一种新的操作系统安全模型*

李益发**

(信息工程大学信息工程学院，郑州 450002；江南计算技术研究所博士后科研工作站，无锡 214000)

沈昌祥

(海军计算技术研究所，北京 100841)

摘要 机密性和完整性是操作系统安全的两个重要特性，BLP 模型只解决了机密性的问题，而 Biba 模型只解决了完整性的问题，其他模型也都类似，无法使二者同时兼顾．基于一个特殊可信主体，提出了一种新的操作系统安全模型，融合了以上两种模型，可以同时保持机密性与完整性．

关键词 操作系统 安全模型 **BLP** **Biba** 形式化证明

操作系统是计算机资源的直接管理者．它管理硬件设备并为用户提供接口，同时也是计算机软件的基础和核心．而计算机网络则由计算机互联而成．因此，网络中信息流的安全极大地依赖于各主机操作系统的安全．可以说，操作系统是整个计算机系统安全的基础，没有操作系统的安全，就谈不上建立于其上的数据库安全、和其他应用软件的安全，更谈不上整个网络的安全．因此，安全操作系统的研究一直是信息安全的一个重要内容．

操作系统的安全性体现在以下几个方面：可用性、机密性、完整性、可追溯性(也称非否认性或负责性)．可用性是指操作系统能有效地提供必要的服务，即能对计算机资源进行预期的有效管理．病毒发作，或者被远程非法控制，导致计算机的合法用户无法使用，侵害的就是计算机的可用性．机密性是指计算机中的信息只能被合法的用户所见，而不能被非授权的用户获得．完整性是指计算机中的信息不能被非法修改或破坏．可追溯性是指对信息的操作可以追查到操作者，使操作者无法否认，从而必须对此操作负责．

收稿日期: 2005-02-05；接受日期: 2005-10-21
* 江南计算技术研究所博士后科研基金和国家“九七三”项目基金(035801)资助
** E-mail: alphalyf@163.com

由于操作系统的复杂性，要保证一个实际的操作系统真正满足以上特性是极为困难的. 至少有以下两方面的问题需要研究. 一方面，以上特性能否同时满足? 或者说，如何从理论上保证操作系统能够实现以上特性? 另一方面，如何确认一个实际的操作系统已经满足以上特性?

回答这些问题是相当困难的，因此，人们首先希望能建立一个理论的安全操作系统模型，满足以上特性的要求. 这就导致了对操作系统安全模型的研究.

1 安全模型的发展

1.1 安全模型的概念

所谓安全模型，就是对安全策略所表达的安全需求的简单、抽象和无歧义的描述[1]. 它为安全策略与其实现机制的关联提供了一种框架. 安全模型描述了对某个安全策略需要用哪种机制来满足，而模型的实现则描述了如何把特定的机制应用于系统中，从而实现某一特别的安全策略所需要的安全保护.

1.2 安全模型的发展

最早的安全模型是由Bell和LaPadula提出的，称为BLP模型[2,3]. BLP模型主要解决对客体机密性的控制，防止信息由高安全级流向低安全级. 其基本思想是，给每个主体和客体赋予一个安全级(机密级)，当主体申请对客体进行操作时，安全机制通过安全级来判别是否允许其操作. 如果主体的安全级高于客体，则允许其读而不允许其写. 反过来，当主体的安全级低于客体的安全级时，允许其写但不允许其读. 这样就保证了机密信息不被泄露给安全级较低的主体. 但BLP模型缺少完整性控制，且不能有效地限制隐蔽通道. Biba模型[4]则注重保护信息的完整性，但缺少机密性控制. 长时间来，这两者都难以调和.

此外还有RBAC模型、信息流模型、无干扰模型[5]、DTE模型[6]、Clark-Wilson(CW)模型等. 这些模型各有千秋，如RBAC的结构简单，易于实现，也易与其他模型结合，但很难控制同一用户以不同的身份进入系统破坏安全性；信息流模型可以解决隐通道问题，但严格的安全条件有些是不可判定的；而CW模型可以在理论上较好地解决完整性问题，但不易在现实系统中实现. 所以，以上模型都不能圆满地解决操作系统的安全性问题，或者说不能满足操作系统的所有安全需求.

鉴于安全模型研究的困难性，甚至有人提出，根本就不存在真正的“安全模型”.

