基于任务的强制访问控制模型

许 访[1], 沈昌祥[2]
(1. 海军工程大学, 湖北 武汉 430030; 2. 海军计算技术研究所, 北京 100841)

摘 要: 提出了一种基于任务的强制访问控制模型, 详细介绍了模型的基本组成, 用户模型规则和特权模型规则, 并针对其中存在的隐蔽通道, 在原有规则的基础上加入信息流控制策略。

关键词: 任务; 访问控制; 模型; 授权
中图法分类号: TP302.1 **文献标识码**: A **文章编号**: 1001-3695(2004)11-0070-02

Task-based Mandatory Access Control Model

XU Fang[1], SHEN Chang-xiang[2]
(1. *Naval Engineering University, Wuhan Hubei* 430030, *China*; 2. *Computer Technology Institute of Navy, Beijing* 100841, *China*)

Abstract: The paper proposes a task-based mandatory access control model. It introduces the components of the model, the rules of general model and privileged model. To resolve the covered channels, the police of information flow control is added into the model.

Key words: Task; Access Control; Model; Authorization

传统访问控制从主/客体角度出发, 考虑访问权的授予, 简单易行, 但存在以下弊端: 主体拥有某客体的访问权, 可在任何时刻执行任意次访问, 与主体当前执行的活动和任务无关。采用这种策略, 虽然可以防止非法用户对系统资源的访问, 却无法控制合法用户对访问权的正确使用。针对这一问题, 本文提出了一种基于任务的强制访问控制模型, 核心思想是: 主体访问某客体, 仅当访问满足强制访问控制策略, 并且该访问是主体执行当前授权任务所必需的。该模型可对客体实施细粒度的强制访问控制, 确保用户合理使用访问权, 防止隐蔽信道, 提高系统的安全性。

1 模型基本组成

1.1 模型基本元素

该模型主要由主体、客体、任务、安全级和访问权五要素组成, 各要素定义如下。

定义 1 主体及主体类

主体是系统中的活动实体, 如进程等。若当前主体集用 S 表示, 则 $S = \{S_1, S_2, \cdots\}$。依据主体的业务范围, 每个主体可属于一个或多个主体类, 不同主体类集合用 S_Class 表示, 则 $S_Class = \{S_Class_1, S_Class_2, \cdots\}$。$Class(S_i)$ 表示主体 S_i 所属的主体类集。

定义 2 客体及客体类

客体是系统中的被动实体, 如文件、记录等。若当前客体集用 O 表示, 则 $O = \{O_1, O_2, \cdots\}$。客体可依照它所服务的对象划分成类, 不同客体类集合用 O_Class 表示, 则 $O_Class = \{O_Class_1, O_Class_2, \cdots\}$。$Class(O_j)$ 表示客体 O_j 所属的客体类集。

定义 3 任务及任务类

每个系统应用是一系列任务的集合, 任务由若干操作组成, T 表示任务集, 则 $T = \{T_1, T_2, \cdots\}$。任务依照其所应用的服务对象, 可划分成类, 不同任务类集合用 T_Class 表示, 则 $T_Class = \{T_Class_1, T_Class_2, \cdots\}$。$Class(T_k)$ 表示任务 T_k 所属的任务类集。

主体当前正在执行的任务称为当前任务, 用 CT 表示, 则 $CT(S_i)$ 表示主体 S_i 的当前任务。

主体授权执行的任务称为授权任务, 用 AT 表示, 则 $AT(S_i)$ 表示主体 S_i 的授权任务集。存在主体 S_i, 其授权任务集 $AT(S_i)$ 为空。

每个主体在不同应用中有不同授权任务集, 每个主体在活动期有且仅有一个当前任务集, 且 $CT(S_i) \in AT(S_i)$。

主体类、客体类及任务类均对应同一个类型集 C, 三者间存在下述关系: S_Class = O_Class = T_Class = C

定义 4 安全级

系统中的每个主体或客体均属于一个安全级, 它表示主体或客体安全等级的高低, 安全级集用 Level 表示, $Level = \{Level_1, Level_2, \cdots\}$。$Level(S_i)$ 表示主体 S_i 的安全级, $Level(O_j)$ 表示主体 O_j 的安全级。

定义 5 访问权

主体对客体所执行的操作, A 表示访问权集, A = {read, write, delete, create}。若 $x \in A$, 则 $S_i \times O_j$ 表示主体 S_i 对客体 O_j 实施操作 x。

完成某项任务所必需的客体操作权称为必须访问权, 用 NA 表示, NA 为四元组 $(T_k, O_Class_j, Level_j, x)$, 表示执行任务 T_k 时, 必须对客体类 O_Class_j 中安全级为 $Level_j$ 的客体执行访问权 x。

