第 36 卷 第 7 期 计 算 机 工 程 2010 年 4 月
Vol.36 No.7 Computer Engineering April 2010
·安全技术· 文章编号：1000—3428(2010)07—0132—04 文献标识码：A 中图分类号：TP393.08

面向分层式资源的基于属性的访问控制方法

陈 凯，郎 波
(北京航空航天大学软件开发环境国家重点实验室，北京 100191)

摘 要：针对 Internet 上经常使用的分层式资源管理模型，提出一种基于属性的访问控制模型 HR_ABAC，采用 XACML 国际标准作为该模型的策略描述语言，研究实现基于属性的访问控制决策机制。对该决策机制进行测试与分析，结果表明所实现的基于属性的访问控制方法具有有效性和实用性。

关键词：基于属性的访问控制；可扩展访问控制标记语言；策略决策点；策略管理点

Attribute Based Access Control Method Oriented to Hierarchical Resource

CHEN Kai, LANG Bo
(State Key Lab of Software Development Environment, Beihang University, Beijing 100191)

【Abstract】This paper presents an Attribute Based Access Control(ABAC) model HR_ABAC for hierarchical resource model which is commonly used in Internet. An access control decision mechanism is proposed and implemented for this model by using eXtensible Access Control Markup Language(XACML). It tests and analyzes this decision mechanism, and result indicates the mechanism is valid and practical.

【Key words】Attribute Based Access Control(ABAC); eXtensible Access Control Markup Language(XACML); policy decision point; policy administration point

1 概述

随着计算机应用技术的发展，访问控制技术逐渐成为了一种重要的计算机安全保护机制。近年来几种访问控制技术相继出现，其中包括自主访问控制[1](Discretionary Access Control, DAC)、强制访问控制[1](Mandatory Access Control, MAC)以及基于角色的访问控制(Role Based Access Control, RBAC)等基于身份的访问控制(Identify-based Access Control, IBAC)技术。在开放环境下，访问交互各方可能处于不同的安全域内，交互各方并不能够完全知道对方的信息，而传统的 IBAC 是基于主体和客体的唯一标识来实现的，对于这种开放的动态环境，IBAC 无法很好满足其安全需要。

随着分布式技术的发展，一种新的访问控制模型，即基于属性的访问控制[1](Attribute Based Access Control, ABAC)模型被提出。在 ABAC 中，访问决策基于请求者、资源等实体的属性[2]，不须像 IBAC 那样通过标识来实现，这样使得 ABAC 有充分的灵活性和可扩展性，能够适应开放、动态的环境。

最近在 ABAC 的研究和开发主要都是基于 X.509 属性证书的[2]。从 20 世纪 90 年代开始，公开密钥体系[2](Public-key Infrastructure, PKI)以及 X.509 这 2 种证书随着计算机网络的发展被广泛地用于信息认证[3]。1997 年属性证书(Attribute Certificates, AC)被加入到 X.509 标准中[4]，在 X.509 V4 版本中规定了 AC 的格式和内容。AC 将用户名与一个或多个属性进行绑定，这种数字证书包含所有者 ID、签发者 ID、签名算法、有效期等信息。国际上有一些使用属性证书进行属性认证的典型系统，它们已被用于一些网格系统中进行属性认证，但这些系统都存在灵活性不足及平台兼容性不强等问题。针对这些问题，需要一种新的 ABAC 模型来满足新的应用需要。

OASIS(结构化信息标准推进组织)于 2003 年 2 月发布了可扩展访问控制标记语言[5](eXtensible Access Control Markup Language, XACML)规范，该规范由 SUN, IBM 等公司开发。XACML 定义了访问请求和访问策略的语法，并描述了访问请求决策的基本流程。在 XACML 中访问请求描述的自然语义为“某种条件下，主体以某种方式访问资源”，其中，条件、主体、方式、资源是通过属性值来描述的。目前 Sun 等公司已经对 XACML 规范决策核心部分进行了实现，但这种实现只是一种演示 XACML 语言功能的演示性实现。

当前互联网上的资源非常丰富，这些资源经常按照分层资源模型进行组织，目前还缺乏一种针对分层资源的 ABAC 模型。本文的目标是要研究并提出一种 ABAC 策略模型 HR_ABAC(Attribute Based Access Control on Hierarchical Resource)，并实现一种 ABAC 决策机制对分层资源进行保护。

