网格环境下基于上下文和角色的访问控制*)

张建风[1] 徐艳丽[1] 王汝传[1,2] 王海艳[1]

(南京邮电大学计算机学院 南京 210003)[1] (南京大学计算机软件新技术国家重点实验室 南京 210093)[2]

摘 要 基于角色的访问控制RBAC是当前比较流行的访问控制模型,但和其它的传统访问控制模型一样采用的是静态授权,没有考虑所处的上下文环境,在应用于以动态性为显著特征的网格计算环境,必然导致一定的缺陷。在基于角色访问控制RBAC模型的基础上进行了扩展,加入了用户活动角色、系统活动权限、用户活动权限、主体上下文、客体上下文的概念,并通过状态矩阵实现了基于上下文的访问控制,可以解决网格环境中上下文敏感的访问控制。

关键词 RBAC,用户活动角色,系统活动权限,用户活动权限,主体上下文,客体上下文

Context and Role Based Access Control in Grid

ZHANG Jian-feng[1] XU Yan-li[1] WANG Ru-chuan[1,2] WANG Hai-yan[1]

(College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)[1]

(State Key Laboratory for Novel Software Technology Nanjing University, Nanjing 210093, China)[2]

Abstract Role based access control is the most popular access control module at present, and can reduces workload and complexity of authorization administration. But it uses static authorization like other traditional access control method, and doesn't take the contexts into account. When applied to the computer grid which is characterized by dynamic contexts it must have some limitation. Extended role based access control module by adding user active roles, system active privileges, user active privilege, user contexts and operation contexts, and realizes context based access control by using state matrix, it can be applied to the dynamic access control in grid.

Keywords RBAC, USER_ACT_ROLES, SYS_ACT_PRMS, USER_ACT_PRMS, User_Contexts, Operation_Contexts

1 引言

网格计算就是整合不同组织、机构的计算机资源,形成一台巨大的虚拟计算机,完成超大规模的计算,以实现资源的充分利用和信息整合。网格要达到资源共享的目的,必须解决资源的访问控制问题。访问控制的核心是授权策略,即用于确定一个主体是否能对客体拥有访问能力的一套规则。传统的访问控制模型有自主访问控制(DAC)模型和强制访问控制(MAC)模型,在上世纪90年代提出了基于角色的访问控制(Role-Based Access Control, RBAC)模型,1996年Sandhu等提出了RBAC96[1]模型,系统全面地描述了RBAC多层次、多方面的意义,从而使RBAC得到了广泛的认可。RBAC模型在用户和权限之间引入角色,实现了用户与权限的逻辑分离,给管理员提供了一种从组织角度进行安全建模的有效途径,大大减少了授权管理的工作量和复杂性,迅速得到了广泛的应用。由于RBAC模型还是从系统的角度(控制环境是静态的)出发保护资源,不能随着执行环境的改变而动态授权。在RBAC模型基础上提出的基于任务的访问控制TBAC[2]模型和基于任务和角色的双重Web访问控制T-RBAC[3]模型,能随着执行任务所处的上下文环境动态授权,但它们是从执行任务的角度建立动态授权模型。文献[4]提出的网格环境下的一种动态跨域的访问控制策略,是根据用户的行为改变它的声誉,当到达极限值,撤销其角色来实现动态授权,这些都没有从执行环境的角度建立动态授权模型,对于在网格环境下很普遍的下面几种情况都不能适应:

(1)只允许用户在某个时间段对某网格资源进行访问,如晚上7:00到早上7:00;

(2)只允许用户在资源负载不重时对资源进行访问,如CPU负载<80%;

(3)只允许用户访问特定的文件,如文件大小<3M等。

文献[5]提出的带时间特性的角色访问控制,根据约束的时间特性(激活时间范围、激活时间长度和时间范围内激活时间长度)动态调整用户的激活角色,只可以解决第一种情况。针对网格环境下访问控制的特点,本文提出了用户活动角色、系统活动权限、用户活动权限、主体上下文、客体上下文的概念,实现了网格环境下上下文敏感的访问控制机制,给出了基于上下文和角色(Context and Role based Access Control, CRBAC)的访问控制模型。本文第2节介绍了网格环境下扩展RBAC模型的定义及实现;第3节讨论了该模型的可行性;最后总结本模型的特点和下一步的工作。

