第 34 卷 第 8 期 计 算 机 工 程 2008 年 4 月
Vol.34 No.8 Computer Engineering April 2008
·博士论文· 文章编号：1000—3428(2008)08—0019—03 文献标识码：A 中图分类号：TP311

基于 XACML 的访问控制与 RBAC 限制

努尔买买提·黑力力[1,2]，罗振兴[1]，林作铨[1]
(1. 北京大学信息科学系，北京 100871；2. 新疆大学数学与系统科学学院，乌鲁木齐 830046)

摘 要： 限制可以视为是基于角色的访问控制(RBAC)的主要动机。该文分析基于 XML 的访问控制规范语言(XACML)的 RBAC 框架并指出了该框架的缺点，通过提出的角色激活机构对该框架进行扩充，使得 XACML 支持 RBAC 模型中的职责分离和基数限制等限制。
关键词： Web 服务；XML 的访问控制规范语言；基于角色的访问控制

XACML-based Access Control and RBAC Constraints

Nuermaimaiti·Heilili[1,2], LUO Zhen-xing[1], LIN Zuo-quan[1]
(1. Department of Information Science, Peking University, Beijing 100871;
2. College of Mathematics and System Sciences, Xinjiang University, Urumqi 830046)

【Abstract】 Constraints are considered to be the principal motivation for Role-Based Access Control(RBAC). This paper analyzes XML based access control language XACML and points out some shortcomings of the XACML profile for RBAC. It provides role enablement authority to extend this profile, in this way, several kinds of constraints of RBAC such as separation of duty constraints and cardinality constraints can be enforced and implemented using XACML.
【Key words】 Web services; eXtensible Access Control Markup Language(XACML); Role-Based Access Control(RBAC)

1 概述

基于角色的访问控制(RBAC)[1-2]是一种可扩展的访问控制模型，具有通用和灵活的特性。可扩展访问控制标记语言(eXtensible Access Control Markup Language, XACML)定义了一种通用的用于保护资源的策略描述语言。OASIS定义了一个XACML的RBAC框架[3]，规范了使用XACML来表示NIST RBAC标准模型[4]中的核心(core)模型和层次(hierarchical)模型。XACML的RBAC框架中的角色是主体(subject)的一个属性，而不是与主体对等的实体，表达RBAC的一些基本要素，如用户角色指派(激活)、职责分离和基数限制等，不易于在实际的访问控制系统中实施。

本文对 XACML 的 RBAC 框架进行了分析，指出该规范存在的一些缺陷：不能完整地支持 RBAC 限制，也没有给出使用 RBAC 策略的数据流程。角色限制是 RBAC 模型中提出的一个重要概念，是整个模型中的一系列约束条件，用于实施利益冲突策略，防止组织内的用户超出自己的权限范围。一些典型的限制包括职责分离和角色基数限制。对 XACML 的 RBAC 规范进行了扩充，使用 XACML 编写 RBAC 策略，使得 RBAC 模型中的职责分离和基数限制在 XACML 中得以实现，保持了 XACML 本身的特性，即 PEP/PDP 模式。由于 XACML 是无状态的，因此引入了用户角色激活状态的概念来解决动态职责分离、动态角色基数限制等问题。可以为用户指派权限提供比较复杂的限制条件以及对这些复杂的限制进行评估，使得 XACML 能进一步支持 RBAC 中的各个要素。

2 XACML 的 RBAC 框架的扩充

2.1 基于 XACML 的 RBAC 策略的表达

XACML的RBAC框架[3]规范了如何使用XACML来表示RBAC NIST标准模型中的核心模型和层次模型。角色指派和激活不是由XACML PDP完成，而是由一个单独的实体来完成，该实体称为角色激活机构(role enablement authority)。XACML的RBAC框架没有对角色激活机构进行规范，只是给出了几种可能的实现方式。角色激活机构可以通过Role Assignment <Policy>或<PolicySet>来判断用户(主体)是否指派了某个角色属性。

OASIS 提出的 XACML 的 RBAC 框架存在着一些缺陷，不能完整地支持 RBAC 限制，也没有给出使用 RBAC 策略的数据流程，而角色限制是 RBAC 模型中非常重要的一方面，是 RBAC 模型的主要动机之一。角色激活机构在 XACML 体系中缺乏具体的定位，

本文对 OASIS 的 XACML 的 RBAC 框架进行扩充，以便支持 RBAC 的静态和动态职责分离以及基数限制，如图 1 所示。管理员通过 PAP 事先定义好 Role Assignment <PolicySet>，管理员对角色指派关系的管理需要经过角色管理 PDP 授权，确保用户角色指派操作满足角色的限制条件(如静态职责分离、静态基数限制)。当用户建立会话并激活角色时，PEP 将向上下文处理器发送角色激活请求。上下文处理再向 PDP 发送 XACML 格式的角色激活请求，其中，主体是用户；资源则是角色；操作是激活角色。该过程把角色看成一个特殊的资源。