2 HR_ABAC模型的定义

ABAC 是以参与决策的相关实体的属性(而不仅仅是身份)为基础进行授权决策的一种访问控制机制。属性是指某实体(主体(Subject)、资源(Resource)、环境(Environment)和操作(Action))相关的一些特性。主体属性(Subject Attribute)包括主体的身份、角色、已验证的 PKI 证书等；资源属性(Resource Attribute)包括资源的身份、位置(URL)、大小、值等属性；环

基金项目：国家“863”计划基金资助项目“网格计算中基于属性的访问控制模型及实现方法研究”(2006AA01Z441)；国家“863”计划基金资助重点项目“可信的国家软件资源共享与协同生产环境之面向服务的软件生产线”(2007AA010301-02)

作者简介：陈 凯(1985－)，男，硕士研究生，主研方向：信息安全；郎 波，副教授

收稿日期：2009-10-20 **E-mail：**chenkai@nlsde.buaa.edu.cn

境属性(Environment Attribute)是与事务处理关联的属性，如时间、日期、系统状态、安全级别等；操作属性(Action Attribute)是与主体对资源所进行的操作有关的属性。利用主体、资源、环境和操作的属性来定义授权，既简化了管理，又增加了灵活性。

在对网络资源进行管理时，经常使用一种分层的资源组织方式。这种分层的资源组织方式使用资源类别对资源进行分类，首先将所有资源分为多种第 1 层类型，然后将每种第 1 层类型资源分为多种第 2 层类型，直到某层类型中的资源无法再进行分类。该资源结构能够使用一种树状结构进行表示。在这种树状结构中，叶节点代表的都是具体资源，非叶节点代表的是资源类型。资源树上每一个叶节点(具体资源)均有一个从根节点到该叶节点的节点链与其相关联。

在分层资源结构中，可使用与分层资源结构相对应的方法来管理访问控制策略。这种策略管理方法是将策略与资源相绑定，将策略定义到具体资源(叶节点)或资源种类(非叶节点)上，在请求要求对资源进行访问时，从请求资源到根节点间所包含的节点链上绑定的所有策略都将会作为判断请求能否通过的依据，同时针对每个资源都包含一个策略合并算法用来对这些策略的判断结果进行处理。网站资源层次结构如图 1 所示，根节点下有 movie/, picture/等第 1 层节点。movie/节点下有 PG-13/等第 2 层节点，PG-13/节点下包含 Lord Of War 等资源。一个访问请求(REQUEST)请求对“/movie/PG-13/Lord Of War”资源进行访问，该资源到根目录间可以使用一个节点集合{/, /movie/, /movie/PG-13/, /movie/PG-13/Lord Of War}来表示。在这个节点集合中 movie/和 PG-13/2 节点有绑定策略，这 2 条策略将用于请求判断。

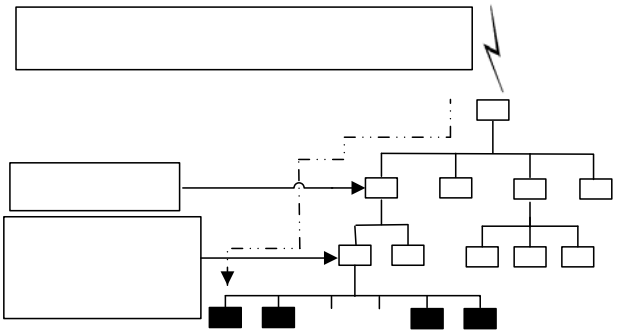

图 1　分层资源层次结构

下面给出分层资源 ABAC 模型的形式化定义：

(1)定义符号：使用 *S, R, E* 和 *A* 分别表示主体、资源、环境和操作。使用 $SA_k(1\leqslant k\leqslant K)$, $RA_m(1\leqslant m\leqslant M)$, $EA_n(1\leqslant n\leqslant N)$, $AA_o(1\leqslant o\leqslant O)$分别代表 *S, R, E* 和 *A* 的预定义属性。使用 *Perm* 来表示策略的判断结果集合，该集合中包括 2 个元素 Permit/Deny 表示通过/拒绝。使用[]符号表达某种元素的排列(元素可重复，有顺序)。使用 Policy 来表示所有策略的集合。使用四元组：*request*(*s, r, e, a*)表示一条访问请求。使用 *ATTR*(*s*), *ATTR*(*r*), *ATTR*(*e*)和 *ATTR*(*a*)分别表示主体、资源、环境和操作的属性赋值关系，分别为各自预定义属性的子集，其取值与具体的 *s, r, e, a* 有关，故有：

