基于使用控制和上下文的动态网格访问控制模型研究*)

崔永泉 洪 帆 龙 涛 刘 铭
(华中科技大学计算机学院 武汉 430074)

摘 要 网格环境动态、多域和异构性的特点决定其需要灵活、易于扩展和精细的授权机制。近来在网格环境下的访问控制方面做了大量研究,现有的模型大多在相对静止的前提下,基于主体的标识、组和角色信息进行授权,缺乏具体的上下文信息和灵活的安全策略。本文提出了网络环境下基于使用控制和上下文的动态访问控制模型。在该模型中,授权组件使用主体和客体属性定义传统的静态授权;条件组件使用有关的动态上下文信息体现了对主体在具体环境中的动态权限控制。在该模型的基础上,本文实现了一个原型系统,以验证模型的效率和易于实现性。

关键词 网格安全,访问控制,使用控制,上下文

Dynamic Context_aware Usage Control-based Grid Access Control Model

CUI Yong-Quan HONG Fan LONG Tao LIU Ming
(College of Computer Science, Huazhong University of Science and Technology, Wuhan 430074)

Abstract Due to inherent heterogeneity, multi domains characteristic and highly dynamic nature, grid environment requires scalable, flexible, and fine-gained access control mechanism. Despite the recent advances in access control for grid application do address important aspects of the overall authorization, these efforts focus on the pre-defined access control policies where authorization depends on identity or role of the subject. However, they are lacks of flexible approaches to adapt the dynamically security request. This paper proposes a dynamic context_aware usage control based grid access control model. In this model, authorization component evaluates access requests based on subject attributes, object attributes and requests. While condition component dynamic grants and adapts permission to the subject based on a set of contextual information collected from the user and system environments. As a proof-of-concept we design and implement a prototype system based on our proposed architecture and conduct experimental studies to demonstrate the feasibility and performance of our model.

Keywords Grid security, Access control, UCON, Context_aware

1 引言

网格是网络计算中一个具有重要创新思想和巨大发展潜力的研究方向。网格把地理上分散的多种资源集成为一体,将网络变成一个功能强大、无处不在的计算设施。网格计算的重要战略意义及其广阔的应用前景,使其成为众多研究人员和巨大资金投入的研究热点[1,2]。网格环境异构、动态和多域的特点决定了它需要易于扩展的、灵活的和精细的访问控制机制。

近来的研究者在网格安全方面做了大量研究,提出了网格安全基础设施 GSI 来解决认证和通信方面的安全问题。后来的一些研究者提出了 CAS 和 VOMS 等技术,但这些技术都集中关注相对静止的授权机制,没有考虑网格的动态特性。在高度动态的网格环境下,主体的访问能力不仅依赖于标识和角色信息,还与主体所处的具体上下文信息相关联,比如系统负载、网络带宽和访问时间等。主体的访问权限应该由静态授权和动态环境两个方面来共同约束。为了满足网格的动态访问控制需求,本文在下一代访问控制模型——使用控制模型的基础上,提出了基于使用控制和上下文的动态网格访问控制模型。文章第 2 节对相关研究工作进行了回顾。第 3 节提出了三个需要实施动态访问控制的应用情景。第 4 节详细描述了形式化的基于使用控制和上下文的动态网格访问控制模型。第 5 节给出了原型系统的体系结构和核心框架,第 6 节对原型系统的测试进行了总结和分析。最后引出了本文的结论。

2 相关研究工作和进展

在访问控制研究几十年的历史上,也曾经出现过一些非常著名的模型。比如矩阵模型、格模型和基于角色访问控制(RBAC)模型[3,4]。随着信息技术的发展,访问控制也出现了许多新的需求。使用控制(UCON)模型从义务组件、条件组件、控制时机和属性等多个方面对 RBAC 模型进行了扩展,富有潜力的下一代访问控制模型[5~7]。

在网格访问控制方面也取得了下列进展。Globus 网格项目是最为著名的网格项目之一。它在集成 Kerberos 和 PKI 技术的基础上提出了 GSI[8] 解决认证和通信的安全问题。Kesselman 和 Foster 等人提出了面向 VO 内用户和权限管理的 CAS[9,10]。其核心思想是为 VO 内所有用户进行集中式的授权和管理。与 CAS 类似的是 VOMS[11] 技术,两者的主要区别在于 CAS 颁布的用户授权保证 Assertion 中直接保