1.3 安全模型的存在性探讨

操作系统的功能就是要管理计算机资源，并为用户提供方便的界面. 计算机

资源可分为硬件、软件和数据. 但不论管理的对象是什么, 操作系统在本质上都是对信息流的管理.

作为管理者, 首先要明确的是, 任一管理目标的实现, 都必须通过操作系统的协助才能完成, 否则操作系统就不是真正意义上的所有计算机资源的管理者. 其次, 操作系统实现对资源的管理, 本质上就是实现信息资源在计算机管辖范围内的流动. 第三, 任一操作系统都是为人服务的, 它所做的任何操作都代表着用户(记为 u). 当一个人以用户的身份登录到操作系统中使用计算机时, 本质上, 计算机的一切行为都是代表用户进行的, 只不过, 安全的操作系统能恰如其分地实现用户的意图, 而不安全的操作系统会以用户的名义做违背用户意愿的操作. 操作系统作为计算机资源的管理者, 是通过各种操作过程实现其管理的. 安全模型的建立, 就是为了实现对操作过程的精确控制, 以确保每一个操作尽可能遵循用户的意图.

任一操作过程有以下三要素: 操作、操作者(操作的执行者)和操作对象. 通常, 我们称操作者为主体(记为 s). 操作对象可能是另一个主体. 如果操作对象不是主体, 则称该操作对象为客体(记为 o). 将所有主体的集合记为 S, 所有操作的集合记为 A, 所有客体的集合记为 O, 则操作过程是定义在 $S\times A\times S$ 或 $S\times A\times O$ 上的关系.

安全模型的目标是, 允许执行适合用户意图满足安全要求的操作, 而拒绝不适合用户意图不满足安全要求的操作. 为此, 引进一个二元集 $B=\{0, 1\}$, 则安全模型可以看成 $S\times A\times S\times B$ 和 $S\times A\times O\times B$ 上的关系.

以 $S\times A\times O\times B$ 为例($S\times A\times S\times B$ 完全类似), 任一元素$(s, x, o, 0)$表示在模型中 s 不能对 o 执行操作 x, 而$(s, x, o, 0)$则表示能进行操作. 换个角度看, 它是 $S\times A\times O$ 到 B 上的映射, 像为 1 的是允许操作集, 而像为 0 的则是不允许操作集. 既然是映射, 就必须有映射的规则, 而所谓安全模型, 本质上就是对这种映射规则的描述.

但是, 操作系统中存在这样的操作过程, 其操作结果是产生一个新的主体和客体, 从而导致 S 和 O 都是不确定的集合. 而在一个不确定的集合上定义映射, 从理论上说似乎是不成立的(给定一个映射的前提是, 先给定集合 X 和 Y, 再给出 X 到 Y 的对应规则). 这也就是建立一个好的安全模型之所以困难的原因所在.

然而, 这并不等于说, 在理论上不可能建立安全模型. 因为我们可以换个角度来重新审视这一问题. 尽管在总体上抽象地看待 S 和 O 时, 它们是不确定的, 但我们的出发点是确定的. 也就是说, 最初的 S 和 O 都是确定的, 事实上它们都是有限集. 这样, 我们就可以从有限的确定集 S 和 O 开始, 递归式地给出不断扩展的集合上的映射. 在这样的意义下, 安全模型是有理由存在的.

2 一种新的安全模型

2.1 若干基本概念

基本操作: 基本操作有以下几种. 只读(读出数据, 但不作其他操作, 记为 *r*)、添加(不读, 只在客体数据尾添加, 亦即只写, 记为 *a*)、读写(记为 *w*)、执行(记为 *e*). 在基本操作的基础上, 我们还定义一个特殊的操作——创建, 并用符号“ э ”表示. *sэo* 表示主体 *s* 创建一个新的客体 *o*, 同样也可以有 *sэs′*, *s′эS*, 如一个用户在执行一个应用程序时, 首先要创建一个进程 *s* 来代表它进行其后的操作, 这时就有 *uэs*.

记号: 用 *x*(*s*, *o*)表示主体 *s* 对客体 *o* 进行了 *x* 操作, 其中 *x* 为 *r*, *e*, *a* 或 *w*. 用符号“ε”表示“请求”, 用“∂”表示“可以”. 于是, ε (*r*(*s*, *o*))表示 *s* 请求对 *o* 进行读操作, 而∂(*r*(*s*, *o*))表示 *s* 可以对 *o* 进行读操作. 在不致引起混淆的情况下, 也简写为 εr(*s*, *o*)和∂r(*s*, *o*).