$ATTR(s)\subseteq SA_1\times SA_2\times\cdots\times SA_K$

$ATTR(r)\subseteq RA_1\times RA_2\times\cdots\times RA_K$

$ATTR(e)\subseteq EA_1\times EA_2\times\cdots\times EA_K$

$ATTR(a)\subseteq AA_1\times AA_2\times\cdots\times AA_K$

(2)函数定义：在 ABAC 框架下使用策略对请求进行判断，决策过程中所使用的变量是请求包含实体所具有的属性，判断结果类型为 *Perm*，可将策略判断函数定义如下：

$Policy(ATTR(s), ATTR(r), ATTR(e), ATTR(a))\rightarrow Perm$

由于 SA_k 等预定义属性对于一个确定的系统来说是固定的，因此其取值只与所处的 *request* (*s, r, e, a*)有关，一个 *Policy* 对于一个请求进行决策可以定义函数 *PolicyDecision* 如下：

$PolicyDecision(policy,s,r,e,a)\rightarrow Perm$ (1)

任何资源节点都存在唯一的合并算法，定义函数来表示从资源查找合并算法的过程：

$ResourceToCombining(Resource)\rightarrow CombiningAlgorithm$

任何资源节点都可能绑定一条策略，使用⊥表达某资源无绑定策略，判断结果为找到一条合适策略或无策略，故可定义函数来表示通过资源找策略：

$ResourceToPolicy(Resource)\rightarrow Policy\cup\perp$

在已经有策略判断结果的情况下通过合并算法(*CombiningAlgorithm*)对结果(*Perm* 类型)进行合并，返回 *Perm* 类型结果，定义如下函数：

$Combine([Perm],CombiningAlgorithm)\rightarrow Perm$ (2)

由于资源集合采用层次模型来表达，因此在资源集合 *R* 上存在一种偏序关系，使用≤符号来表达。使用根目录(/)表达最大的元素对于 *R* 上的任意元素 *r* 都有 $r\leqslant/$。使用 *parent*(*r*)来表达 *r* 资源的父节点规定 $r\leqslant parent(r)$，根目录无父节点。在 *R* 集合上规定区间$[q, p]$表示所有 $z\in R$，同时 $z\leqslant p$ 且 $q\leqslant z$ 所组成的集合。一个给定资源 *r*，使用 $Chain(r) = [r, /]$表示由 *r* 到根节点间所有资源组成的有序集合。故可用函数表达从某资源找其相关策略：

$PolicyOfResource(Resource)\rightarrow[Policy]$

对于该公式可用如下定义式：

$PolicyOfResource(r)\equiv\{ResourceToPolicy(o)|o\in Chain(r)\}$ (3)

(3)推演：对于一个请求元组(*s, r, e, a*)是否能够达到访问目的，可以使用如下函数：

$Access\_Allowed(s,r,e,a)\rightarrow Perm$

可用定义式：

$Access\_Allowed(s,r,e,a)\equiv Combine(\{PolicyDecision(PolicyOfResource(r),s,r,e,a)\}, ResourceToCombining(r))$

通过式(1)~式(3)可以推出：

$Access\_Allowed(s,r,e,a)= Combine(\{PolicyDecision(ResourceToPolicy(o),s,r,e,a)|o\in Chain(r)\},ResourceToCombining(r))$

Access_Allowed 是对访问请求进行判断的函数。在分层资源模型下主体请求对某资源进行访问，发出请求(*s, r, e, a*)，决策器收到请求后用式(3)中的 *Chain*(*r*)找到请求资源到根节点间的节点，再用式(3)中 *ResourceToPolicy* 函数找资源对应策略，然后用式(1)中 *PolicyDecision* 函数对得到的各个策略使用这个请求四元组进行决策得到一组策略判断结果，最后使用式(2)将 *r* 资源所对应的 *CombiningAlgorithm* 对这组 *Perm* 结果进行处理，得到最终的判断结果。