基金项目： 国家自然科学基金资助项目(60373002, 60496322)；国家“973”计划基金资助项目(2004CB318000)
作者简介： 努尔买买提·黑力力(1976－)，男，博士研究生，主研方向：网络信息安全；罗振兴，博士研究生；林作铨，教授、博士生导师
收稿日期： 2007-04-30 **E-mail：** nur@is.pku.edu.cn

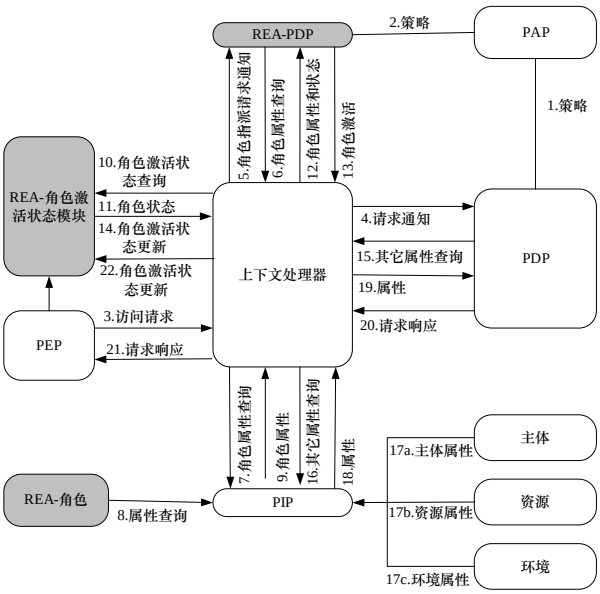

图 1　XACML 和角色激活机构之间的交互流程

2.2　静态职责分离

静态职责分离是职责分离的最简单的一种形式，如果定义 2 个角色为静态的角色互斥，任何一个用户都不能同时被指派给这 2 个角色，可以看成一种管理策略，对管理员的用户角色指派操作执行限制。XACML的RBAC框架中没有实现它。本文借助于XACML3.0[5]安全策略管理规范，建立静态互斥角色的管理策略来实现静态职责分离。例如，假设　2 个角色employee和contractor是静态互斥的角色，通过以下策略可以描述这种情况(简化的XACML策略):

<Policy PolicyId="Static Separation of Duty Policy"
PolicyCombiningAlgId="deny-overrides">
<Target>…Anne(DelegatedSubject)…employee(DelegatedResource)
…assign(DelegatedAction)
…Bob(Delegate…subject-id)…Administrator(Delegate…role)
</Target>
<Condition>…contractor(is-not-in…getAssignedRoleSet) …
</Condition>…</Policy>

管理员通过编写 Role Assignment<PolicySet>定义用户角色指派时，不能把互斥的角色指派给同一个用户，指派给用户的角色总数不该超过总数限制。

该管理策略描述管理员Bob在Anne没有被指派角色contractor的情况下可以创建给用户Anne指派角色employee的策略。自定义函数getAssignedRoleSet返回用户Anne已经被指派的角色集合。如果这个集合包含角色contractor，则管理员角色的指派请求被拒绝(指派请求可以参考XACML3.0[5]中的管理请求)。

2.3　动态职责分离

动态职责分离是在会话过程中对角色的激活进行限制，如果定义 2 个角色为动态的角色互斥，那么给一个用户可以指派这 2 个角色，但是在任何一个会话中都不能同时激活它们。假设在 Role Assignment<PolicySet>中定义用户 Anne 被指派了角色 employee 和 contractor。当用户在请求激活角色 contractor 的时候，上下文处理器将向角色激活状态模块查询用户 Anne 是否已经激活角色 employee。如果是，则不能再激活角色 contractor；否则允许激活角色 contractor。上下文处理器将生成义务把该角色指派信息 enable(Anne, contractor)添加到角色激活状态模块中。该动态职责分离策略可以参考 Separation of Duty<PolicySet>。

2.4　角色基数限制

角色基数限制用于指定一个角色可被同时指派或激活的数目，有静态和动态之分。静态的角色基数限制也可以看作是一种管理策略。下面的规范表示，角色 manager 的静态角色基数限制为 2。自定义函数 getAssignedUserCounter，参数为角色，返回被指派该角色的用户数目。