*)本课题得到国家自然科学基金(60573141和60773041),江苏省高技术研究计划(BG2006001),国家高科技863项目(2006AA01Z201、2006AA01Z219、2006AA01Z439、2007AA01Z404、2007AA01Z478),2006江苏省软件专项,南京市高科技项目(2007软资106,2007软资127),现代通信国家重点实验室基金(9140C1105040805),江苏省计算机信息处理技术重点实验室基金(kjs06006)资助和江苏省高校自然科学研究计划(07KJB520083)资助。**张建风** 硕士研究生,主要研究方向为计算机软件、计算机网络、移动代理和P2P技术等;**徐艳丽** 硕士研究生,主要研究方向为网格安全;**王汝传** 教授,博士生导师,主要研究方向是计算机软件、计算机网络、信息安全、移动代理和虚拟现实技术等;**王海艳** 副教授,博士生,主要研究方向为计算机软件、计算机网络、信息安全、移动代理等。

2 网格环境下基于上下文和角色的访问控制

2.1 标准 RBAC 扩展的定义

NIST 的 RBAC 模型按功能分成了 4 个组件，分别是核心 RBAC、层次 RBAC、静态职责分离 SSD 和动态职责分离 DSD。我们的定义是基于标准核心 RBAC，采用了核心 RBAC 的部分定义。在这里，我们将上下文分为主体上下文（和用户主体环境相关）和客体上下文（和资源、服务的状态相关），将用户角色中处于激活状态的角色集合称为用户活动角色，用户活动角色是用户角色的子集，随着用户主体上下文的改变而改变；同样，将用户权限（从用户活动角色继承来的以及直接分配的权限）中处于激活状态的权限称为用户活动权限，用户活动权限随着主体上下文和客体上下文的改变而改变。定义如下：

USERS，ROLES，OPS 及 OBS 分别表示用户、角色、操作及对象，SYS_ACT_PRMS 表示系统活动权限，USER_ACT_ROLES 表示用户活动角色，USER_ACT_PRMS 表示用户活动权限，$CONTEXTS_{主}$ 表示主体上下文特征信息（如链接的链路状态），$CONTEXTS_{客}$ 表示客体上下文特征信息（如 CPU 负载<80%、只允许在晚上 7:00 至早上 7:00 访问资源）。

$UA \subseteq USERS \times ROLES$ 一个多对多的用户和角色的分配映射关系

$(r:ROLES) \rightarrow 2^{users}$ 将角色 r 映射到一个用户集合，即 $U(r:ROLES)=\{u \in USERS \mid (u,r) \in UA\}$

$PRMS=2^{OPS \times OBS}$ 权限集合

$PA \subseteq ROLES \times PRMS$ 一个多对多的角色和权限的分配映射关系

$(r:ROLES) \rightarrow 2^{PRMS}$ 将角色 r 映射到一个权限集，即 $P(r:ROLES)=\{p \in PRMS \mid (r,p) \in PA\}$

$RC \subseteq ROLES \times CONTEXTS_{主}$ 一个多对多的角色和主体上下文的对应关系

$PC \subseteq PRMS \times CONTEXTS_{客}$ 一个多对多的权限和客体上下文的对应关系

$USER\_ACT\_ROLES(u:USERS, c_{主}:CONTEXTS_{主})=\{r \in ROLES \mid (u,r) \in UA \wedge (r,c_{主}) \in RC\}$

$SYS\_ACT\_PRMS(c_{客}:CONTEXTS_{客})=\{p \in PRMS \mid (p,c_{客}) \in PC\}$

$USER\_ACT\_PRMS(u:USERS, c_{主}:CONTEXTS_{主}, c_{客}:CONTEXTS_{客})$
$=\{p \in PRMS \mid r \in USER\_ACT\_ROLES(u:USERS, c_{主}:CONTEXTS_{主}) \wedge (r,p) \in PA \wedge p \in SYS\_ACT\_PRMS(c_{客}:CONTEXTS_{客})\}$