3　基于XACML的HR_ABAC决策机制

3.1　用XACML描述的HR_ABAC策略

XACML 是基于 XML 的策略描述语言，由于其策略决策是基于请求实体属性的，因此它能够被用来描述 ABAC 策略。XACML 规范中核心元素 PolicySet(策略集)内各元素的层次

关系如图 2 所示。该图包括规范中主要元素。

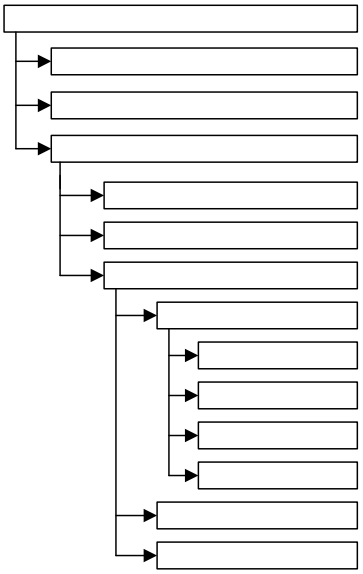

图 2　XACML 元素结构

PolicySet(策略集)为 XACML 策略语言结构最高层，在一个请求要求对某资源访问时，适合该资源的 PolicySet 将会对该请求进行判断，决定该请求是否能够访问资源。PolicySet 由 Target(目标)、CombiningAlgorithm(合并算法)以及若干个 Policy(策略)组成。Target 用来说明所属的 PolicySet 是否适合访问请求，Target 由 4 种元素组成，Target 使用这些元素分别对请求内容进行匹配，如果匹配成功则说明所属 PolicySet 能够适用该请求，Policy 和 Rule 也包含 Target 元素。CombiningAlgorithm 表示生成 PolicySet 决策结果时对 Policy 决策结果的合并方法。Policy 元素为 PolicySet 判断提供支持，Policy 由 Target, CombiningAlgorithm 和 Rule(规则)组成。Rule 元素为 Policy 判断提供支持，其由 Target, Condition(条件)和 Obligation(义务)组成。Condition 为 Rule 对请求判断过程中所包含判断逻辑的部分，其判断结果决定 Rule 的判断结果。Obligation 为 Rule 规定的访问资源之前必须完成的任务。

在 HR_ABAC 模型中的策略是与资源相关的，每一个资源种类或资源都可能绑定一条策略。从一个资源开始到根节点间的资源链绑定了多条策略，本文使用一定的合并算法(取并，取交，或首策略优先等)将这些策略组织起来形成了一个策略集。这个策略集就是与这个资源相关的策略集，用来保护这个资源。

3.2　XACML的实现机制

XACML 模型的框架如图 3 所示。XACML 分为 5 个部分：策略执行点(Policy Enforcement Point, PEP)，策略决策点(Policy Decision Point, PDP)，策略信息点(Policy Information Point, PIP)，策略管理点(Policy Admission Point, PAP)和上下文处理器(Context Handler)。PEP 负责与外部应用交互，将从外部应用获取的请求传递给 PDP，再从 PDP 获得返回结果。PDP 为判断请求是否能够访问的决策部分。PIP 为 PDP 提供决策过程中需要的属性信息。PAP 为 PDP 提供用来决策的策略。上下文处理器负责前 4 个部分间消息传递。因为 XACML 模型框架具有开放性、灵活性和跨平台性，而且其规范只规定各模块间的消息传递适合开放系统，又因为 XACML 策略语言是基于请求者等实体的属性进行决策的，所以 XACML 是适合用来描述 HR_ABAC 策略的工具。

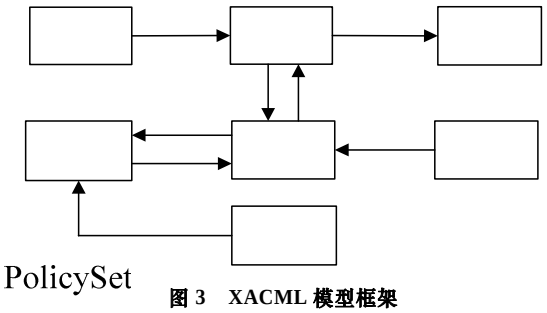

图 3　XACML 模型框架

3.3　HR_ABAC决策机制

对 ABAC 实现模型进行研究并结合 XACML 框架结构进行分析，提出一种 HB_ABAC 决策机制模块结构，如图 4 所示，该结构与 XACML 框架结构相似，主要包括 PEP, PDP, PAP, PIP 这 4 个部分。