*)本课题得到国家自然科学基金(60403027)、湖北省自然科学基金(2005ABA243)资助。崔永泉 博士研究生,主要研究方向为分布式系统安全、访问控制;洪 帆 教授,博士生导师,主要研究方向为信息安全、访问控制、网络安全等。

存用户对某个资源的访问权限信息，而 VOMS 颁布的 Assertions 中是用户所属的组别和角色信息，在资源提供者实施访问控制时，将组别或角色信息再转换为具体的访问权限。另一类面向资源的管理技术如 Akenti[12] 和 Permis[13]，主要通过属性证书、条件证书和 PMI 技术的结合，解决分布式环境下实现资源的有效访问，在网格环境的资源管理中有一定的借鉴作用。上述的解决方案主要体现在预先对主体进行静态的授权，而忽略了网格环境的动态特征。

网络环境中高度动态变化的特性也引发了研究者的关注，上下文信息也成为授权中需要重视的参考因素[14,15]。G. Zhang 和 M. Paradom 等人提出了基于角色和上下文的动态访问控制模型[16]，其核心思想是为用户维护一个角色状态机，上下文信息的动态变化会触发主体的角色发生变化，从而对主体的访问权限进行动态的调整。该模型的弱点就是角色状态机难于管理和维护，而且不能支持灵活的扩展和大规模的角色层次。

3 网格访问控制中的动态需求背景

建立网格的主要目标之一就是为了有效聚合分散在多域的资源来完成某项特定任务。网格应用通常涉及许多实体和资源动态的加入和合作。比如，从遍布世界的跨国公司和大型研究机构的工程师可以在一个虚拟组织(VO)内共同协作设计、开发和测试检验某项新产品。在协作过程中，各个参与者需要共享某些资源。而资源提供者希望在安全策略的保护下提供资源的共享。本文总结了三种需要实施动态访问控制的情形。

· 系统资源的负载能力总是有限的，如果有多个工程师同时访问某个资源，则该资源的系统负载就会很高。那么访问控制应该能够提供某种机制，保证优先为某些关键的用户提供服务。当系统负载下降时，再为其他的用户提供服务。

· 某个主体需要建立用户代理 Proxy 来支持长时间运行的程序。程序开始运行的时候，资源提供者会检查用户 Proxy 证书的有效性。当程序运行一段时间以后，如果 Proxy 证书过了有效期，则系统应该能够提供某种机制，终止该用户进程的执行。

· 主体可能从遍布世界的地点访问 VO，如果某个主体处于非加密的网络连接状态下提出访问请求，资源提供者将会拒绝对某些机密信息的访问请求。

4 基于使用控制和上下文的动态网格访问控制模型(DC_GUCON)

传统的基于标识和角色信息的静态授权方式已经不能适应动态的安全目标，本文提出网格中动态访问控制模型，它能够根据主体静态属性和动态的上下文信息对主体的访问权限进行控制和调整。

4.1 使用控制模型(UCON)

传统访问控制模型，比如自主访问控制、强制访问控制和基于角色访问控制，一个突出的局限性就是仅仅关注访问控制的授权过程。使用控制模型，除了授权组件以外，还包括义务组件和条件组件。当主体提出访问请求时，授权(Authorization)组件将根据主体请求的权限，检查主体和客体的安全属性，比如主体身份、角色、安全类别和客体的安全标签等，从而决定该请求是否被允许。义务(oBligation)就是在访问请求执行以前必须由义务主体履行的行为，比如用户需要从某个网站下载某项产品使用手册时，该网站要求用户必须进行注册。另外，访问请求的执行必须满足某些系统和环境的约束，比如系统负载、访问时间限制等，这些就称为条件(Condition)组件。在目前高度动态的网格环境下，义务和条件也是决定访问请求是否能够被允许的至关重要因素。

在传统的访问控制模型中，授权决定仅仅在主体提出访问请求时起检查作用，而在权限执行的进程中没有任何限制。使用控制模型对控制时机进行了扩展，控制决定不但在提出访问请求时实施检查，而且在访问执行的过程中也会进行控制。