同时, 用 *U* 表示所有用户组成的集合, 它是主体集的一个子集. 虽然用户也是主体, 但不同于用户所创建的其他主体. 我们总是假定, 来自于鼠标和键盘的指令所形成的进程不同于其他应用程序所创建的进程. 操作系统不仅有必要而且也能够区分这一点. 当我们使用符号 *u* 时, 我们总是默认 $u \in U$.

特殊可信主体: 定义一个特殊可信主体π, 它代表机密性检查员和完整性检查员. 机密性检查员有权检查并改变客体的机密级, 而完整性检查员则有权检查并改变客体的完整级. 设置π是本文的一个特色所在. π 作为完整性检查员的工作模式如下图所示:

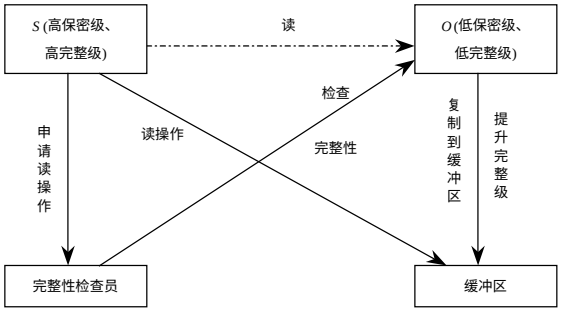


图 1

图 1, 对 *o* 的检查合格后, π可提升其完整级. 当然, π 作为机密性检查员和完整性检查员职责可分开, 由两个主体来担任. 但由于这两个主体都是可信的, 合而为一本质上是一样的. 其他操作与本图类似.

特殊主体π 所做的是一种特殊的操作. 表面上, π 的操作改变了客体的机密

级和完整级. 但实际上, 它改变了主体对特定客体的临时操作权限. 设 s 是一个主体, o 是一个客体. 假定按照 s 的机密级和完整级, s 不能对 o 进行某种操作, 但在π的帮助下, 通过改变 o 的级别, 使得 s 可以对 o 进行操作. 这相当于π对 s 与 o 产生了一个指定, 该指定使得不论 s 自身的机密级和完整级如何, 都可以对 o 进行指定的操作. 记该指定为$\lambda(\pi, x(s, o))$. 因为π是唯一的, 简记为$\lambda(x(s, o))$.

特殊用户: 定义一个特殊的用户θ, 它是公共客体的拥有者的代表. 此外, 由于主体π的特殊性, 我们将π也看成一个特殊用户, 即$\pi \in U$.

标记: 标记是一组参数, 用来表示一个主体或客体的各种属性. 每个标记或者按照一定的规则由操作系统自动生成, 或是系统在接纳新成员时由高级用户指定, 或是由用户自己定义. 任一主体和客体的标记只有满足一定的条件才能改变.

客体的标记: 每个客体都用以下几类参数进行标记: 所有者、机密级和完整级. 所有者是指该客体归属于哪个主体. 为方便起见, 我们用τ表示主体与客体之间的二元关系"拥有", $\tau(o) = s$ 就表示 s 是 o 的所有者. 机密级用 C 表示, 用于机密性保护; 完整级用 L 表示, 用于完整性保护. C 和 L 的取值都是非负整数, 通常取值为 0, 1, 2, 3. 其中, C 为 0, 1, 2, 3 时分别表示公开、秘密、机密和绝密; L 为 0, 1, 2, 3 时分别表示普通、重要、很重要和极重要.

于是一个客体 o 可标记为: $(\tau(o), C(o), L(o))$.

主体的标记: 为了对主体进行较好的安全管理, 每个主体用以下标记来进行区别: 用户、拥有者、组别、辖域及权限.

首先, 在主体之间也可以定义一个"拥有"关系, 即如果一个主体 s 创建了一个新的主体 s', 则称 s 拥有 s', 也称 s'隶属于 s, 记为$\tau(s') = s$. 需要说明的是, 这里τ仍然表示拥有关系, 所不同的是, 这里的τ是主体之间的二元关系. 由于$\tau(o)$与$\tau(s)$可以显然地区分开来, 因此不会混淆.