当前状态, 主体对某客体的访问权称为当前访问权集 CA, CA 是由四元组 (S_i, T_k, O_j, x) 组成的集合, 它表示当前主体 S_i

收稿日期: 2003-06-28; 修返日期: 2004-03-15

执行任务 T_k 时, 可对客体 O_j 执行的所有操作集合。

1.2　模型约束

下述安全约束说明, 当系统处于某状态时, 必须满足的条件。由于添加和删除客体会引起系统状态的变化, 为确保系统安全, 两个连续状态之间也必须满足一定的条件。

约束 1: 主体当前执行的任务已授权给该主体。

若 $\forall S_i \in S$, 则 $CT(S_i) \in AT(S_i)$

约束 2: 主体当前执行某任务, 主体类包含当前任务类。

若 $\forall S_i \in S, O_j \in O, T_k \in T, CT(S_i) = T_k$,

则 $Class(T_k) \subseteq Class(S_i)$

约束 3: 主体创建客体, 仅当创建客体对当前任务是必须的。

若 $(CT(S_i), O_Class_j, Level_j, create) \notin NA, O_j \notin O$,

则 $O_j \notin O'$

约束 4: 主体删除客体, 仅当删除客体对当前任务是必须的。

若 $(CT(S_i), Class(O_j), Level(O_j), delete) \notin NA, O_j \in O$,

则 $O_j \in O'$

2　系统模型

在该系统中, 除使用系统资源的各用户外, 还有负责管理系统资源安全, 划分用户权限的安全管理员。相应地, 系统模型也可划分为普通用户使用的用户模型, 以及安全管理员使用的特权模型。

2.1　用户模型规则

用户模型描述系统普通用户的操作活动, 主要包括读客体、写客体、创建主体、创建客体及删除客体, 具体定义如下:

(1) 读客体。主体 S_i 执行任务 T_k 时, 读客体 O_j。

If $CT(S_i) = T_k, (T_k, Class(O_j), Level(O_j), read) \in NA, (S_i, T_k, O_j, read) \in CA$

then S_i read O_j

(2) 写客体。主体 S_i 执行任务 T_k 时, 写客体 O_j。

If $CT(S_i) = T_k, (T_k, Class(O_j), Level(O_j), write) \in NA, (S_i, T_k, O_j, write) \in CA$

then S_i write O_j

(3) 创建主体。主体 S_i 创建新主体 S_{new}。

If $CT(S_i) = nil, S_{new} \notin S$,

then $S' = S \cup \{S_{new}\}, CT(S_{new}) = nil, AT(S_{new}) = AT(S_i)$,

$S_Class(S_{new}) = S_Class(S_i), Level(S_{new}) = Level(S_i)$

(4) 创建客体。主体 S_i 执行当前任务 T_k 时, 创建 O_Class_j 类安全级为 $Level_j$ 的客体 O_{new}。

If $CT(S_i) = T_k, (T_k, O_Class_j, Level_j, create) \in NA, O_{new} \notin O$,

$(S_i, T_k, O_{new}, create) \in CA$,

then $O' = O \cup \{O_{new}\}, Class(O_{new}) = O_Class_j, Level(O_{new}) = Level_j$

(5) 删除客体。主体 S_i 执行任务 T_k 时, 删除客体 O_j。

If $CT(S_i) = T_k, (T_k, Class(O_j), Level(O_j), delete) \in NA, O_j \in O$,

$(S_i, T_k, O_j, delete) \in CA$,

then $O' = O - \{O_j\}, CA' = CA - \{(S_i * T_k * O_j * A) \cap CA\}$

2.2　特权模型规则

系统安全管理员负责分配用户权限, 管理系统资源, 它可增加/取消用户的授权任务, 增加/取消用户的读/写权限, 以及创建/删除客体的权限。依照最小特权原则, 上述特权操作可由若干(两个或三个) 安全管理员协同完成。

(1) 增加授权任务。为主体 S_i 增加授权任务 T_k。

If $Class(T_k) \subseteq Class(S_i)$,

then $AT'(S_i) = AT(S_i) \cup \{T_k\}$

(2) 取消授权任务。取消主体 S_i 的授权任务 T_k。

If $CT(S_i) \neq T_k$,

then $AT'(S_i) = AT(S_i) - \{T_k\}$

(3) 授予读权限。授予主体 S_i 执行任务 T_k 时, 对客体 O_j 的读权限。

If $T_k \in AT(S_i), Class(O_j) \subseteq Class(S_i), Level(S_i) \geq Level(O_j)$,

$(T_k, Class(O_j), Level(O_j), read) \in NA$,

then $CA' = CA \cup \{(S_i, T_k, O_j, read)\}$

(4) 取消读权限。取消主体 S_i 执行任务 T_k 时, 对客体 O_j 的读权限。

If $CT(S_i) \neq T_k$,

then $NA' = NA - \{(T_k, Class(O_j), Level(O_j), read)\}, CA' = CA - \{(S_i, T_k, O_j, read)\}$