4.2 DC_GUCON 模型的核心思想

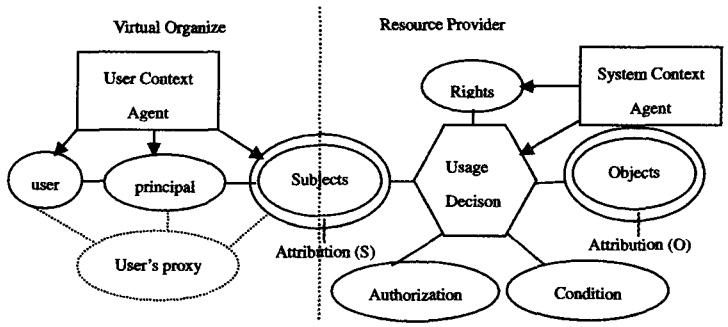

图 1 DC_GUCON 模型组件

图 1 的右边显示了资源提供者(RP)的域内典型的以系统为中心的使用控制模型。该模型主要包括主体、客体、主客体属性、权限、授权和条件组件，我们还引入了系统上下文代理。图 1 的左边显示了 VO 内其他一些实体的抽象关系。在经典的访问控制术语中，“User”是系统日志的单元，“Principal”是系统进行安全认证的单元，“Subject”是系统进行访问控制的单元。

在 VO 内，用户属性在认证的过程中部分传递给参与者，认证以后进一步传递给主体进行访问控制。用户上下文代理会为参与者和主体收集有关的上下文信息。为了支持长期运行的程序，用户一般会创建用户代理 Proxy，如图中虚线所示，并且将全部或部分的权限委托给 Proxy，由 Proxy 代替用户监控程序的执行。图的左边提供了一个用户在整个 VO 内向 RP 域的一个抽象的映射关系。

4.3 DC_GUCON 模型的定义

形式化的 DC_GUCON 模型建立在传统的 RBAC 模型的基础上，许多组件和 RBAC 模型中的相同，另外有一些组件是根据网格环境的安全需要对原有组件的扩展。

• USERS，ROLES，SUBJECTS，OBJECTS，RIGHTS，PERMS集合的定义来自于RBAC模型，它们分别代表用户、角色、主体、客体、权限、授权许可的集合。

• UA和PA也来自于RBAC模型，它们分别代表用户角色指派和权限角色指派。

• $ROLES_G$。该集合代表VO内全局的角色层次集。在DC_GUCON模型中，使用一个CAS维护一个全局的角色层次集。当用户需要访问VO内的某项资源时，CAS首先对其信任证书进行验证，然后根据VO全局的安全策略，赋予用户某个VO的角色。

• $ROLES_{Li}$。该集合代表RP域内的角色层次集合。在本文的授权模型中，首先由CAS给用户赋予某一个VO的角色。但资源提供者并没有将本地的访问权限直接关联于VO的角色上，而是将VO的角色映射到本地的某个角色上。所以，决定将VO角色映射到具体本地角色子集的权限是由资源管理者所掌握，因此该模型很好地体现了资源提供者对本地资源的自主控制。

• $ROLES = (\cup ROLES_{Li}) \cup ROLES_G$，$i=1,2,\cdots,n$。整个模型的角色集合是由VO的角色层次和各个RP域内的角色集共同构成的。

• $RM_i \subseteq ROLES_G \times ROLES_{Li}$ $i=1,2,\cdots,n$。它代表从VO角色到本地角色集的一个映射关系。

• LEVEL。该集合代表了一个主体的安全级，它表示了主体的重要程度。

• UC_CONS。该集合代表了主体的上下文信息，比如用户的位置、提出访问请求的时间、用户proxy证书的有效期限等。我们使用“用户上下文代理”来收集有关主体的上下文信息。

• SC_CONS。该集合代表了系统的上下文信息，比如网络的带宽、系统的负载情况、系统的安全级别、当前的访问时间等。我们使用“系统上下文代理”来收集有关系统的上下文信息。

