第 29 卷 第 7 期
2006 年 7 月
计 算 机 学 报
CHINESE JOURNAL OF COMPUTERS
Vol. 29 No. 7
July 2006

SCoAC：一个面向服务计算的访问控制模型

曹 春 马晓星 吕 建
（南京大学软件新技术国家重点实验室 南京 210093）
（南京大学计算机科学与技术系 南京 210093）

摘 要 提出了一个面向服务计算的访问控制模型 SCoAC. 在该模型中服务实体间的交互被看作是双方平等地服务于应用系统，通过环境中各实体间关系的描述，表达了服务间以所在管理域间信任关系为基础、以所处应用上下文为依据的授权关系，并通过引入“绑定上下文”匹配机制为细粒度策略的实施提供了支持. 面向服务的应用系统的动态变化在模型中得以表达，并直接反映到服务间的授权关系之上.

关键词 服务计算；访问控制；绑定上下文；服务交互；虚拟企业
中图法分类号 TP311

SCoAC：A Service Computing Oriented Access Control Model

CAO Chun MA Xiao-Xing LU Jian
(*State Key Laboratory for Novel Software Technology*, *Nanjing University*, *Nanjing* 210093)
(*Department of Computer Science and Technology*, *Nanjing University*, *Nanjing* 210093)

Abstract To protect the services against illegal accessing, misusing and tampering is the essential problem in service oriented computing paradigm. As existing access control models and mechanisms can hardly meet the requirements of securing the services in the SOC environment completely, an access control model SCoAC is proposed in this paper. Interactions happening between services are viewed as the contributing processes from both sides to the application system. By specifying the relationship among the entities in the system, the model expresses the authorization for services capturing the trust relationship between their administration domains as well as the application context. This paper also introduces a BindingContext matching mechanism to support fine-grained access control. The evolution of application systems can be mapped onto the changing of the authorization status for the services effectively.

Keywords service computing; access control; BindingContext; service interaction; virtual enterprise

1 引 言

新兴的面向服务的计算模型(Service Oriented Computing, SOC)将信息资源、计算资源等封装为服务(基本服务被独立地开发、部署、发布)，通过查询、绑定等过程，将适当的服务组合到一起并定义、约束其相互间的交互行为，从而实现复杂应用系统

收稿日期：2006-02-17；修改稿收到日期：2006-04-20. 本课题得到国家“九七三”重点基础研究发展规划项目基金(2002CB312002)、国家“八六三”高技术研究发展计划项目基金(2005AA113160，2005AA113030，2005AA119010)和国家自然科学基金(60233010，60403014)资助. 曹 春，男，1978 年生，博士研究生，主要研究兴趣为访问控制技术、服务计算、软件 Agent 技术. E-mail：caochun@ics.nju.edu.cn. 马晓星，男，1975 年生，博士，副教授，主要研究兴趣为 Internet 软件技术、软件体系结构. 吕 建，男，1960 年生，博士，教授，博士生导师，CCF 高级会员，主要研究领域为软件自动化、并行程序形式化方法、面向对象语言和环境.

的快速开发和部署[1,2]. 这一面向服务的体系结构(Service Oriented Architecture, SOA)成为了新一代的软件系统开发模式.

保护服务资源使其免受非法访问、滥用或篡改等是服务计算模式下安全机制所需解决的首要问题. 在通过定义并实施安全策略来防止对服务实体非法访问以保护服务所封装资源的同时,参与同一应用系统的各成员服务实体之间又需共享其所封装的资源,以保障应用系统的正确运行. 例如,对于一个虚拟企业[3,4]而言,其业务过程突破了其自身边界、部分或全部依赖于其它企业,相应的业务系统则通过整合各个合作伙伴提供的服务构建而成. 这样一个业务应用系统能以较低成本实现快速开发、部署,服务与服务之间接口的松耦合性又使得系统能便捷地更新、替换某些功能部件以灵活适应不断变化的环境. 各合作伙伴所提供的服务表达了企业业务逻辑,提供了企业内部资源,对这些服务的保护关系到企业的直接现实利益. 将服务对属于同一虚拟企业的其他成员开放是虚拟企业业务过程顺利执行的前提条件,同时必须将访问限定于特定的业务过程上下文. 也就是说,在服务计算模式下,安全策略表达为仅允许对应用系统运行而言必要的服务间交互发生,籍此确保服务实体的私密性和完整性,实施此策略的访问控制机制需根据服务所处的应用系统上下文做出服务间能否交互的决策.