(5) 授予写权限。授予主体 S_i 执行任务 T_k 时, 对客体 O_j 的写权限。

If $T_k \in AT(S_i), Class(O_j) \subseteq Class(S_i), Level(S_i) \leq Level(O_j)$,

$(T_k, Class(O_j), Level(O_j), write) \in NA$,

then $CA' = CA \cup \{(S_i, T_k, O_j, write)\}$

(6) 取消写权限。取消主体 S_i 执行任务 T_k 时, 对客体 O_j 的写权限。

If $CT(S_i) \neq T_k$,

then $NA' = NA - \{(T_k, Class(O_j), Level(O_j), write)\}, CA' = CA - \{(S_i, T_k, O_j, write)\}$

(7) 创建客体权限。主体 S_i 创建客体 O_j, 仅当客体类属于主体类, 主体安全级不高于客体安全级, 且对当前任务 T_k 是必需的。

If $CT(S_i) = T_k, Class(O_j) \subseteq Class(S_i), Level(S_i) \leq Level(O_j)$,

$(T_k, Class(O_j), Level_j, create) \in NA, O_{new} \notin O$,

then $(S_i, T_k, O_{new}, create) \in CA$

(8) 删除客体权限。主体 S_i 删除客体 O_j, 仅当客体类属于主体类, 主体安全级不低于客体安全级, 且对当前任务 T_k 是必需的。

If $CT(S_i) = T_k, Class(O_j) \subseteq Class(S_i), Level(S_i) \geq Level(O_j)$,

$(T_k, Class(O_j), Level(O_j), delete) \in NA, O_j \in O$,

then $(S_i, T_k, O_j, delete) \in CA$

3　信息流控制

根据 2.1 节介绍的用户模型规则, 主体对客体实施授权访问, 当且仅当满足强制访问控制, 且该访问是执行当前任务必须的。采用该策略虽然为系统提供了更高强度的安全保护, 但在某些情况下, 系统仍有隐蔽通道存在。在图 1 示例中, 用户仍能通过合法手段获取非授权信息。

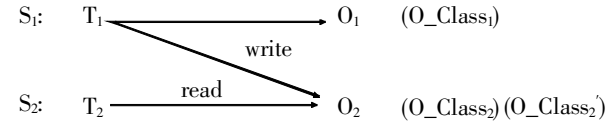

图 1　隐蔽通道示例

假设有主体 S_1, S_2, 它们的主体类不尽相同。在任务 T_1 中, O_1 属于客体类 O_Class_1, O_2 属于客体类 (下转第 74 页)

$$\frac{\partial\Psi}{\partial r}=(u_b-ug)\left(2r+\frac{R_c^3}{r^3}\right)\frac{\sin^2\theta}{2}$$
$$\frac{\partial\Psi}{\partial\theta}=(u_b-u_g)\left(r^2-\frac{R_c^3}{r}\right)\sin\theta\cos\theta \tag{15}$$

Rc 气泡晕的半径, *ug* 为相对运动速度。分别可以表示为:

$$R_c=\frac{D_b}{2}\left(\frac{u_b+2u_g}{u_b-u_g}\right)^{1/3}$$
$$u_g=\frac{u_{mf}}{\varepsilon_{mf}} \tag{16}$$

代入后可以得出径向和周向的速度:

$$u_r=-\left(u_b-\frac{u_{mf}}{\varepsilon_{mf}}\right)\left(1-\frac{R_c^3}{r^3}\right)\cos\theta$$
$$u_\theta=\left(u_b-\frac{u_{mf}}{\varepsilon_{mf}}\right)\left(2+\frac{R_c^3}{r^3}\right)\frac{\sin\theta}{2} \tag{17}$$

根据计算出的气泡晕中的颗粒的运动速度,对气泡晕内的颗粒运动进行虚拟的表示。就可以表示出气泡的形状和尾涡内的颗粒的卷入运动。为了对多个气泡的运动进行描述,分为以下几步实现:①先构建 bubble 类,能实现单个气泡的上升运动并破碎。②从 Cbubble 类继承并构建 CbubbleSystem 类,实现在某一个风帽上的一连串的气泡运动。③从 CbubbleSystem 类继承并构建 CbubbleSystemGroup 类,实现整个床层内的气泡运动。

气泡和尾涡内的颗粒运动效果如图 3 所示。

由于采用了面向对象的编程技术和 Intra3D 的框架结构,所以很容易实现各个对象的组合。我们可以将颗粒运动和气泡的运动相结合,即可以形成流化床炉膛内运动的虚拟表示。如图 4 所示。