<Policy PolicyId="Static Role Cardinality Policy"
PolicyCombiningAlgId="deny-overrides">
<Target>…Anne(DelegatedSubject)…employee(DelegatedResource)
…assign(DelegatedAction)
…Bob(Delegate…subject-id)…Administrator(Delegate…role) …
</Target>
<Condition>…getAssignedUserCounter(manager) …integer-less-than-or-equal…2…</Condition>…</Policy>

当用户成功激活一个角色后，上下文处理器生成义务将该信息记录在角色激活状态模块中。自定义计数函数 getEnabledUserCounter，参数为角色，返回已经激活该角色的用户数目。该函数需要在角色激活状态中进行查询。下面的策略表示如果当前激活角色 manager 的用户数目大于 2，用户激活角色 manager 的请求将被拒绝。

<Policy PolicyId="Dynamic Role Cardinality Policy"Policy CombiningAlgId="deny-overrides">
<Rule RuleId="manager role cardinality"Effect="Permit">
<Target>…<Subject><AnySubject/></Subject>
<Resource>…manager…</Resource>
<Action>…enable…</Action>…</Target>
<Condition>…getEnabledUserCounter(manager)…integer-less-than-or-equal…2…</Condition>…</Rule>…</Policy>

3　实现机制

根据上述的讨论，对 SUN 的 XACML 进行了扩展，实现了一个支持 RBAC 的基本要素的原型系统，特别是限制，主要分为 5 个模块：登录模块，策略执行模块，上下文处理模块，用户状态记录模块，策略决策模块，如图 2 所示。在这个系统框架中，把管理请求和管理策略看成普通的 XACML 请求和普通的策略，本文主要讨论 RBAC 动态的限制(动态职责分离、动态基数限制)。

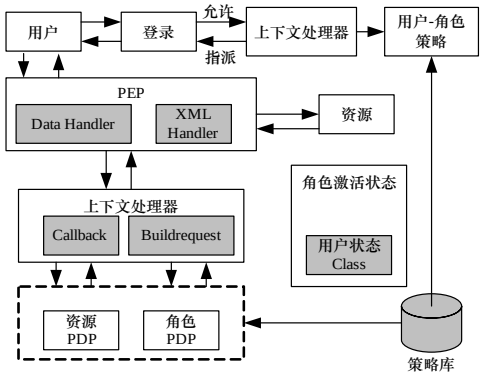

图 2　体系结构

登录模块是基于线程类的，用来对首次访问的用户进行身份认证。认证方式可以是用户名加口令或证书，这部分内容采用了笔者以前的部分工作[6]。不同的是它通过对策略解析实现用户的角色分配。用户一旦登录成功，将会得到相应的角色，可以选择适当角色来访问，并且用户以后的所有访问交由策略执行模块处理。

策略执行模块也是基于线程类的，主要包括数据处理和XML 的解析，实现请求的解析和要访问数据的读写工作。DataHandler 是基于线程和 Selector(NIO)类的，使用 Buffer 对象来处理要读写的数据。XML Handler 采用 Java 关于 XML 解析的 API，能够得到被解析的 XML 文档的 DOM 实例，从而将用户的信息提取出来，更易于程序调用用户的信息。

上下文处理模块连接着策略执行模块，用户状态记录模块和策略决策模块，是一个交通枢纽，包含 Callback 和 Buildrequest 功能模块。Callback 用来处理线程间的通信，进行信息查询，并将用户的必要信息写入状态记录模块。Buildrequest 将组合用户的相关请求信息重新生成标准的基于 XML 的 XACML 的请求，如生成多个主体的请求，由 Callback 发送给策略决策模块来决策。

用户状态记录模块是基于线程独立运行的模块，只要有用户会话它就会作相应的更新。它通过一个 HASHMAP 实例来对用户的状态进行管理，将一个用户视为一个类的对象实例作为 HASHMAP 的值，用户 ID 作为 KEY，这样易于访问用户的状态信息。当前类对象包含用户的 ID，当前激活的角色及角色个数等信息。

策略决策模块使用事件驱动的模式基于线程处理用户的请求，逻辑上分别处理不带角色的用户和带角色的用户，使用 SUN 实现的核心 PDP 实现本文的决策。相关的类主要是策略查找、资源查找、匹配算法等。

4 结束语

RBAC 模型是在大规模的 Web 应用中减少安全管理的复杂性和费用的一种策略，XACML 是基于 XML 的访问控制和授权标准。把 RBAC 模型嵌入到 XACML 有很重要的研究意义，它们之间的协调使它们进一步为 Web 服务的安全性提供灵活的和有效的访问控制。

本文对基于 XACML 的 RBAC 框架进行了扩充，使得 RBAC 模型中的动态职责分离和动态基数限制在 XACML 中得到实现。借助于 XACML3.0 安全策略管理规范，解决了使用 XACML 来表示 RBAC 中的静态职责分离和静态基数限制问题。下一步的工作是把 RBAC 管理模型结合到 XACML 中，使用 RBAC 管理模型来管理 XACML 策略本身以及 XACML 中的 RBAC 策略。

参考文献

[1] Ferraiolo D, Kuhn R. Role-based Access Controls[C]//Proceedings of the 15th NIST-NCSC National Computer Security Conference. Baltimore, USA: [s. n.], 1992.