Web 服务技术作为目前服务计算面向 Internet 平台的一个具体实现,沿用了传统的基于主体标识的访问控制模型;而另有的一些考虑访问上下文的模型并不完全适用于服务计算应用系统(详见下文). 因此,本文提出了一个面向服务计算的访问控制模型. 在该模型中,从应用系统的角度出发,我们将服务间交互看作是双方平等地服务于应用系统的一个过程,而非直接地调用与被调用,进而从三个层次表达对交互合法性的判定:(1)服务所属的管理域间是否存在信任关系;(2)服务间交互是否具有合法的应用系统上下文;(3)交互方法的个体属性是否满足具体规则. 一方面,该模型符合一般的应用系统开发部署流程:服务被集成的先决条件是服务提供方与系统开发方之间存在相互信任关系,只有信任关系确立后,服务提供方才能提供服务,而系统开发方才愿意使用该服务(以保障应用系统包括安全性在内的各项非功能属性);另一方面,服务所处应用上下文将决定服务间具体交互过程的合法性,该上下文包含有多个层面的信息,包括应用实例信息、服务实体在应用中所服务的功能需求、服务被集成到应用系统时具有的个体属性等. 通过这个访问控制模型,可表达面向服务计算模式中安全策略,并可指导具体访问控制机制的实现.

本文第 2 节介绍相关工作并分析服务计算模式对实施访问控制的新要求,阐述本文工作的动机;第 3 节具体介绍面向服务计算的访问控制模型 SCoAC;第 4 节通过一个实例说明该模型在服务计算场景中的应用;第 5 节我们讨论模型设计中的一些问题和今后的工作;最后一节总结全文.

2 问题及相关工作

服务计算利用已有服务作为基本元素开发应用软件系统,整合联系于已有服务的背后资源和计算逻辑为其所用,而其关键在于如何协同参与实体间的行为使得这种整合的最终结果满足系统的应用需求. 由多个服务组合而成的应用系统的运行具体表现为协同各个成员服务的行为,也就是控制服务的执行,管理其间交互[2].

访问控制技术通过限定合法交互对象保护服务资源不受非法访问. 作为目前服务计算面向 Internet 环境的一个具体实现,Web 服务技术通过制定一组规范提供了在多个层面上保护 Web 服务的安全机制:WS-Security 用以实现消息安全性;WS-Policy 定义了服务端的安全策略的表述形式;WS-Trust 描述了如何在 Web 服务环境中建立企业之间的信任关系以支持域间主体标识的映射;SAML 语言被用以描述在服务间传递的认证和授权等信息[5]. 这些规范将已有的安全机制进行标准化,而访问控制机制则仍建立在传统的基于主体标识的访问控制模型之上. 实体经过认证或一组(属性)规则映射到某个主体标识,根据这一标识对其访问进行限制. 此类授权及访问控制方式可实现服务资源的简单发布,即直接定义服务的合法访问者,但这一定义过程缺乏指导,完全依赖于管理者对环境内各个实体的认知,而忽略了对其它层次的信息,例如实体所属团队、组织,访问发生的上下文等. 访问控制模型中缺乏这些高层信息导致高层安全策略无法直接定义并获得实施,而是需要先翻译为底层的主体标识的映射规则和主体标识的访问规则,当高层策略发生变化时还可能导致大批底层规则的变动,带来较重的管理负担.

单纯以主体标识为授权依据的方式并不与服务

计算的理念相一致. 服务计算模式中，各个基本服务的设计、实现、部署处处体现“可被组合”的理念，其目标是要能与其它服务通过组合方式共同服务于某个应用系统. 从安全角度考虑，一方面，在确定的应用系统中，服务的访问模式、用法(usage)和信息流向等均会相对明确；另一方面，对服务的访问被限定于某些特定的应用系统内，符合“need-to-know”的安全原则，访问者只有在一些明确的场景下或在执行某些确定的活动时，才能使用与该场景或该活动相关的资源. 也就是说，访问者和被访问者所处的应用上下文都将决定该次访问能否进行. 相应地，在服务计算模型中服务实体的部署、运营模型也将显现出这一特点. 我们相信，除了服务体的开发和部署方外，还会出现独立的服务运营方. 运营方通过开发或限定服务所能参与的应用系统，定义服务的应用场景或访问模式. 只有在特定的应用上下文条件下，访问者才能获得访问权限，而被访问者才能提供其服务. 因此，服务所处的应用上下文应被显式表达并作用于访问控制决策.

在访问控制模型中引入非主体标识这一思想在多个已有的模型中被应用. 例如，TBAC[6,7] 中通过 Task 定义用户在某项任务中的权限，并随着用户所执行任务的不断变化，动态地管理用户的有效权限. TMAC[8] 用团队(team)这一抽象实体来支持应用场景中对协作用户群体安全策略的表达需求，当用户处于某个协作团队时，他将被额外授予该团队所具有的权限. Cohen 等提出的 CBAC[9] 模型面向联盟协作环境建立，联盟由多个合作组织组成，并被赋予某些使命. 以完成这些使命为目的，组织拥有的资源被授权执行这些使命的组织成员访问. 通过组织与联盟、个体与组织等关系以及联盟使命、个体任务、所属团队等上下文信息决定用户的授权状态，进而实施对其的访问控制. 该模型中，主体与客体间关系通过多层抽象实体桥接，明确有效地表达了系统的组织状态，增强了所定义规则的易理解性并减轻了管理负担.