• C_CONS=UC_CONS∪SC_CONS。模型中上下文信息集合包括用户上下文信息和系统上下文信息，上下文信息的变化会触发系统授权决定的变更。

• IDENTITIES。该集合代表了主体的标识。

定义1 模型中的映射定义如下。

sRA：$S \rightarrow 2^R$，代表主体经过从VO角色到本地角色的RM_i变换，所拥有的角色子集。

actR：$S \rightarrow 2^R$，代表在一次会话中仅仅激活用户所拥有角色的子集。

sID：$S \rightarrow SID$，主体到标识集的内射。

sL：$S \rightarrow L$，代表主体到一个安全级的映射。

pR：$P \rightarrow 2^R$，代表授权许可到本地角色集的一个映射，可以看作将对一个对象和操作权限共同赋予某些角色。

定义2 模型中的主体、客体属性和有关上下文信息定义如下。

$Atrri(S) = \{sRA, actR, sID, sL\}$

$UC\_CONS(S) = \{sLink, sAccessTime, sExp\}$

$Atrri(O) = \{pR\}$

$SC\_CONS(env) = \{sysload, sysSec, curT\}$

定义3 模型中对主体以某种权限访问客体的授权组件定义如下。

$P=\{(o,r)\}$

$Role' = \{role \mid \exists r1 \in sRA(s), \exists r2 \in pRA((o,r)), r1 \geqslant role \geqslant r2\}$

$Role'' = \{role \mid \exists r1 \in actR(s), \exists r2 \in pRA((o,r)), r1 \geqslant role \geqslant r2\}$

$Allowed(s, r, o, uc\_cons) => Role' \neq \phi$

$$actR(s) = \begin{cases} actR(s) & \text{if } Role'' \neq \phi \\ act(R)(s) \cup \Gamma(Role') & \text{if } Role'' = \phi \end{cases}$$

在RBAC模型中，授权许可是(对象、权限)的序偶，UA和PA代表用户指派和权限指派。在DC_GUCON模型中，对于UA而言，角色可以看作主体的属性，即一个主体可能有多个角色；对于PA来说，角色可以看作对象o和权限r所共有的属性，即一个对象o可以有多个角色以权限r的方式去访问。因此，如果一个主体可以访问对象o，则要求主体必须至少存在一个角色可以支配客体o所属的角色子集中某一个角色，即Role′不能为空集。

在一次会话中，可能主体只是激活了其中的某些角色，如果活动集中不存在支配角色，即Role″为空集，则r表示从Role′中随机选择一个角色，添加到活动角色集中。

定义4 模型中对主体以某种权限访问客体的条件组件定义如下。

$Condition(s, r, o, uc\_conditions(s)) = CN1 \cap CN2 \cap \cdots \cap CNn$

$CNi = \langle CT \rangle \langle OP \rangle \langle Value \rangle$

$CNi(i=1,2,\cdots,n)$是一系列的规则表达式，$CNi = \langle CT \rangle \langle OP \rangle \langle Value \rangle$，CT是一些需要实施控制的约束条件，OP可以是逻辑操作符$\{=,>,<,\geqslant,\leqslant,\neq\}$，也可以是用户自己定义的操作符，VALUE是管理员为CT设定的阈值。CNi用来比较CT的当前值和设定的CT阈值，返回一个布尔型的值，满足条件返回真，否则为假，计算CNi的过程成为评估主体s是否满足Condition组件的过程。

举例来说，我们可以定义如下的条件组件：

$$Condition(s,r,o,uc\_cnd) = \begin{cases} (sLink = \text{'enc'}) \wedge (curT < sExp), & \text{if } sL \geqslant \text{'Critical'} \\ (sLink = \text{'enc'}) \wedge (curT < sExp) & \text{if } sL < \text{'Critical'} \\ \quad \wedge (sysLoad < \text{'80\%'}) \wedge (sysSec > \text{'high'}), & \end{cases}$$

该条件组件的定义说明，根据主体的安全级不同，条件组件可以分为两类。当主体是比较重要的用户时，即安全级大于等于“Critical”，那么主体应该满足下列两个要求：一个是主体的网络链接必须是加密“Enc”状态；另外一个约束条件是主体的证书不能超过有效期。只有这两个条件同时满足时，条件组件判断的结果才能为真，则主体的访问请求才能被批准，否则主体的访问请求将会被拒绝。

当主体是不太重要的用户时，即安全级小于“Critical”。主体除了满足上述的两个约束以外，又添加了两个约束，即系统的负载小于80%而且系统的安全级为“high”的时候才能通过条件组件的检查。