[2] Sandhu R S, Coynek E J, Feinsteink H L, et al. Role-based Access Control Models[J]. IEEE Computer, 1996, 29(2): 38-47.

[3] OASIS. Core and Hierarchical Role Based Access Control(RBAC) Profile of XACML Version 2.0[EB/OL]. (2005-10-30). http://docs.oasis-open.org/xacml/2.0

[4] Ferraiolo D F, Sandhu R, Gavrila S, et al. Proposed NIST Standard for Role-based Access Control[J]. ACM Transactions on Information and System Security, 2001, 4(3): 224-274.

[5] OASIS. XACML v3.0 Administration Policy Version 1.0[EB/OL]. (2006-05-30). http://www.oasis-open.org/committees/tc_home. php?wg_ abbrev= xacml.

[6] Luo Zenxing, Nuermaimaiti Heilili, XU Dawei, et al. Web Application Security Gateway with Java Non-blocking IO[C]//Proc. of the 6th Workshop on Next Generation Information Technologies and Systems. [S. 1.]: Springer-Verlag, 2006.

(上接第 11 页)

加梯度最小，说明算法 c 对密度密度分布不敏感。这是因为算法 c 是动态分配空间网格。而算法 a 和算法 b 在同样条件下，随着 k 值的增大，算法 b 的优点逐渐减弱，而缺点逐渐显现出来。这是因为即使开始已经规划好网格，做好了预处理，但是却无法确定被测点一定处于网格中心的位置，需要搜索 8 邻域网格内的点，使搜索区域增大，花费了大量时间。

从以上对测量点分布密度不同的实验结果可以得出，只要不出现 m 值过小和 k 值过大的状况，算法 c 对测量点的密度分布是最不敏感的算法。

在表 1 和表 2 中，当 k 值增大到一定数量时，算法 b 和算法 c 的优越性逐渐减弱，甚至是不如算法 a 的计算效率高，特别是算法 b 最为明显。这是因为随着 k 值的增加，即使 m 值很大，但是花在每个点的 k 邻近计算时间也在增加。因此，当 k 值过大时，算法 a 效果最好。

5 结束语

本文提出的动态网格算法避免了包容盒法在划分空间网格时，由于网格内点数的不确定性所带来的缺陷。而本文则是根据点的密度，随意扩大或缩小该网格，从而可以快速求得 k 邻近点。从以上两组实验结果的分析来看，当测量点数、m 以及 k 值都较小时，直接使用算法 a；当测量点数逐渐增大，而 m 和 k 值都很小时，算法 b 比较合适；当测量点数、m 以及 k 值都较大时，算法 c 最合适；当 m 很小，k 值过大时，算法 a 最为有效。在反求工程领域里，通常获取的测量点从几万到几十万，数据量相当庞大，而且通常不会只求一个点的 k 邻近(即 m 通常不会很小)，一般都在几十到几百个点以上，另外 k 值一般不会太大。因此，算法 c 即本文的算法实用性更强。

参考文献

[1] 熊邦书, 何明一, 余华璟. 三维散乱数据的 k 个最近邻域快速搜索算法[J]. 计算机辅助设计与图形学学报. 2004, 16(7): 909-912.

[2] Goodsell G. On Finding p-th Nearest Neighbors of Scattered Points in Two Dimensions for Small p[J]. Computer Aided Geometric Design, 2000, 17(4): 387-392.

[3] Dickerson M T, Drysdale R L S, Sack J R. Simple Algorithms for Enumerating Interpoint Distances and Finding k Nearest Neighbors[J]. International Journal of Computational Geometry and Applications, 1992, 2(3): 221-239.

[4] Piegl L A, Tiller W. Algorithm for Finding All k Nearest Neighbors[J]. Computer Aided Design, 2002, 34(2): 167-172.

[5] Nandy S C, Das S, Goswami P P. An Efficient k Nearest Neighbors Searching Algorithm for a Query Line[J]. Theoretical Computer Science, 2003, 29(1): 273-288.

[6] 周儒荣, 张丽艳, 苏　旭, 等. 海量散乱点的曲面重建算法研究[J]. 软件学报, 2001, 12 (2): 249-255.