引进其他机制(如查询 LDAP),提供其他实体属性的获取功能。PEP 解析 SOAP 请求中包含的 SAML 断言[15],从中可获取请求者提供的主体属性。然后获取目标对象、环境以及动作属性并构造决定请求发送给 PDP。最后根据 PDP 做出的决定拒绝或允许用户的访问。

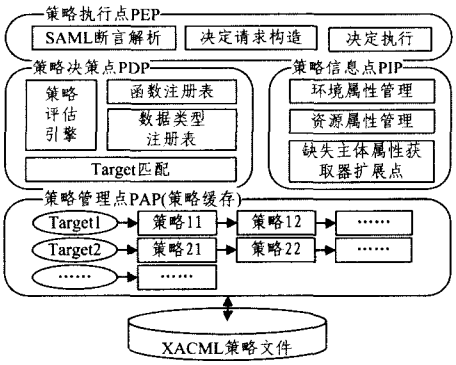

图 4 访问控制模型的实现结构

5 测试与分析

测试机 CPU 为 Pentium(R) D 3.40GHz,1GB 内存,操作系统为 Windows XP,JVM 内存设置为 40MB 至 256MB。实验中使用 5 个大小不同的访问控制策略库(分别包含 50,100,200,500,1000 个策略文件),每个策略包含 5 条规则和 1 个 Target(描述该策略适用的资源),策略文件大小为 6k~7k。

实验对比本文实现与开源实现 SunXACML[18]的性能。对于每种实现,以不同的策略库启动访问控制模块,并记录当前策略库加载的时间(表 2 中第 2 栏)。然后随机发起 500 个决定请求,访问控制模块对它们逐个做出允许或拒绝决定,并记录总时耗(第 3 栏)。表 2 所列的是 5 次实验的平均值。

表 2 访问控制模块实现的性能测试

策略库大小(单位:个)	加载时间(单位:毫秒) 本文实现/SunXACML	500 次处理时间(单位:毫秒) 本文实现/SunXACML
50	1334.2/856.2	128/172
100	2018.6/1184.2	137.6/265.4
200	3215.4/1768.6	137/440.4
500	6184.4/3319	128.2/1165.8
1000	10368.8/5759	140.4/2653.2

两者的加载时间都随策略库增大而增加,但 SunXACML 的加载性能比本文实现好一倍,因为本文实现中采用索引机制,加载策略的同时需要分析策略的 Target 子元素,并建立反向索引。但 500 次决定请求的处理时间,本文实现优于 SunXACML,并且不随策略库增大而增加。因为索引机制使得适用策略匹配时间为一次散列时间,而单一策略评估时间是稳定的。SunXACML 没有使用这种机制,导致适用策略匹配速度慢,且随策略个数增加。由于策略加载只在系统启动时进行,因此这种优化提高了访问控制模块的整体性能。

结束语 本文的工作可以总结为以下 4 点:①基于 ABAC 模型和对 Web 服务资源特征的研究,提出了 Web 服务资源的两层访问控制框架 2L-ABAC;②定义了 2L-ABAC 的访问控制模型并给出了策略规则评估算法;③针对访问控制策略发布问题,提出基于授权断言(ws-xacml:XACMLAuthzAssertion)[16]扩展 WS-Policy 的方法,并根据各层特点提出了不同的发布方式;④基于国际来规范实现了 ABAC 模型,测试并分析了其性能。2L-ABAC 适应 Web 服务资源的特征,能够对来自其他安全域的用户进行访问控制。另外,它基于国际规范实现,所以具有通用性。下一步我们将研究用户隐私机制以及匿名授权机制,因为主体属性中可能包含用户的隐私信息。另一个研究方向是对访问控制策略决策点进行语义扩展,使其能根据已知实体属性进行语义推理,以获得缺失属性的值。

参 考 文 献

[1] Foster I, Czajkowski K, et al. Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF[J]. Proceedings of the IEEE, 2005, 93: 604-612

[2] Graham S, Karmarkar A, Mischkinsky J, et al. Web Services Resource 1.2. OASIS Standard[S]. April 2006

[3] Mewar V S, Aich S, Sural S. Access Control Model for Web Services with Attribute Disclosure Restriction[C]// Second International Conference on Availability, Reliability and Security (ARES'07). Washington D. C. USA: IEEE Computer Society, 2007: 524-531

[4] Shen Hai-bo, Hong Fan. An Attribute-based Access Control Model for Web Services[C]// Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06). Washington DC USA: IEEE Computer Society, 2006: 74-79

[5] 李晓峰,冯国登,陈朝武,等. 基于属性的访问控制模型[J]. 通信学报, 2008, 29(4): 90-99

[6] Yuan E, Tong Jin. Attributed-based Access Control (ABAC) for Web Services[C]// Proceedings of the IEEE International Conference on Web Services (ICWS'05). Washington DC USA: IEEE Computer Society, 2005: 561-569

(下转第 133 页)

不同的运用场景，其应用效果也有差异。RTBAC 模型以角色和任务作为访问控制基础，能较好地应用于下列领域：

1)远程协作办公平台。经济全球化的发展，企业日益分散的物理办公地点导致该方面的应用需求旺盛；

2)远程教育及远程医疗领域。该类系统要求很好的交互和协作性，从而决定系统对访问控制要求很高；

3)电子商务领域。虚拟企业、供应链管理等系统的发展提出了对系统访问控制方面更多的需求。

在分析和建立了 RTBAC 模型后，我们将其应用到了本实验室的学者网(scholat)项目的实践过程中，学者网面向科研工作者和科研团队，提供社交网络服务，管理个人和团队学术信息和资源，分享论文写作、投稿经验，对期刊、会议的评价等，可以帮助研究者发现科研热点或某一领域的研究群体，促进学术交流和创新。