从上面的条件组件看，如果系统负载很高，高过80%以后，则非关键的主体将不能通过条件组件的检查而被拒绝访问，而对于关键的主体而言，没有这个限制。

定义5 模型中的持续条件组件的检查和实施定义如下。

$allowed(s, r, o, uc\_cond(s)) => Ongoing\_CondChecked(condition(s, r, o, uc\_cond(s)))$；

对于一个持续执行时间很长的用户访问过程而言，如果

系统的上下文信息发生了变化，或者系统经过一段固定的时间周期以后，会触发条件组件的持续性检查。Ongoing_CondChecked 将会返回一个持续检查的结果。如果结果为真，则用户访问过程继续执行，否则将会终止用户的访问过程。比如，用户 Proxy 的证书过期，则会导致程序被终止运行。

5 DC_GUCON 模型在原型系统中的实施框架

5.1 原型系统的体系结构

DC_GUCON 模型已在实验环境中得以实施，图 2 给出该原型系统中实施 DC_GUCON 模型的总体结构。

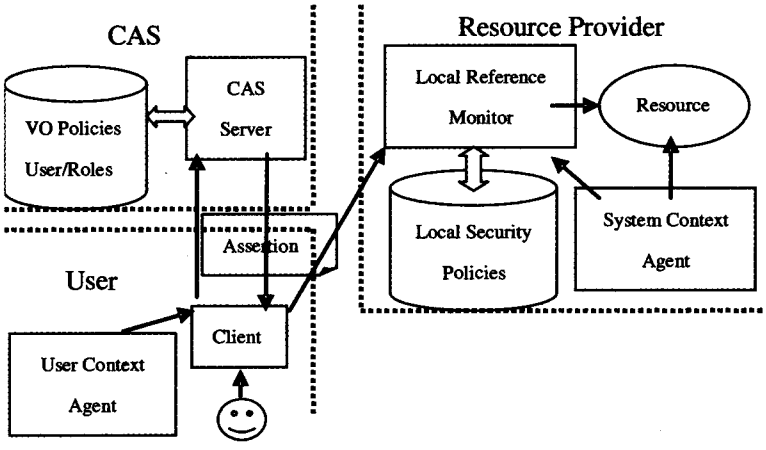

图 2 DC_GUCON 模型在原型系统中体系结构图

如图 2 所示，用户在 VO 内访问资源的步骤如下。

1) 用户首先登录 Client 程序，Client 程序首先访问 CAS 服务器，CAS 验证用户的信任证书。通过验证以后，CAS 在 VO 安全策略库中寻找与用户有关的用户角色指派，为用户产生授权保证 Assertion，返回给 Client 程序。

2) 用户上下文代理收集用户当前环境中的信息，加上主体的访问请求和从 CAS 接收的 Assertion 保证，一起递交给资源提供者。

3) 资源提供者的核心部件是访问监控器。它首先验证主体提供的 Assertion，从中提取出需要的 VO 全局角色，然后根据本地策略库的有关记录，将 VO 角色转化到本地角色。然后授权组件将根据主体的属性、客体的属性和访问请求，判断是否满足授权组件的要求。

4) 系统上下文代理收集系统环境有关的上下文信息，加上从客户端接收主体上下文信息一起送给条件组件检查，条件组件根据预先定义的上下文判断规则，判断条件组件是否得以满足。

授权组件和条件组件都得到满足的情况下，主体的访问请求才会得以批准。

在程序执行的过程中，上下文信息的变化也会触发条件组件的持续检查。如果不满足条件组件，进程的访问过程将会被终止。

5.2 访问监控器的概念模型

从系统结构的观点看，实施 DC_GUCON 模型最关键的部件就是资源提供者域内的访问监控器(Reference Monitor)。ISO 制订了有关访问监控器的标准配置和可信计算的基础框架[16]。根据该标准，访问监控器由访问实施设施(AEF)和访问决定设施(ADF)组成。每一个访问请求被 AEF 接收，AEF 向 ADF 提交，由 ADF 决定是否该请求被允许执行。访问监控器是可信计算的核心基础部件，一直运行在系统中，而且不能被绕过。