根据前文所分析，服务计算环境中各基本服务间的授权应以交互发生的应用上下文为依据，访问控制模型需参照 TBAC 和 TMAC 等引入显式的应用上下文元素；类似于 CBAC 的应用场景，安全管理域的组织模型及其间的直接或间接信任关系也应在访问控制模型中得以反映，并作为域间实体访问授权的基础. 在综合已有这些访问控制模型思想的基础上，我们在建立面向服务计算的访问控制模型时还需考虑该新型计算模型的其它特点. 首先，面向服务的应用系统本身可只定义抽象的应用逻辑或业务过程，对外部具体功能的需求可显式声明，并通过集成绑定多个成员服务来完成一个具体应用的构建. 因此，从应用集成角度来看，按应用逻辑定义发生交互的双方都服务于该应用系统，这种交互过程不应被看作是通过方法调用完成的一方为另一方提供服务的过程，交互双方应被看作通过同样的形式对等地服务于应用系统. 判断两者能否交互不再像 CBAC 等模型中一样判断一方是否具有操纵另一方的权限，而是判断两者是否联系于同一应用系统以及此次交互是否是为这一应用系统运行而进行的. 其次，应用系统的对外服务需求声明类似于 CBAC 中将一个“使命”分解为多个子任务，所不同的是，服务计算中的交互发生于满足各功能需求的服务实体之间，而非将访问主体与被访问资源归并到同一子任务中实现对主体的授权. 服务实体间的双向对等关系决定了访问控制机制须通过考察双方各自的属性得到决策，而非单一地访问主体所具属性(如被赋予角色等). 此外，对交互双方的细粒度访问控制的支持也是必需的. 例如在某个应用中，交互发生于信息生产者与信息消费者之间，要求信息仅能被安全级别不低于生产者的消耗者所接收，则安全机制须对交互双方的属性细节加以评估.

归纳而言，对于成员服务间的访问控制，需考察它们之间的三层关系：(1)其间的组织结构关系，即它们之间是否存在随应用系统构建而建立的信任关系以作为对交互授权的基础；(2)这两者的交互是否定义于某一应用系统，也就是它们之间是否处于相同的应用上下文中；(3)需确定交互双方是否满足进行交互的具体要求.

前述的访问控制需求可通过一个简单的实例来说明. 旅行社(CyberTA)开发了一个旅行预定业务系统(CyberTrip)，该业务系统接收客户旅行需求，为客户安排旅行计划并预订机票、房间. 由于预订机票、酒店等业务过程需要通过较为复杂的协议(包括从银行获得信用卡授权及服务契约协商等)完成，CyberTA 使用由代理公司(AllInOneAgent)提供的 $S_{\mathrm{TScheduler}}$ 服务执行具体的预订机票和房间的流程. 这一应用系统还需根据客户要求选择具体航空公司和酒店，并将这两者的预订服务指定给 $S_{\mathrm{TScheduler}}$ 以完成相应预订程序. 假设客户指定乘坐 WorldAir 公司的飞机并下榻于 BeachHotel，CyberTrip 与这两个商业实体间需建立信任关系或订立协议. 在这个

例子中，$S_{TScheduler}$ 被获准分别同 WorldAir 的 $S_{AirBook}$ 及 BeachHotel 的 $S_{RoomRserv}$ 这两个服务交互，并且这些交互的授权仅当 $S_{TScheduler}$ 服务于此 CyberTrip 时才有效. 另外需定义 $S_{AirBook}$ 同 $S_{RoomRserv}$ 间没有交互的可能性. CyberTA 与任意一家商业实体间的信任关系或协议的失效将使所有涉及该实体所提供服务的交互被禁止. 如果某个客户指定的酒店(HotelAtCityA)仅接受本地代理公司预订，则 $S_{TScheduler}$ 的房间预订请求会被拒绝，CyberTA 需另外选择与 HotelAtCityA 位于同一城市的 AgentAtCityA 公司代理这项业务. 在下文中，我们将详细介绍 SCoAC 这一面向服务计算的访问控制模型，说明上述的安全策略如何在该模型下得到表达并获实施.

3 SCoAC 访问控制模型

3.1 实体关系模型

建立一个面向服务计算的访问控制模型首先应描述该计算模式中各实体及其间的相互关系. 从系统安全的角度，我们在模型中考虑安全管理域间组织结构以及应用系统与成员服务间的关系. 其中涉及的实体包括：

(1) Service. 它是基本服务实体；

(2) Application. 它是指面向服务计算开发的应用系统. Application 定义了抽象的应用逻辑或业务流程，通过集成绑定具体服务实体为其提供所需功能，实现完整的应用功能；

(3) VirtualEnterprise. 它是指一个 Application 所处的安全管理域. 一个跨管理边界的商业应用系统所属的利益实体一般被称为一个“虚拟企业”，我们在此沿用这一术语；

(4) Organization. 它用于标识各个服务实体所处的安全管理域，即服务所属的利益实体. 对一个虚拟企业来说，这些实体为其“合作伙伴”；

(5) Port(端口). 它定义了应用系统对外界功能需求(此处 Port 不同于 WSDL 标准中 Port 的定义). 一个应用系统通过多个 Port 组合不同的服务为其应用需求服务；

(6) Role. 在 RBAC 及其诸多扩展模型中，角色(Role)均被定义为用户在某一组织机构内被赋予的“职务”(job title)[10]，在该机构内定义的职务被直接联系于一个权限的集合. 用户在被授予角色的同时也即被赋予了该角色对应的权限. 在 SCoAC 模型中，Role 具有不同的语意：Role 并不直接表达对服务实体的授权，而仅是服务能否被用于构建某个应用系统的限定条件.