其次, 依据拥有关系可以自然地定义一个顺序关系"$\succ$": $s = \tau(s') \rightarrow s \succ s'$; 若 $s \succ s'$且 $s' \succ s''$, 则 $s \succ s''$. 一般地, 如果 $s \ni s'$, 则称 s 是 s'的父主体, 而 s'是 s 的子主体. 如果 $s \succ s'$, 则称 s 是 s'的前辈或祖先, 而称 s'是 s 的晚辈或后代. 在这样的定义下, 显然每个主体都位于某个主体链中. 主体链是一个有限集, 因此存在一个极大元——某个用户 u.

由顺序关系$\succ$又可以自然地定义一个偏序关系"$|\succ$": $s|\succ s' \leftrightarrow (s = s' \vee s \succ s')$. 这种偏序关系说明主体集是一个偏序集. 令 $G(s)=\{s': s|\succ s'\}$, 称为 s 的组别(group). 显然, 主体的组别形成一个范畴. 这实际上是一个家族范畴, 每个主体都以自身为起点, 由它和它的所有后代形成一个组, 并且该组自然地包含在其祖先所形成的组中. 操作系统创建的进程可以在创建过程中自然形成组. 而在用户层面上, 多

个用户的系统中，低级用户的“辈份”则是由高级用户定义的.

第三，由主体与客体之间的拥有关系可以定义一个集合 $D(s)=\{o: \tau(o)\in G(s)\}$，它是 s 的组别所能拥有的所有客体组成的集合，我们自然地把 $D(s)$看成主体所能操作的客体范围，因而称之为辖域(domain).

最后，每个主体分配有不同的权限. 权限有以下几种，即机密权限、完整权限、和指定权限. 机密权限又称为机密级或密级，记为 C，用于标定主体在机密性上所拥有的权限；完整权限又称为完整级，记为 L，用于标定主体在完整性上所拥有的权限. 指定权限是一个特殊的权限，只有特殊主体π才有“指定”权，它通过“指定”来实现特定主体对特定客体的指定操作.

于是，一个主体 s 可表示为以下的数组: $(\tau(s), G(s), D(s), C(s), L(s))$.

公式: 设Γ是集合, $z\in\Gamma$表示 z 是集合Γ的元素. 则公式归纳定义如下:

(1) $x(s, o)$是基本操作公式(x 是 r, e, a 或 w);

(2) $s\ni s'$, $s\ni o$, $\tau(s)=s'$, $\tau(o)=s$, $s\succ s'$, $\lambda(x(s, o))$, $z\in\Gamma$ 和基本操作公式是原子公式;

(3) 若φ是基本操作公式，则$\varepsilon(\varphi)$, $\partial(\varphi)$是原子公式;

(4) 原子公式是公式;

(5) 若φ是公式，则$\neg\varphi$, $\forall s(\varphi)$, $\exists s(\varphi)$, $\forall o(\varphi)$, $\exists o(\varphi)$是公式;

(6) 若φ和ψ是公式，则$\varphi\wedge\psi$, $\varphi\vee\psi$, $\varphi\rightarrow\psi$, $\varphi\leftrightarrow\psi$是公式;

在上述定义下，所有公式形成一个公式集，记为$\mathfrak{H}$. 以下的讨论都是在$\mathfrak{H}$中进行的. 这里$\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$分别表示非、与、或、蕴含和当且仅当. $\forall$和$\exists$分别表示任意和存在.

2.2 强制安全规则

以下若非特别说明，所有主体符号 s, s', s''等都不等于π.

本模型给定以下普通安全规则:

规则 1. 若 $s\ni o$，则 s 自然是 o 的拥有者，其逻辑公式为

$$(s\ni o\rightarrow\tau(o)=s).$$

规则 2. 关于π 和θ 的基本定义: $\forall s(\theta\in G(s)\wedge s\in G(\pi))$.

注 **1** θ 可以理解为所有公用客体的拥有者，具有最低的机密级和完整级(都为 0). 所以自然有$\forall s(\theta\in G(s))$. π 是特殊主体，不仅具有最高的机密级和完整级，还能对所有客体的权限进行操作，它是最高层的用户，所以规定$\forall s(s\in G(\pi))$.