本文根据 DC_GUCON 模型的需要，如图 4 所示，在 ISO 标准结构的基础上提出了一个新的实施 AUCON 模型的安全结构，并且把 AEF 和 ADF 分成了若干个不同的组件。UEF 接收主体对客体的访问请求并把请求转发给 UDF 设施。授权组件将会像传统的访问控制授权过程一样，检查主体属性、客体属性和相关的权限，确定是否允许该请求访问系统。条件组件将检查有关的上下文信息，判定预定义的条件是否得到满足。条件组件的满足将在监控模块下进行监督和实施。条件组件的检查结果也会影响到访问请求最终是否得到批准。最后批准或者拒绝请求的结果信息由 AEF 返回给提出请求的主体。

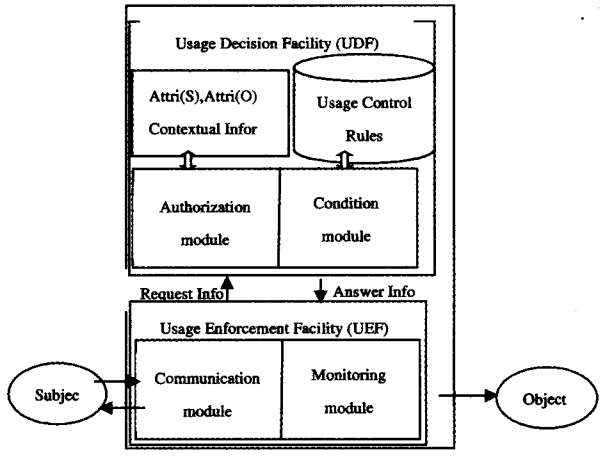

图 3 AUCON 模型访问监控器的体系结构

6 原型系统运行和测试结果分析

我们原型系统的实验环境和机器配置如下。由三台 PIV-2.1G 的主机使用 100M 的网络连接。三台主机的内存都是 512M，硬盘 60G，安装的操作系统是 Redhat linux9，内核版本为 linux2.4-20，它们分别充当 CAS 服务器、客户端 client 主机和 RP 域资源服务器。

其中 CAS 服务器安装的程序有 Apache、OpenSSL、OpenCA、stunnel 和 Mysql 数据库系统，我们在 openssl 的基础上编写了一个产生 VO 用户角色指派的服务程序，用户角色指派信息保存在 mysql 数据库中。客户端主机安装的软件

有模拟 client 程序、Openssl、stunnel 和用户上下文代理模拟程序。RP 域内服务器安装软件有 Apche、Openssl、stunnel、Mysql、RefrenceMonitor 监控程序和系统上下文代理模拟程序。

用户登录 Client 程序，CAS 服务器验证用户的证书，建立 stunnel 安全加密通信信道，CAS 服务器返回给 Client 程序一个带有 VO 用户角色指派的 X.509 扩展证书，称为 Assertion。Client 发送上下文信息、Assertion 和打开服务器上某一个指定文件的服务请求。RP 内的服务器监控程序验证 Assertion，提取有关 VO 的角色，转换为本地角色并且实施授权和义务组件的检查。然后打开 client 指定的文件。

我们就按照上述过程分别进行了两组实验，第一组实验按照标准的 CAS 技术，Assertion 中直接保存用户对指定文件的访问权限；第二组实验按照 DC_GUCON 的模型的标准实施访问请求。每一组实验分别打开 10kB、100kB、1000kB 和 10000kB 大小的文件，然后对 client 程序从登录 CAS 服务器到实际打开服务器文件的响应时间求平均值。最后平均的实验数据如图 4。

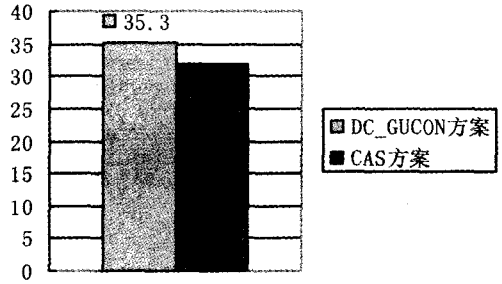

图 4　原型系统测试数据结果对比图

从图 4 的系统测试结果来看，DC_GUCON 方案的消耗的时间比 CAS 消耗的时间多 11%。相比较而言，性能并没有显著降低。说明真正 RefrenceMonitor 访问监控程序消耗的时间在整个访问过程中并不占有最重大的比重，时间还要消耗在 CAS 服务程序、Client 程序、RP 域三者之间的网络连接、证书验证和交换数据等中间过程上。