如图 3 所示，学者网的系统架构遵循的是三层模型的体系结构。表示层负责提供用户界面，业务逻辑层负责实现业务逻辑，数据持久层负责业务逻辑层中所有数据的持久化存储。数据存储采用数据库表和 XML 文件两种存储方式。用户的个人信息、角色任务信息、任务权限信息等以数据库表的形式存储在数据库里。而角色的层次结构信息、工作流信息等就以 XML 文件的形式存储。数据持久层主要包含两个部分：1)DBAccess，对数据库表的访问与操作；2)XMLProcess，对以 XML 文件存储的数据进行读取、修改、查询等操作。业务逻辑层的基本模块包括 UserService，RoleService，TaskService，WorkflowService，主要是由 ManagerService 调用来完成业务逻辑的处理过程。

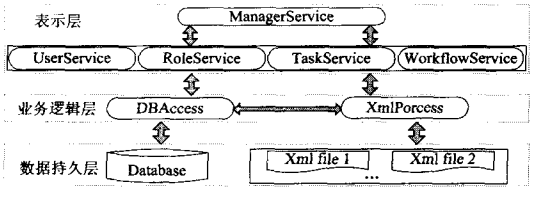

图 3　学者网系统架构设计图

结束语　本文根据 CSCW 系统特点分析了该类系统对访问控制的需求，并进一步针对该需求提出了基于角色和任务的 CSCW 系统访问控制模型(RTBAC 模型)，并将 RTBAC 模型实际用在了学者网(scholat)项目的开发过程中。RTBAC 模型基于角色对用户实现分组权限管理，有方便的授权/取消机制；角色间提供了偏序继承和角色指派关系定义，实现了职责分离原则及用户的转授权关系；通过基于任务的角色访问控制和对任务类型的划分，实现了对独立性任务和流程性任务采用不同的访问控制方法，满足“最小特权原则”及用户权限根据任务的执行过程实现动态分配。

参 考 文 献

[1] 汤庸，冀高峰，朱君. 协同软件技术及应用[M]. 北京：机械工业出版社，2007

[2] 龚能，李玉顺，史美林. 协作环境中的关键技术研究[J]. 计算机科学，2005，32(9)：230-232

[3] Tolone W，Ahn G-J，Pai Tanusree，et al. Access control in collaborative systems[J]. ACM Computing Surveys，2005，37(1)：29-41

[4] David F，Richard K. Role-based access controls [C]// Proceedings of the 15th NIST2NCSC National Computer Security Conference. Baltimore，MD，1992：555-560

[5] Oh S，Park S. Task-role-based access control model [J]. Information Systems，2003，28(6)：535-542

[6] Zhu H B，Zhou M C. Role-based collaboration and its kernel mechanisms[J]. IEEE Trans. on Systems，Man，and Cybernetic-Part C：Applications and Reviews，2006，36(4)：579-580

[7] 李成锴，詹永照，茅兵，等. 基于角色的 CSCW 系统访问控制模型[J]. 软件学报，2001，11(7)：931-934

[8] 邓集波，洪帆. 基于任务的访问控制模型[J]. 软件学报，2003，14(1)：76-80

[9] Sandhu R，Conyne E J，Lfeinstein H L，et al. Role based access control models[J]. IEEE Computer，1996，29(2)：38-43

(上接第 129 页)

[7] Jung I Y，Cho I S，Yeom H Y. A Stateful Web Service with Scalable Security on HVEM DataGrid[C]// Third IEEE International Conference on e-Science and Grid Computing. Washington DC USA：IEEE Computer Society. 2007：360-369

[8] Moses T，Inc E. eXtensible Access Control Markup Language (XACML) Version 2. 0. OASIS Standard[S]. Feb 2005

[9] Winsborough W H，Jacobs J. Automated Trust Negotiation Technology with Attribute-based Access Control[C]// Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03). 2003，2：60-62

[10] Bajaj S，et al. Web Services Policy Framework (WS-Policy) Version 1. 2[S]. March 2006

[11] Ballinger K，Bissett B，et al. Web Services Metadata Exchange (WS-MetadataExchange) Version 1. 1[S]. August 2006

[12] Li Ninghui，Mitchell J C. RT：A Role-based Trust-management Framework[C]// DARPA Information Survivability Conference and Exposition (DISCEX). Washington D. C，2003，1：201-212

[13] Winslett M，Yu Ting，Seamons K E，et al. Negotiating Trust on the Web[J]. IEEE Internet Computing，2002，6：30-37

[14] Coetzee M，Eloff J H P. Towards Web Service access control [J]. Computers & Security，2004，23：559-570

[15] Cantor S，Kemp J，Philpott R，et al. Assertions and Protocols for the OASIS Security Assertion Markup Language(SAML)[S]. OASIS Standard，March 2005

[16] Anderson A. Web Services Profile of XACML (WS-XACML) Version 1. 0. Working Draft 5[S]. October 2006

[17] Turkmen F，Crispo B. Performance Evaluation of XACML PDP Implementations[C]// Proceedings of the 2008 ACM Workshop on Secure WebSservices. New York，USA：ACM，2008：37-44

[18] http://sunxacml. sourceforge. net/

[19] Vedamuthu A S，Orchard D，et al. Web Services Policy 1. 5-Attachment[S]. W3C Recommendation 04，September 2007