规则 3. 子主体继承创建它的父主体的机密级和完整级.

$$s\ni s'\rightarrow(L(s')=L(s)\wedge C(s')=C(s)).$$

规则 4. 任一主体 s 创建客体 o 时, o 继承 s 的密级和完整级:

$$s\ni o\rightarrow(L(o)=L(s)\wedge C(o)=C(s)).$$

规则 5.　当且仅当客体的组别范畴包含于主体组别范畴, 同时密级大于等于客体密级且完整级小于等于客体完整级时, 或者存在一个相应的指定时, 主体可以对客体进行 r 和 e 操作.

$$((o\in D(s)\wedge C(s)\geq C(o)\wedge L(s)\leq L(o))\vee\lambda(r(s,o)))\leftrightarrow\partial r(s,o).$$

$$((o\in D(s)\wedge C(s)\geq C(o)\wedge L(s)\leq L(o))\vee\lambda(e(s,o)))\leftrightarrow\partial e(s,o).$$

规则 6.　当且仅当客体的组别范畴包含于主体组别范畴, 并且主体的密级小于客体的密级, 同时主体完整级大于等于客体完整级时, 或者存在一个相应的指定时, 主体可以对客体进行 a 操作.

$$((o\in D(s)\wedge C(s)\leq C(o)\wedge L(s)\geq L(o))\vee\lambda(a(s,o)))\leftrightarrow\partial a(s,o).$$

规则 7.　当且仅当客体的组别范畴包含于主体组别范畴, 同时主体的密级等于客体的密级, 并且完整级等于客体的完整级时, 或者存在一个相应的指定时, 可以对客体执行 w 操作.

$$((o\in D(s)\wedge C(s)=C(o)\wedge L(s)=L(o))\vee\lambda(w(s,o)))\leftrightarrow\partial w(s,o).$$

规则 8.　设φ是任何一个基本操作公式, 则

$$\varepsilon(\varphi)\wedge\partial(\varphi)\leftrightarrow\varphi.$$

$$\varepsilon(\varepsilon(\varphi))=\varepsilon(\varphi).$$

$$\partial(\partial(\varphi))=\partial(\varphi).$$

以上规则都是针对只读、执行、添加和读写这些基本操作, 而没有涉及高级用户对低级用户的授权等更高级的操作.

2.3　系统模型的可用性

通常, 保密性和完整性不能兼顾, 所以才有 BLP 和 Biba 模型的不同. 二者合二为一时, 由于主体在创建客体时, 客体的密级和完整级由主体的密级和完整级唯一确定, 从而会严重影响系统的可用性.

例如: 主体 s 和 s'分别创建客体 o 和 o'. 设 $C(s)=L(s)=3$, $C(s')=L(s')=2$, 则由规则可知, $C(o)=L(o)=3$, $C(o')=L(o')=2$. 于是, 受完整级的限制, s 不能对 s'创建的任何客体进行读操作; 同时, s'也不能对 s 创建的任何主体进行添加操作. 这就大大限制了操作系统的可用性.

但我们的模型中存在一个特殊主体π, 它在一定的条件下可以改变客体的机密级和完整级. 在上例中, 当 s 申请读 o'时, 征得π的同意, 可以有$\lambda(r(s,o'))$. 这一指定是通过π改变 o'的完整级实现的, 就是将 o'完整级指定为 3(记为 $\lambda(L(o')=3)$), 这样 s 就可以对 o'进行读操作. 同样, 当 s'申请向 o 进行添加操作时, 可以有$\lambda(L(o)=2)$, 这样 s'就可以对 o 进行添加操作了. 当然, π不会接受所有改变客体密级和完整级的申请, 它会根据一定的条件来判别何时接受何时拒绝.

例如: 向上添加可能会带来病毒, 但π可以先进行查毒, 确认无毒后, 才将 o

的完整级改变成 3, 此后 s 才能对 o 进行读操作.

所以表面上看, π控制的是客体的权限. 但实际上, π实现的是在一定条件对特定主体访问特定客体进行授权.

需要特别强调的是, π必须是可信主体, 否则系统将无法保证安全.

3 系统模型的安全性分析

3.1 几个相关的引理

引理 **1** $\forall o\exists s(\tau(o)=s)$. 即任一客体都属于某个主体.