其间的关系可以通过图 1 来说明.

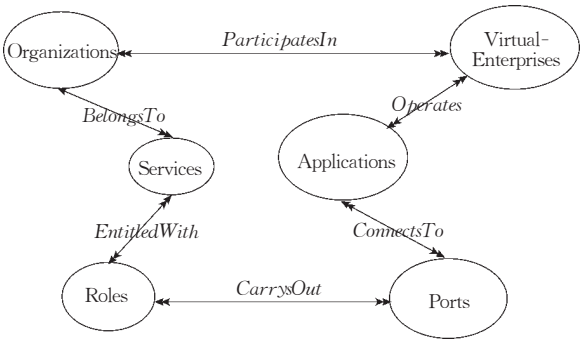

图 1 SCoAC 实体关系模型

我们用 $SERVICES$，$ORGS$，$VENTPS$，$APPS$，$PORTS$，$ROLES$ 分别表示环境中所存在的所有 Service，Organization，VirtualEnterprise，Application，Port 和 Role 等实体的集合. 对于图 1 中的各个关系的定义如下：

(1) $BelongsTo \subseteq SERVICES \times ORGS$.

它定义了服务实体与管理域之间的关系，指明某一实体所属的管理域；

(2) $ParticipatesIn \subseteq ORGS \times VEPNTS$.

它定义各服务管理域与应用系统所属管理域间的信任关系，即某一虚拟企业由哪些合作伙伴所组成；

(3) $Operates \subseteq VEPNTS \times APPS$.

它定义了应用系统所属利益实体，即某个应用系统由哪个实体部署并受其管理；

(4) $EntitledWith \subseteq SERVICES \times ROLES$.

它定义了服务所具有的角色，这些角色在当服务属性满足角色所表达的条件时被授予该服务实体；

(5) $CarrysOut \subseteq ROLES \times PORTS$.

它定义通过某个 Port 可被组合的服务所需具备的角色；

(6) $ConnectsTo \subseteq PORTS \times APPS$.

它定义应用系统所对外开放的 Ports，即应用系统对外的功能需求.

这些关系在多个层面受到出于不同目的的关注：$BelongsTo$ 关系随域管理员在域内部署或撤消个别服务实体而改变；$ParticipatesIn$ 通过在域间建立信任关系或制订合约来确定，随信任关系变更或合约执行、失效而变化；应用系统管理域部署或撤消整个应用系统(开始或停止某个业务过程)改

变 $Operators$ 关系；$ConnectsTo$ 关系在应用系统演化过程中对外功能需求发生改变时变更；$CarrysOut$ 的改变可反映在系统对提供外部功能的服务实体的资格认定标准的变化；$EntitledWith$ 关系的变更可调整对实体对系统提供服务的资质认定，服务也可通过激活或去活自己的角色以决定自己是否服务于某个应用系统来改变这一关系. 因此，这些关系状态受到环境中各种变化的影响. 我们使用 $BelongsTo^t$，$ParticipatesIn^t$，$Operates^t$，$EntitledWith^t$，$CarrysOut^t$ 和 $ConnectsTo^t$ 表示在当前环境内的有效关系.

3.2　基本授权模型

系统的有效状态空间 $StatusSpace^t$ 是贯彻于上述的诸个实体及其间有效关系之上的一个多元关系：

$$StatusSpace^t \subseteq SERVICES \times ORGS \times VENTPS \times ROLES \times PORTS \times APPS,$$

$$\begin{aligned} d = \langle d_s, d_o, d_{ve}, d_r, d_p, d_a \rangle \in StatusSpace^t \Leftrightarrow \\ & BelongsTo^t(d.d_s, d.d_o) \wedge \\ & ParticipatesIn^t(d.d_o, d.d_{ve}) \wedge \\ & Operates^t(d.d_{ve}, d.d_a) \wedge \\ & EntitledWith^t(d.d_s, d.d_r) \wedge \\ & CarrysOut^t(d.d_r, d.d_p) \wedge \\ & ConnectsTo^t(d.d_p, d.d_a). \end{aligned}$$

函数 $f_{Participants}: VENTPS \times APPS \rightarrow 2^{SERVICES}$ 定义为

$$f_{Participants}(ve, a) = \{s \mid d.ds = s, d.dve = ve, d.da = a, \text{ for } d \in StatusSpace^t\},$$

其中 $2^{SERVICES}$ 是 $SERVICES$ 的幂集.

$StatusSpace^t$ 定义了环境内所有应用系统与参与服务间的关系；$f_{Participants}(ve, a)$ 计算结果为所有服务于 ve 管理下的应用系统 a 的服务实体集合.