结论　为满足网格应用中复杂的访问控制需求，本文提出基于使用控制和上下文的动态网格访问控制模型，该模型中授权组件使用主体和客体属性，比如角色信息、主体标识等，控制传统的静态授权；条件组件可以根据主体上下文信息和系统上下文信息对主体的访问权限进行动态调整和控制。该模型具有以下优点：①模型能够动态实施，具备很高的灵活性。在设计时，资源提供者可以灵活的方式定义复杂的上下文敏感授权策略；在系统运行时，基于上下文信息的动态变化，系统可以对主体的权限做出相应的调整。②模型易于扩展，因为 Condition 组件是一组规则表达式，上下文信息与应用程序无关，可以根据系统的安全需求，非常容易的添加或者修改规则表达式，很方便地实施扩展。③模型引入了 VO 角色到本地角色的映射关系，如果资源提供者不愿意继续提供资源的共享，只需要删除 VO 角色到本地角色的映射关系，本地安全策略无需改变，最大限度体现了资源管理者的自主控制权限。

原型系统的运行和测试表明该模型能够非常容易的部署和实施，而且相比传统的 CAS 技术而言，性能不会受到很大的影响。

参 考 文 献

1 Foster I, Kesselman C, Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organization. Int'l Journal of Supercomputer Applications and High-performance Computing, 2001, 15(3): 200～222

2 Foster I, Kesselman C, Nick J M, et al. Grid Services for Distributed System Integration. IEEE Computer, 2002, 35(6): 37～46

3 Sandhu R, Coyne E, Feinstein H, et al. Role-based access control models. IEEE Computer, February 1996, 29(2)

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

5 Sandhu R, Park J. Usage control: A vision for next generation access control. In: Proceedings of The 2nd International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security. 17～31

6 Park J, Sandhu R. Towards usage control models: beyond traditional access control. In: Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies, ACM Press. 57～64

7 Park J, Sandhu R. The $UCON_{ABC}$ Usage Control Model. ACM Transactions on Information and Systems Security, Feb. 2004, 7(1): 128～174

8 Kesselman F C, Tsudik G, Tuecke S. A Security Architecture for Computational Grids. In: Proceedings of the 5th ACM Conference on Computer and Communications Security, San Francisco, CA, USA, 1998. 83～92

9 Pearlman L, Welch V, Foster I, et al. A Community Authorization Service for Group Collaboration. In: Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002

10 Cannon S, Chan S, Olson D, et al. Using CAS to Manage Role-based VO Sub-Groups. In: Proc. of Int'l Conference for Computing in High Energy and Nuclear Physics, 2003

11 VOMS Architecture v1.1. http://gridauth.infn.it/docs/VOMS-v1_1.pdf, May 2002

12 Thompson M, Johnston W, Mudumbai S, et al. Certificate-based Access Control for Widely Distributed Resources. In: Proceedings of the Eighth Usenix Security Symposium, Aug 1999

13 Chadwick D, Otenko A. The Permis X.509 Role Based Privilege Management Infrastructure. In: Proceedings of SACMAT 2002 Conference, ACM Press, 2002. 135～140

14 Bouquet P, Serafini L, Brezillon P, et al. Modelling and Using Context. In: Proceedings of the 2nd International and Interdisciplinary Conference, CONTEXT99, Lecture Notes in Artificial Intelligence, Vol 1688. Springer Verlag, 1999

15 Henricksen K, Indulska J, Rakotonirainy A. Modeling Context Information in Pervasive Computing Systems. In: Proceedings of the 1st International Conference, Pervasive 2002 - Zurich August 2002, Lecture Notes in Computer Science, Vol 2414. Springer Verlag, 2002. 167～180

16 Zhang G, Parashar M. Dynamic context-aware access control for grid applications. In: IEEE Computer Society Press, editor. 4th International Workshop on Grid Computing (Grid 2003), Phoenix, AZ, USA. 2003. 101～108

17 ISO. Security frameworks for open systems: Access control framework: [Technical Report]. ISO/IEC 10181-3. 1996