证 在系统创建之初, 任一客体显然只属于两个主体: π和θ. 其后, 由于规则保证每产生一个新的客体, 都存在一个该客体的创建者, 因而每个客体都有其拥有者.

引理 **2** $\forall o\exists u(u\succ\tau(o))$.

证 由引理 1, o 存在一个创建者 s. 而由规则 2, $s\in G(\pi)$, 即$\pi\succ\tau(o)$. 这表明任一客体也总是归属于某个用户, 至少, 它属于π.

引理 **3** $(s\ni s'\leftrightarrow\tau(s')=s)\wedge(s|\succ s'\leftrightarrow s'\in G(s))$.

证 这里的两个充要条件是由符号的定义决定的. 该引理是说当一个主体(父主体)创建一个新主体(子主体)时, 创建者是被创建者的拥有者, 并且子主体自然地属于其前辈主体所在的组别.

作为这个引理的简单推论, 我们有:

引理 **4** $s'\in G(s)\rightarrow G(s')\subseteq G(s)$.

证 若 $s'\in G(s)$, 且 $s''\in G(s')$, 则由引理 3, $s|\succ s'$, 且 $s'|\succ s''$, 所以 $s|\succ s''$, 仍由引理 3 得, $s''\in G(s)$.

引理 **5** $(\tau(o)\in G(s)\leftrightarrow o\in D(s))\wedge(\tau(o)=s\rightarrow o\in D(s))$.

证 前面的充要条件是由 $D(s)$的定义决定的. 对后一式, 若$\tau(o)=s$, 则由 $G(s)$的定义可知, 必有$\tau(o)\in G(s)$, 从而有 $o\in D(s)$.

该引理说明, 当$\tau(o)=s$ 时, o 自然地属于 s 的辖域. 反过来, 如果 o 在 s 的辖域中, 则$\tau(o)$也必与 s 在同一组别中.

引理 **6** $G(s')\subseteq G(s)\leftrightarrow D(s')\subseteq D(s)$.

证 若 $G(s')\subseteq G(s)$, 设 $o\in D(s')$, 则由引理 4, $\tau(o)\in G(s')$, 从而$\tau(o)\in G(s)$, 再由引理 5 即得, $o\in D(s)$, 所以 $D(s')\subseteq D(s)$.

反过来, 设 $D(s')\subseteq D(s)$, $s''\in G(s')$, 则对任一 o 使得$\tau(o)=s''\in G(s')$, 有 $o\in D(s')\subseteq D(s)$. 即有 $o\in D(s)$. 由 $D(s)=\{o:\ \tau(o)\in G(s)\}$知, $s''=\tau(o)\in G(s)$. 所以 $G(s')\subseteq G(s)$.

引理 **7** $s|\succ s'\rightarrow D(s')\subseteq D(s)$.

证　由引理 3, 4, 6 即得. 这表明客体的拥有者的前辈仍是该客体的拥有者, 它反映了客体集作为主体的辖域也自然形成一个偏序集.

引理 **8**　$G(u)\cap G(u')=\{\theta\}\rightarrow\forall s\forall s'((s\in G(u)\wedge s'\in G(u'))\rightarrow G(s)\cap G(s')=\{\theta\})$.

证　显然有 $G(s)\cap G(s')\supseteq\{\theta\}$. 如果 $G(s)\cap G(s')\supset\{\theta\}$, 则存在 $s''\neq\theta$, 使得 $s''\in G(s)\cap G(s')$, 但 $G(s)\subseteq G(u)$, $G(s')\subseteq G(u')$, 所以有 $G(s)\cap G(s')\subseteq G(u)\cap G(u')$, 从而有 $s''\in G(u)\cap G(u')$. 这与 $G(u)\cap G(u')=\{\theta\}$矛盾.

该引理说明, 如果两个用户在不同的组中, 则它们的晚辈也必不在同一组中.

引理 **9**　$G(s)\cap G(s')=\{\theta\}\rightarrow D(s)\cap D(s')=D(\theta)$.

证　若 $o\in D(s)\cap D(s')$, 则由引理 5 有 $\tau(o)\in G(s)\cap G(s')=\{\theta\}$, 从而 $o\in D(\theta)$.

这表明不同组别的主体, 其辖域除公共集外不相交.