两个服务间的协作关系 $Cooperates$ 表示两者参与了同一应用系统：

$$\forall s_1, \forall s_2, s_1 \neq s_2, Cooperates(s_1, s_2) \Leftrightarrow \exists ve, \exists a, s_1 \in f_{Participants}(ve, a) \wedge s_2 \in f_{Participants}(ve, a).$$

定理 1.　对任意 s_1, s_2，若 $Cooperates(s_1, s_2)$ 成立，则 s_1, s_2 所属管理域 o_1, o_2 共同参与了某一虚拟企业 ve，并且 s_1, s_2 同服务于该 ve 管理的某一应用系统 a.

定理证明相对简单，此处不赘. 这一定理说明如果两个服务存在协作关系，则它们各自所属的安全管理域同时参与了某个虚拟企业（建立了信任关系），并且两者处于相同的应用上下文.

基本授权模型是允许存在协作关系的服务间进行交互. 一个服务 s 的授权集合（即它可与之交互的服务集合）$Auth(s)$ 定义为

$$Auth(s) = \{s' \mid s' \in SERVICES, Cooperates(s, s')\}.$$

3.3　绑定上下文

在 SCoAC 模型中，实体间交互被看作是双方平等地服务于应用系统，它们各自所具的属性应同时被作为访问决策的依据. 我们引入“绑定上下文”(BindingContext) 概念描述服务在应用系统内的参与状态，包括其参与系统所通过的 Port 以及连接到该 Port 上时所具有的 Role：

$$BindingContext \subseteq PORTS \times ROLES.$$

对交互双方属性考察的安全策略表达为一个绑定上下文匹配关系集合 $BCMatch$，定义相容的 $BindingContext$ 对：

$$BCMatch \subseteq APPS \times (BindingContext)^2.$$

$BCMatch^t$ 表示当前被激活的 $BindingContext$ 匹配对. 这一关系支持了安全策略中的两类策略的描述：(1) 定义连接于某些 Port 的实体间不存在交互的可能；(2) 存在交互可能性的双方的交互将受到某些细粒度的约束.

函数 $f_{MatchBC}: APPS \times PORTS \times ROLES \rightarrow 2^{ROLES \times PORTS}$ 根据 $BCMatch^t$ 计算与某个 $BindingContext$ 相容的 $BindingContext$ 集合.

$$\begin{aligned} f_{MatchBC}(a, (p, r)) = \\ & \{(p', r') \mid BCMatch^t(a, (p, r), (p', r')) \vee \\ & BCMatch^t(a, (p', r'), (p, r))\}. \end{aligned}$$

一个服务在某一应用系统的绑定上下文可通过函数 $f_{BindingContext}$ 计算. $f_{BindingContext}: SERVICES \times VEPNTS \times APPS \rightarrow 2^{PORTS \times ROLES}$ 定义为

$$f_{BindingContext}(s, ve, a) = \{(p, r) \mid (s, o, ve, r, p, a) \in StatusSpace^t, o \in ORGS\}.$$

s_1 和 s_2 可交互的必要条件为 $\exists (ve, a) \in CarrysOut$，使得

$$f_{BindingContext}(s_1, ve, a) \cap f_{BindingContext}(s_2, ve, a) \neq \varnothing.$$

如果对任意 $r_1, r_2 \in ROLES$，

$$\{(p_2, r_2)\} \cap f_{MatchBC}(a, (p_1, r_1)) = \varnothing, \quad a \in APPS, \ p_1, p_2 \in PORTS$$

表示 $BCMatch$ 定义在应用系统 a 中 p_1, p_2 两个 Port 所连接的服务之间不存在交互.

3.4　SCoAC 的授权算法

在基本授权的基础上，我们将绑定上下文匹配关系引入授权模型中. 在某一个应用系统管理域 ve 内，服务 s 可交互的授权集合 $AuthSet(s, ve)$ 可通过以下算法计算获得.

```
AuthSet(s,ve) ← ∅
for each a that (ve,a) ∈ Operates
  serviceBCSet ← f_BindingContext(s,ve,a)
  peers ← f_Participants(ve,a) − {s}
  for each peer ∈ peers
    peerBCSet ← f_BindingContext(peer,ve,a)
    for each pbc ∈ peerBCSet
      for each sbc ∈ serviceBCSet
        if pbc ∈ f_MatchBC(a,sbc) then
          AuthSet(s,ve) ← AuthSet(s,ve) ∪ {peer}
          break
        endif
      endfor
    endfor
  endfor
endfor
```

服务 s 的总授权集合为

$$Auth(s)=\bigcup_{ve\in VENTPS} AuthSet(s,ve).$$

服务授权状态的一致性表达为对服务实体 s_1 和 s_2，$s_1\in Auth(s_2)\Leftrightarrow s_2\in Auth(s_1)$.