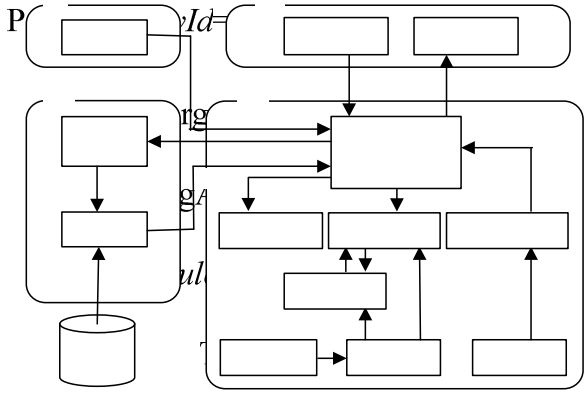

图 4　HR_ABAC 决策机制模块

PEP 模块在决策机制中是与外界交互的模块，从外部应用中获取访问请求，转发至 PDP，通过 PDP 获得最终决策结果。PIP 为 PDP 在决策过程中提供所需属性，PIP 定义获取属性的接口，不同应用有不同的实现方法，例如使用属性证书 CA。PDP 模块负责对请求进行决策判断，它获取从 PEP 处发送的请求，通过从 PAP 中获取合适策略集，PDP 将此策略集拆分为单个子策略，然后基于对子策略进行目标匹配得到子策略结果，最后对这些结果进行合并得到最终结果，在此过程中用到了计算函数和合并算法等 XACML 规范中的元素，PDP 支持了这些元素，PDP 对应 ABAC 形式化模型中 *PolicyDecision* 函数，具有核心判断功能。PAP 模块负责为 PDP 提供合适的策略。它首先从 PDP 中获取了请求，从资源库中查找请求资源对应的资源链，然后在策略库中查找该资源有关策略集返回给 PDP，PAP 对应 ABAC 形式化模型中 *PolicyOfResource* 函数。下面介绍 PDP 和 PAP 算法流程。

3.4　策略决策算法

策略决策算法是 PDP 模块的核心算法，其功能是对请求进行判断，伪代码如下所示：

```
Decision(request)
begin
    //使用 PAP 根据 request 获取适合的策略集
    policyset = PAP.find_PolicySet(request)
    combinealg =policyset.CombineAlg
    //从策略集中获取所有子策略并判断结果
    policy[] = policyset.getChild()
    for i = 1 to policy.length do
        decision[i] = policy[i].evaluate(request, PIP);
```

```
    //使用合并算法合并子策略结果得到最终结果
    result = combinealg.combine(decision)
    return result
end
```

算法首先用PAP查找所有与请求有关的策略组成的策略集，然后获取该策略集对应的策略合并算法，再针对每一条策略对请求进行判断得到每一条策略的决策结果，最后使用策略合并算法对这些结果进行合并返回判断结果。

3.5 策略查找算法

策略查找算法是PAP模块的核心算法，其功能是根据请求获取对应的策略集，伪代码如下所示：

```
find_PolicySet(request)
begin
    resource = request.resource
    combinealg = CombineAlg.find(resource)
    //获取从资源节点到根节点查找有关策略
    do
        p = findPolicy(resource)
        if (p != NULL) do
            policy[i++] = p
        resource = resource.findParent()
    while resource != root //将策略与合并算法合并成为策略集
    policyset = makePolicySet(policy, combinealg)
    return policyset
end
```

算法首先获取请求中的资源及其对应合并算法，然后在层次结构的资源树中从资源节点开始查找父节点，直到根节点，并将过程所有资源对应的策略保存，最后将策略与合并算法合并成为策略集并返回。