下面是扩展定义的说明：

主体上下文（$Context_{主}$）表示和主体环境相关的一些动态信息集合，如用户地点、用户时间、链路状态、本地资源状态等，主体上下文决定角色的即时有效性。

客体上下文（$Context_{客}$）表示和资源、服务的时间特性、统计等相关的一些动态信息，客体上下文决定权限的即时有效性。

系统活动权限集（System Active Permission）是一个动态权限集，表示系统在当前的客体上下文下处于激活状态的权限集。

用户活动角色集（User Active Perms）是一个动态角色集，表示特定的用户在该用户当前的主体上下文下处于激活状态的角色集。

用户活动权限集（User Active Perms）是一个动态权限集，表示特定的用户在当前的主、客体上下文下有效的权限集。

上下文敏感的或者依赖上下文，是指用户角色及权限要动态地根据上下文信息进行调整，而不是过去的静态授权关系。动态调整的方法依赖具体的实现，可以采取很多现有的成熟技术加以实现，如有限状态机、状态转移矩阵、图标法等。

2.2 网格环境下基于上下文和角色的访问控制

这里我们使用状态转移矩阵来实现网格环境下基于上下文和角色的访问控制。下面是各状态转移矩阵的说明及由状态转移矩阵求出用户活动角色集、系统活动权限集和用户活动权限集的计算过程。

UR（用户、角色矩阵）是用户、角色分配关系的状态转移矩阵，以用户为行、角色为列。当用户分配了相应的角色时，对应的矩阵元素值为 1，否则为 0，随着用户、角色及分配关系的变化而变化（如当增加、删除用户时，增加、删除一行，当用户分配的角色改变时，修改相应元素值）；

RC（角色、主体上下文矩阵）是角色、主体上下文对应关系的状态转移矩阵，以角色为行、主体上下文为列。当角色在某主体上下文条件下被允许时，对应的矩阵元素值为 1，否则置 0，随着角色、上下文及对应关系的变化而变化；

RP（角色、权限矩阵）是角色、权限分配关系的状态转移矩阵，以角色为行、权限为列。当角色分配了权限时，对应的矩阵元素值为 1，否则置 0，随着角色和权限及分配关系的变化而变化；

PC（权限、客体上下文矩阵）是权限、客体上下文对应关系的状态转移矩阵，以权限为行、客体上下文为列。当权限在某客体上下文条件下被允许时，对应的矩阵元素为 1，否则为 0，随着权限、上下文及对应关系的变化而变化。

$USER\_ACT\_ROLES(u:USERS, c_i, c_j, c_k \cdots:CONTEXTS_{主})=\{r_m \mid UR[u,r_m]=1\} \cap \{r_i \mid RC[r_i,C_I]=1\} \cap \{r_j \mid RC[r_j,c_J]=1\}\{r_k \mid RC[r_k,c_K]=1\} \cap \cdots$

表示用户在主体上下文 $c_i, c_j, c_k \cdots$（如 c_i 为早上 9:00 至下午 5:00，c_j 为内部安全链接…）下的活动角色集，当用户处于多个主体上下文时，用户活动角色是在这些上下文下都处于有效状态的用户角色；

$SYS\_ACT\_PRMS(c_I, c_J, c_K \cdots:CONTEXTS_{客})=\{p_i \mid PC[p_i,c_I]=1\} \cap \{p_j \mid PC[p_j,c_J]=1\} \cap \{p_k \mid PC[p_k,c_K]=1\} \cap \cdots$

表示在客体上下文 $c_I, c_J, c_K \cdots$（如 c_I 为晚上 7:00 至早上 7:00，c_J 为 CPU 负载>80%…）下，系统中处于激活状态的权限集；