这一性质确保对交互双方实施的访问控制不会出现矛盾. 证明如下：

根据算法，当 $s_1\in Auth(s_2)$ 时，必然 $\exists ve\in VENTPS, s_1\in AuthSet(s_2,ve)$. 继而，$\exists a\in APPS$, $(ve,a)\in Operates$ 使得算法可找到两个绑定上下文 bc_1 和 bc_2 分别满足条件 $bc_1\in f_{BindingContext}(s_1,ve,a)$ 和 $bc_2\in f_{BindingContext}(s_2,ve,a)$，并且 $(a,bc_1,bc_2)\in BCMatch^t$ 或 $(a,bc_2,bc_1)\in BCMatch^t$. 由于 $f_{BindingContext}(s_2,ve,a)$ 非空，根据 $f_{BindingContext}$ 定义，必然存在某 $p\in PORTS, o\in ORGS$ 及 $r\in ROLES$ 使 $(s_2,o,ve,r,p,a)\in StatusSpace^t$ 成立. 从 $f_{Participants}$ 定义可得 $s_2\in f_{Participants}(ve,a)$. 至此，再运行算法计算 $AuthSet(s_1,ve)$ 可得 $s_2\in AuthSet(s_1,ve)$ 及 $s_2\in Auth(s_1)$. 证毕.

4 实 例

第 2 节中所提的应用实例 CyberTrip 的部署和运行可用图 2 表示.

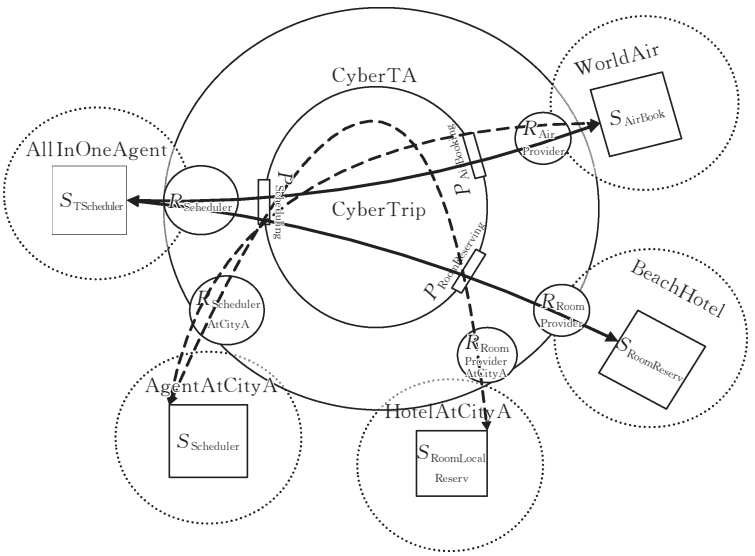

图 2 CyberTrip 应用示意

环境中各实体及关系如下：

$SERVICES=\{S_{TScheduler}, S_{AirBook}, S_{RoomReserv}, S_{RoomLocalReserv}\}$,

$ORGS=\{$CyberTA, AllInOneAgent, WorldAir, BeachHotel, AgentAtCityA, HotelAtCityA$\}$,

$VENTPS=\{$CyberTA$\}$,

$APPS=\{$CyberTrip$\}$,

$Operates=\{$(CyberTA, CyberTrip)$\}$,

$PORTS=\{P_{Scheduling}, P_{AirBooking}, P_{RoomReserving}\}$,

$ROLES=\{R_{Scheduler}, R_{SchedulerAtCityA}, R_{AirProvider}, R_{RoomProvider}, R_{RoomProviderAtCityA}\}$,

$BelongsTo=\{(S_{TScheduler}$, AllInOneAgent$)$, $(S_{AirBook}$, WorldAir$)$, $(S_{RoomReserv}$, BeachHotel$)$, $(S_{RoomLocalReserv}$, HotelAtCityA$)$, $(S_{Scheduler}$, AgentAtCityA$)\}$,

$ParticipatesIn=\{$(AllInOneAgent, CyberTA), (WorldAir, CyberTA), (BeachHotel, CyberTA), (AgentAtCityA, CyberTA), (HotelAtCityA, CyberTA)$\}$,

$ConnectsTo=\{(P_{Scheduling}$, CyberTrip$)$, $(P_{AirBooking}$, CyberTrip$)$, $(P_{RoomReserving}$, CyberTrip$)\}$,

$CarrysOut=\{(R_{Scheduler}, P_{Scheduling})$, $(R_{SchedulerAtCityA}, P_{Scheduling})$, $(R_{AirProvider}, P_{AirBooking})$, $(R_{RoomProvider}, P_{RoomReserving})$, $(R_{RoomProviderAtCityA}, P_{RoomReserving})\}$,

$EntitledWith=\{(S_{TScheduler}, R_{Scheduler})$, $(S_{Scheduler}, R_{Scheduler})$,

$(S_{\text{Scheduler}}, R_{\text{SchedulerAtCityA}}), (S_{\text{AirBook}}, R_{\text{AirProvider}}),$
$(S_{\text{RoomReserv}}, R_{\text{RoomProvider}}), (S_{\text{RoomLocalReserv}},$
$R_{\text{RoomProviderAtCityA}})\},$