4 访问控制决策机制的测试

为了验证本文所实现的访问控制决策工具的正确性和效率，将决策工具部署到网络服务(Web Service)环境下进行验证和测试。

本文使用XACML标准[5]提供的测试用例套件进行功能性测试，测试用例套件包括374个策略请求对，测试套件不但包括结果为通过和不通过的用例，还包括XACML标准中规定的各种出错情况的用例。根据对决策工具执行结果与标准所给出的正确结果进行比较得到，该XACML决策工具能够得到正确的结果，从而证明了决策工具功能的正确性。

为了对决策工具的效率进行测试，本文以不同规模的策略为标准对访问控制决策工具进行测试。因为策略的复杂程度是与规则的数量成正比，所以本文使用决策器对不同规模的策略进行决策来分析效率。在CPU为Intel Core 2 E7300 2.66 GHz，内存为2 GB，操作系统为Windows XP SP2环境下对决策工具进行测试，测试结果如表1所示(规则条数为待测试的策略中规则的数量，运行时间为测试过程中运行时间，理论时间为如果运行时间与规则条数呈线性关系则理论上决策工具应该运行的时间)。

表1 决策工具性能测试

规则条数	运行时间/ms	理论时间/ms
5	0	-
10	0	-
20	1	1
50	3	2.5
100	6	5
500	22	25
1 000	47	50
2 000	95	100

本文将规则规模分别设定在从5~2 000的8种情况下进行测试，在规则量小于100的情况下，使用的判断时间小于30 ms，比较接近，尤其在规则条数小于10时运行时间是相同的，故将规则条数小于10时所花时间作为基准值，将其他各项时间减去这个基准时间得到运行时间。由表中数字可以看出，运行时间与规则条数近似呈正比关系，同时列出正比关系时所应该得到的理论时间。通过运行时间与理论时间的比较可以看出，判断规则所使用时间与规则条数是正比关系。通常一条策略中的规则数不会太大，判断时间不会影响系统性能，所以这种模型具有一定的实用性。

5 结束语

本文在层次化资源模型上提出了一种ABAC的形式化模型HR_ABAC，使用XACML作为HR_ABAC模型的实现框架，实现了一种ABAC访问控制决策机制。本文实现的访问控制决策机制能够应用到面向服务的构架等开放式环境中，并根据相关实体属性进行决策为层次型资源提供保护。

参考文献

[1] Yuan E, Tong Jing. Attribute Based Access Control(ABAC) for Web Services[C]//Proc. of the IEEE International Conference on Web Services. Piscataway, USA: IEEE Computer Society, 2005: 561-569.

[2] Perlman R. An Overview of PKI Trust Models[J]. IEEE Network, 1999, 13(6): 38-43.

[3] Lang Bo, Foster I, Siebenlist F, et al. Attribute Based Access Control for Grid Computing[EB/OL]. (2006-04-25). ftp://info.mcs.anl.gov/pub/tech_reports/reports/P1367.pdf.

[4] Park J S, Sandhu R. Smart Certificates: Extending X.509 for Secure Attribute Service on the Web[C]//Proc. of National Information Systems Security Conference. Arlington, USA: [s. n.], 1999: 340-346.

[5] OASIS. eXtensible Access Control Markup Language(XACML) Version 2.0[EB/OL]. (2005-02-01). docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

编辑 任吉慧

（上接第131页）

参考文献

[1] 夏征难. 信息战理论研究综述[J]. 中国军事科学, 2003, 16(3): 151-156.

[2] IWSS16系统[EB/OL]. (2008-10-30). http://www.nsfocus.net/.

[3] 西点军校. 信息保障作战实验室攻防训练研究平台[EB/OL]. (2008-11-20). http://www.defence.org.cn/Article-13-40179.html.

[4] 上海交通大学. 信息安全工程实践综合实验平台与集成[EB/OL]. (2008-10-20). http://863prj.sjtu.edu.cn.

[5] 中国科学院. 网络安全防护若干关键技术与防范实验平台[EB/OL]. (2005-03-16). http://www.kepu.net.cn/gb/innovative_project/strategic/intro/200503160011.htm.

[6] 张 筱, 林孝康. 一种基于OPNET的网络半实物仿真模型[J]. 微计算机信息, 2007, 23(1): 256-294.

[7] 赵志超. 网络攻击及效果评估技术研究[D]. 长沙: 国防科技大学电子科学与工程学院, 2002.

编辑 索书志