$USER\_ACT\_PRMS(u:USERS, \ c_i, c_j, c_k \cdots:CONTEXTS_{主}, c_I, c_J, c_K \cdots:CONTEXT_{客})=\{p_i:PRMS \mid RP[r_I, p_i]=1 \wedge r_I \in USER\_ACT\_ROLES(u:USERS, \ c_i, c_j, c_k \cdots:CONTEXTS_{主})\} \cap \{P_i \in SYS\_ACT\_PRMS(c_I, c_J, c_K \cdots:CONTEXTS_{客})\}$ 表示用户在主体上下文 $c_i, c_j, c_k \cdots$，客体上下文 $c_I, c_J, c_K \cdots$ 下的活动权限集。

在网格中每一个用户被分配一个角色集，叫用户角色集。当用户登录后，根据用户当时的主体上下文和 *RC*（角色、主体上下文矩阵）及用户角色集得到用户活动角色集，用户活动角色集随着用户主体上下文的改变而改变；同样，根据网格环境

（下转第 289 页）

Updating for C // Proceedings ACM SIGPLAN Conference on Programming Language Design and Implementation. Ottawa, Ontario, Canada: ACM Press, 2006: 72-83

[9] Dmitriev M. Safe Evolution of Large and Long - Lived Java Applications. PhD thesis. Scotland: University of Clasgow, 2001

[10] Orso A, Rao A, Harrold M J. A technique for dynamic updating of JAVA software // Proceedings of the International Conference on Software Maintenance (ICSM). Washington. D C, USA: IEEE CS, 2002: 649-658

[11] Baumann A, Heiser G. Providing dynamic update in an operating system // Proceedings of the USENIX Technical Conference. Anaheim, CA, USA: ACM Press, 2000: 279-291

(上接第 271 页)

的客体上下文和 *PC*(权限、客体上下文矩阵)可以得到系统活动权限集,系统活动权限集随着客体上下文的改变而改变。用户活动权限集是指由用户活动角色集继承的权限和直接分配的权限之和与系统活动权限集的交集,它随着用户活动角色集和系统活动权限集的改变而改变。

网格环境中基于上下文和角色的访问控制 CRBAC 模型原理如图 1 所示。

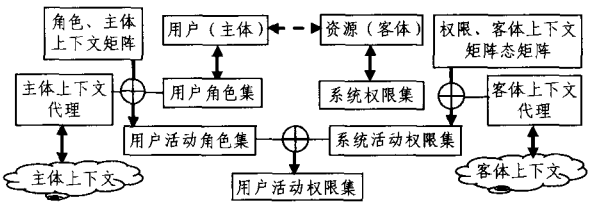

图 1 网格环境中基于上下文和角色的访问控制 CRBAC 模型

3 基于上下文和角色的访问控制 CRBAC 的案例分析

3.1 CRBAC 案例分析

在一个网格环境中有用户 $u1,u2,u3,u4$,角色 $r1,r2,r3,r4$,主体上下文 $c1,c2,c3$,权限 $p1,p2,p3,p4,p5$,客体上下文 $c1',c2',c3',c4',c5'$,*UR*(用户、角色矩阵)、*RC*(角色、主体上下文矩阵)、*RP*(角色、权限矩阵)、PC(权限、客体上下文矩阵)如下所示。

	r1	r2	r3	r4
u1	1	0	0	0
u2	0	1	0	0
u3	0	0	1	1
u4	0	0	0	1

UR(用户角色矩阵)

	c1	c2	c3
r1	0	1	1
r2	1	1	0
r3	1	0	1
r4	1	1	1

RC(角色主体上下文矩阵)

	p1	p2	p3	p4	p5
r1	1	0	1	0	1
r2	0	1	0	1	1
r3	1	1	1	0	0
r4	1	0	1	0	1

RP(角色权限矩阵)

	c1′	c2′	c3′	c4′	c5′	c6′
p1	0	1	1	0	1	1
p2	1	1	0	1	1	1
p3	1	0	1	1	0	1
p4	0	1	1	1	1	0
p5	1	1	1	1	1	1

PC(权限客体上下文矩阵)

若用户当前所处的主体上下文为 $c1$,客体上下文为 $c2'$ 和 $c4'$,系统的活动权限集为:

$SYS\_ACT\_PRMS(c2',c4')=\{p1,p2,p4,p5\}\cap\{p2,p3,p4,p5\}=\{p2,p4,p5\}$

用户 $u3$ 的活动角色集为:

$USER\_ACT\_ROLES(u3,c1)=\{r3,r4\}\cap\{r2,r3,r4\}=\{r3,r4\}$

用户 $u3$ 的活动权限集为:

$USER\_ACT\_PRMS(u3,c1,c2',c4')=\{\{p1,p2,p3\}\cup\{p1,p3,p5\}\}\cap\{p2,p4,p5\}=\{p2,p5\}$

当用户的主体上下文变为 $c2$ 客体上下文变为 $c3'$ 时,系统的活动权限集为:

$SYS\_ACT\_PRMS(c3')=\{p1,p3,p4,p5\}$

用户 $u3$ 的活动角色集为:

$USER\_ACT\_ROLES(u3,c2)=\{r3,r4\}\cap\{r1,r2,r4\}=\{r4\}$

用户 $u3$ 的活动权限集为:

$USER\_ACT\_PRMS(u3,c2,c3')=\{p1,p3,p5\}\cap\{p1,p3,p4,p5\}=\{p1,p3,p5\}$

从以上可以看出:主体上下文决定了用户的活动角色集,客体上下文决定了系统活动权限集,它们一起决定了用户的活动权限集。当用户的主体上下文和客体上下文改变时,用户活动角色集和系统活动权限集也随之改变,从而用户活动权限集也随之改变。

3.2 CRBAC 可行性研究

我们在标准 RBAC 模型的基础上加入了上下文的概念,将上下文分为主体上下文和客体上下文,主体上下文与用户主体环境相关,客体上下文与资源及服务的状态相关,没有冲突,主体上下文决定用户活动角色,客体上下文决定系统活动权限,两者一起决定用户活动权限集,是可行的。

结束语 基于上下文和角色的访问控制 CRBAC 模型在标准核心 RBAC 模型的基础上作了上下文方面的扩展,引入上下文后的访问控制 CRBAC 模型能根据环境上下文动态授权,解决网格环境中上下文敏感的访问控制。由于我们做的扩展是在核心 RBAC 模型的基础上,没有考虑角色继承,这方面还有很多工作需要进一步研究。

参 考 文 献

[1] Sandhu R, Conyne E J, Lfeinstein H, et al. Role - based access control models IEEE Computer, 1996, 29(2): 38-47

[2] 邓集波,洪帆. 基于任务的访问控制模型. 软件学报, 2003: 14(1): 76-82

[3] 陈伟鹤,殷新春,茅兵,等. 基于任务和角色的双重 web 访问控制模型. 计算机研究与发展, 2004: 41(9): 1466-1473

[4] 陈颖,杨寿保,郭磊涛,等. 网格环境下的一种动态跨域访问控制策略. 计算机研究与发展, 2006, 43(11): 1863-1869

[5] 黄建,卿斯汉,温红子. 带时间特性的角色访问控制. 软件学报, 2003, 14(11): 1944-1954

[6] Mossakowski T, Drouineaud M, Sohr K. A temporal-logic extension of role-based access control covering dynamic separation of duties Temporal Representation and Reasoning // 2003 Fourth International Conference on Temporal Logic, Proceedings. 10th International Symposium. 2003: 83-90

[7] Wullems C, Looi M, Clark A. Towards context-aware security: an authorization architecture for intranet environments // Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference. 2004: 132-137

[8] 陈华. 网格的访问控制模型. 微机发展, 2004, 14(8): 27-29

[9] 张纲,李晓林,游赣海,等. 基于角色的信息网格访问控制的研究. 计算机研究与发展, 2002, 39(8): 952-956

[10] 薛伟,怀进朋. 基于角色的访问控制模型的扩充和实现机制研究. 计算机研究与发展, 2003, 40(11): 1635-1642

[11] 孙为群,单保华,张程,等. 一种基于角色代理的服务网格虚拟组织访问控制模型. 计算机学报, 2006, 29(7): 1199-1208