$BCMatch = \{(\text{CyberTrip}, (P_{\text{Scheduling}}, R_{\text{Scheduler}}), (P_{\text{AirBooking}},$
$R_{\text{AirProvider}})), (\text{CyberTrip}, (P_{\text{Scheduling}}, R_{\text{Scheduler}}),$
$P_{\text{RoomReserving}}, R_{\text{RoomProvider}})), (\text{CyberTrip},$
$(P_{\text{Scheduling}}, R_{\text{SchedulerAtCityA}}), (P_{\text{RoomReserving}},$
$R_{\text{RoomProviderAtCityA}}))\}.$

(a) 假设应用系统根据客户旅行需求，将同 AllInOneAgent，WorldAir 及 BeachHotel 建立合作伙伴关系，并绑定服务 $S_{\text{TScheduler}}$，S_{AirBook}，$S_{\text{RoomReserv}}$ 到相应 Port. 此时除 ParticipatesIn 外其它关系均保持初始状态.

$ParticipatesIn^t = \{(\text{AllInOneAgent}, \text{CyberTA}),$
$(\text{WorldAir}, \text{CyberTA}),$
$(\text{BeachHotel}, \text{CyberTA})\}.$

运行授权算法得到

$AuthSet(S_{\text{TScheduler}}) = \{S_{\text{AirBook}}, S_{\text{RoomReserv}}\},$
$AuthSet(S_{\text{AirBook}}) = \{S_{\text{TScheduler}}\},$
$AuthSet(S_{\text{TScheduler}}) = \{S_{\text{TScheduler}}\}.$

根据 EntitledWith 关系的定义，AgentAtCityA 的 $S_{\text{Scheduler}}$ 和 HotelAtCityA 的 $S_{\text{RoomLocalReserv}}$ 同样具有服务于 CyberTrip 的角色，但由于（AgentAtCityA，CyberTA）和（HotelAtCityA，CyberTA）均未在 $ParticipatesIn^t$ 中被激活，所以系统 $StatusSpace^t$ 中不存在使得这两个服务连结到 CyberTrip 应用的关系，这两个服务也就不能与其它服务发生交互.

(b) 当根据客户需求系统需绑定 HotelAtCityA 服务于 $P_{\text{RoomReserving}}$ 时，

$ParticipatesIn^t = \{(AllInOneAgent, CyberTA),$
$(WorldAir, CyberTA),$
$(HotelAtCityA, CyberTA)\},$

此时 $S_{\text{TScheduler}}$ 与 $S_{\text{RoomReserv}}$ 间的交互权限被撤消. 并且由于 $BCMatch^t$ 不包含 $(CyberTrip, (P_{\text{Scheduling}}, R_{\text{Scheduler}}), (P_{\text{RoomReserving}}, R_{\text{RoomProviderAtCityA}}))$，所以原有的 AllInOneAgent 的 $S_{\text{TScheduler}}$ 服务不能再用以完成应用流程.

(c) 当系统重新绑定 AgentAtCityA 的 $S_{\text{Scheduler}}$ 服务后

$ParticipatesIn^t = \{(\text{WorldAir}, \text{CyberTA}),$
$(\text{AgentAtCityA}, \text{CyberTA}),$
$(\text{HotelAtCityA}, \text{CyberTA})\}.$

由于$(\text{CyberTrip}, (P_{\text{Scheduling}}, R_{\text{SchedulerAtCityA}}), (P_{\text{RoomReserving}}, R_{\text{RoomProviderAtCityA}}))$ 存在于 $BCMatch^t$ 中，$S_{\text{Scheduler}}$ 通过可分别激活被授予的 $R_{\text{Scheduler}}$ 和 $R_{\text{SchedulerAtCityA}}$，可先后完成在 WorldAir 的机票预定及 HotelAtCityA 的房间预定业务.

从这一例子可以看出，通过对环境中各实体及其间关系的描述，访问控制模型可直接表达并实施安全策略. 并且，当由于系统状态发生变化而导致安全策略随之更该时，可通过局部变更与系统状态对应的实体关系，使访问控制快速适应这些变化，实现系统运行状态与安全机制实施间的同步关系.

5　今后工作

对于模型中某些实体，我们可以进一步考察其应有特性，增强模型的表达能力. 例如，可在 SCoAC 中角色概念上定义类似 RBAC 中的角色层次结构，通过角色树描述角色之间的关系. 同样的，Port 之间也可引入层次关系. 基于这两者上的层次关系，我们可进而描述 BindingContext 间的关系. 这一工作不但可以简化管理，更为系统建立了一个抽象视图，增强其可理解性.

SCoAC 模型中对应用系统各成员的管理域间的关系只做了最简单的刻画，我们可以扩充、细化现在的基本的参与关系，增强对成员域间关系的表达. 例如属于同一兴趣圈的成员间可具有更强的信任关系，而他们之间的服务实体的交互也将受到较少的约束. 一个应用系统在 SCoAC 中被描述为通过多个 Port 连接多个服务的一个抽象实体. 这一抽象描述可被具体化（例如基于 TupleSpace 协同各成员服务的应用系统，或实现为 IWIM 模型中的 Manager[11]），以此来细化成员服务与应用系统间关系，以增强模型的表达能力.