图 2 颗粒运动

图 3 气泡运动

图 4 气泡和颗粒混合运动

4 结论及展望

通过流化床炉膛颗粒和气泡运动表示的研究可知:对于诸如流化床这样的复杂的热力设备,应用虚拟现实的表示方法,可以生动地再现热力过程。我们将继续进行流化床内各种气体浓度和温度场等虚拟表示的研究,最终建立一个完整的虚拟流化床,并以此来模拟流化床内的燃烧过程、粒子和气泡的运动,最终希望用虚拟流化床直观地对数值计算的结果进行表示,并能在一定程度上取代试验研究。

参考文献:

[1] ichael J Bockelie, et al. A Computational Work-bench Environment for Virtual Power Plant Simulation[EB/OL]. http://www.netl.doe.gov, 2000-10.

[2] 吴启迪. 系统仿真与虚拟现实[M]. 北京: 化学工业出版社, 2002. 1-9.

[3] 费广正, 乔林. Visual C++ 6.0 高级编程技术 OpenGL 篇[M]. 北京: 中国铁道出版社, 2000. 30-36.

[4] 张芹, 谢隽毅, 吴慧中, 等. 火焰、烟、云等不规则物体的建模方法研究综述[J]. 中国图像图形学报, 2002, 5(3): 186-190.

[5] 徐祥, 等. 流化床密相区流动特性的数值模拟[J]. 热能动力工程, 2004, (2).

[6] 章明耀. 增压流化床联合循环发电技术[M]. 南京: 东南大学出版社, 1998. 59-89.

[7] 岑可法, 等. 循环流化床锅炉理论设计与运动[M]. 北京: 中国电力出版社, 1998. 49-94

作者简介:

向文国(1964-), 男, 副教授, 硕士, 研究方向为电厂热力系统仿真; 徐祥(1979-), 男, 硕士, 研究方向为电厂热力系统仿真; 沈来宏(1965-), 男, 副教授, 博士, 研究方向为流化床研究; 肖军(1968-), 女, 副教授, 博士, 研究方向为能源系统优化及技术经济分析。

(上接第 71 页) O_Class_2; 在任务 T_2 中, O_2 属于客体类 O_Class_2', 且 $O_Class_2 \neq O_Class_2'$, $O_Class_1 \neq O_Class_2'$。$S_1$ 执行任务 T_1 时, 可对 O_1 执行读操作, 对 O_2 执行写操作, 则它把 O_1 中的敏感数据写入 O_2。接着, S_2 执行 T_2, 可从 O_2 中读取 O_1 内的敏感信息。

为遏制上述隐蔽通道的存在, 在系统模型中加入下述信息流约束:

信息流约束, 即在任何状态下, 主体 S_i 在同一任务中既对客体 O_1 有读权限, 又对客体 O_2 有写权限, 则 O_2 客体类集应包含于 O_1 客体类集。即: $Class(O_2) \subseteq Class(O_1)$

加入信息流约束后, 其他授权操作保持不变, 授予读权限改为下述形式:

授予读权限, 即授予主体 S_i 执行任务 T_k 时, 对客体 O_j 的读权限。

If $T_k \in AT(S_i)$, $Class(O_j) \subseteq Class(S_i)$, $Level(S_i) \geq Level(O_j)$,
$(T_k, Class(O_j), Level(O_j), read) \in NA$,
$(\forall O_k \in O, (S_i, T_k, O_k, write) \in CA, Class(O_k) \subseteq Class(O_j))$
then $CA' = CA \cup \{(S_i, T_k, O_j, read)\}$

4 结束语

本文详细介绍了基于任务的强制访问控制模型的基本组成, 系统模型及信息流控制策略, 该模型适用于各种计算环境及企业级应用。采用该模型, 不仅可以防止非法用户对系统资源的使用, 而且能有效控制合法用户访问权的滥用, 增强系统安全性。在今后的研究中, 还需与其他安全策略结合, 提高模型的易用性和使用效率。

参考文献:

[1] K Thomas, R S Sandhu. Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management[C]. Proceedings of the IFIP WG11.3 Workshop on Database Security, Lake Tahoe, California, 1997. 11-13.

[2] Fischer-Hubner. IT Security and Privacy[C]. LNCS 1958, Springer-Verlag Berlin Heidelberg, 2001.

[3] achong Park, Ravi Sandhu. Towards Usage Control Models: Beyond Traditional Access Control[C]. 7th ACM Symposium on Access Control Models and Technologies, 2002.

[4] imthy Fraser, Lee Badger. Ensuring Continuity During Dynamic Security Police Reconfiguration in DTE[C]. Proceedings of the 1998 IEEE Symposium on Security and Privacy, 1998.

作者简介:

许访(1976-), 男, 博士研究生, 主要研究方向为信息系统安全、操作系统安全; 沈昌祥, 男, 中国工程院院士, 博士生导师。