目前我们正将 SCoAC 访问控制模型实现并应用到已有的服务组合模型下，如 BPEL4WS① 和 WSCI②. BPEL4WS 所基于的“代理”方式的服务访问模式与 SCoAC 对服务组合模型的假设不完全一致，可通过两种方式将 SCoAC 应用于 BPEL4WS 这类业务过程描述语言所描述的应用系统：(1) 直接应用以实施成员服务和组合服务间交互的访问控制；(2) 可通过信息流向（内部变量）将服务间的间接调用在逻辑上加以直接化，即当某个服务为流程“生产”的某个变量被另外一个服务所“消费”时，我们认为这两个服务发生了逻辑上的交互，这一交互能否发生根据 SCoAC 访问控制模型来决策. WSCI 本身虽是一个

① Business Process Execution Language for Web Services. http://www.ibm.com/developerworks/library/ws-bpel/

② Web Service Choreography Interface (WSCI) 1.0. http://www.w3.org/TR/wsci/

服务动态行为描述语言而非服务组合描述语言，但其中的 Global Model 提供了一个应用系统的全局视图，从这一视图我们可直接抽取出 SCoAC 访问控制中的多个关系，并通过 SCoAC 实施访问控制.

6 结 语

本文介绍了一个面向服务计算的访问控制模型 SCoAC. 服务实体间交互被看作是双方平等地服务于应用系统，通过建立服务计算环境中各实体间的关系模型，表达出服务间的信任和授权关系，并籍此对其间交互实施访问控制，此外还通过绑定上下文机制实现了细粒度访问控制. 该模型符合实际的安全管理模式，通过各实体间关系的定义，安全机制能有效适应面向服务的应用系统的动态性. 这样一个访问控制模型能更好地保护各服务实体，保障面向服务应用系统的正确运行.

参 考 文 献

1 Papazoglou M. P.. Service-oriented computing: Concepts, characteristics and directions. In: Proceedings of the 4th International Conference on Web Information Systems Engineering, 2003

2 Papazoglou M. P., Georgakopoulos D.. Service oriented computing: Introduction. Communications of the ACM, 2003, 46(10), 24～28

3 Camarinha-Matos L. M., Afsarmanesh H.. The virtual enterprise concept. In: Proceedings of the IFIP TC5 WG5.3/PRODNET Working Conference on Infrastructures for Virtual Enterprises, 1999

4 Coetzee M., Eloff J. H. P.. Virtual enterprise access control requirements. In: Proceedings of the 2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology, South African Institute for Computer Scientists and Information Technologists, 2003, 285～294

5 Hartman B., Flinn D. J., Kawamoto S.. Mastering Web Services Security. John Wiley & Sons, Inc., 2002

6 Periorellis P., Parastatidis S.. Task-based access control for virtual organizations. Lecture Notes in Computer Science 3409, 38～47

7 Thomas R. K., Sandhu R. S.. Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented autorization management. In: Proceedings of the IFIP TC11 WG11.3 Eleventh International Conference on Database Securty XI: Status and Prospects, London, UK, 1997, 166～181

8 Thomas R. K.. Team-based access control (TMAC): A primitive for applying role-based access controls in collaborative environments. In: Proceedings of the 2nd ACM Workshop on Role-based Access Control, Fairfax, Virginia, United States, 1997

9 Cohen E., Thomas R. K., Winsborough W., Shands D.. Models for coalition-based access control (CBAC). In: Proceedings of the 7th ACM Symposium on Access Control Models and Technologies, Monterey, California, USA, 2002

10 Sandhu R. S., Coyne E. J., Feinstein H. L., Youman C. E.. Role-based access control models. Computer, 1996, 29(2): 38～47

11 Arbab F.. The IWIM model for coordination of concurrent activities. In: Ciancarini P., Hankin C. eds.. In: Proceedings of the 1st International Conference on Coordination Models and Languages, Springer-Verlag, Berlin, Cesena, Italy, 1996, 34～56

CAO Chun, born in 1978, Ph. D. candidate. His research interests include access control technology, service oriented computing and software agents.

MA Xiao-Xing, born in 1975, Ph. D., associate professor. His current research interests include Internet software technology and software architectures.

LU Jian, borin in 1960, Ph. D., professor, Ph. D. supervisor. His research interests include software automation, formal methods for parallel programming and object-oriented languages and environments.

Background

During the research of Artemis-ARC, a supporting platform for service-oriented Internet applications, new challenges are perceived in securing the services and resources being shared. The dynamism of the application system facilitated by the loose-coupling between the constitutions requires the access control model to be expressive and flexible while not imposing heavy workload in administration. This paper introduces the SCoAC model aiming at scenarios in SOC, where access decisions derive from the inter-relationship of participants as well as the compatibility of their binding statuses against the application system. This work is supported by the National Natural Science Foundation of China under grant Nos. 60233010, 60403014, the National High Technology Research and Development Program (863 Program) of China under grant Nos. 2005AA113160, 2005AA113030, 2005AA119010 and the National Basic Research Program (973 Program) of China under grant No. 2002CB312002.