3.2　系统模型的安全性分析

由上述引理可以得到如下关于系统安全性的定理.

定理 **1**　设 s 是任一主体, o 是任一客体, 则

(1) $(\neg\lambda(r(s, o))\wedge C(s)<C(o))\rightarrow\neg r(s, o)$;

(2) $(\neg\lambda(e(s, o))\wedge C(s)<C(o))\rightarrow\neg e(s, o)$);

(3) $(\neg\lambda(a(s, o))\wedge C(s)>C(o))\rightarrow\neg a(s, o)$;

(4) $(\neg\lambda(w(s, o))\wedge C(s)\neq C(o))\rightarrow\neg w(s, o)$.

即任一主体对任一客体的操作在机密级上满足不能向上读(执行)和不能向下写(执行). 我们称之为机密性定理.

证　(1) 设$\neg\lambda(r(s, o))$、$C(s)<C(o)$和 $r(s, o)$同时成立. 则由规则 8 可得, $r(s, o)\rightarrow\partial r(s, o)$, 再由规则 5 可知, $\partial r(s, o)\rightarrow(\lambda(r(s, o))\vee(C(s)\geq C(o)\wedge L(s)\leq L(o)\wedge o\in D(s)))$, 从而有 $C(s)\geq C(o)$, 矛盾. 所以, 主体在机密级上不能向上读.

同理可对(2)~(4)进行证明.

定理 **2**　设 s 是任一主体, o 是任一客体, 则

(1) $(\neg\lambda(r(s, o))\wedge L(s)>L(o))\rightarrow\neg r(s, o)$;

(2) $(\neg\lambda(e(s, o))\wedge L(s)>L(o))\rightarrow\neg e(s, o)$;

(3) $(\neg\lambda(a(s, o))\wedge L(s)<L(o))\rightarrow\neg a(s, o)$;

(4) $(\neg\lambda(w(s, o))\wedge L(s)\neq L(o))\rightarrow\neg w(s, o)$.

即完整级可有效地控制添加和写操作. 我们称之为完整性定理.

证　(1) 设$\neg\lambda(r(s, o))$、$L(s)>L(o)$和 $r(s, o)$同时成立. 则由规则 8, 有 $\partial r(s, o)$成立. 再由规则 5 得$\partial r(s, o)\rightarrow((o\in D(s)\wedge C(s)\leq C(o)\wedge L(s)\leq L(o))\vee\lambda(r(s, o)))$, 故有 $L(s)\leq L(o)$, 这与 $L(s)>L(o)$矛盾. 同理可证(2)~(4)的情形.

定理 **3**　$\forall s\forall o(G(s)\cap G(s')=\{\theta\}\wedge o\in D(s)\backslash D(\theta))\rightarrow\neg x(s', o)$, 这里 x 是任一基本操作. 即一个组中的主体不能对其他组的辖域中的非公共客体进行任何基本操

作. 我们称之为管辖性定理. 这里 $o\in D(s)\backslash D(\theta)$)指 $o\in D(s)$但 $o\notin D(\theta)$.

证 设 $G(s)\cap G(s')=\{\theta\}\wedge o\in D(s)\backslash D(\theta)$、$\forall s''\neg\lambda(s'', \partial x(s, o))$且 $x(s', o)$同时成立, 则由规则 8 得, $\partial x(s', o)$, 令 $x = r$, 再由定理 1 可得: $(C(s')\geq C(o)\wedge o\in D(s'))$, 从而有 $o\in D(s')$. 于是有 $o\in D(s')\cap(D(s)\backslash D(\theta))=(D(s')\cap D(s))\backslash D(\theta)$. 这与引理 9 矛盾.

同理可对其他几种操作$(x = e, a, w)$进行证明. 略.

由上面的规则和定理可知, 任何操作都取决于主体自身的权限以及可信主体π 的授权与否. 只要最初将用户分配在合适的组别中, 且赋予每个用户合适的权限, 则主体对客体的操作将同时符合机密性和完整性的要求. 所以系统的安全性归结为, 最初对用户的赋权以及特殊主体π 所产生的指定本身是否合理, 是否存在漏洞.

对用户权限的管理是容易实现的. 但保证可信主体总是产生合理的指定却较难实现. 但这并不等于说不能实现. 从实际应用来说, 完全可以有一个可信的应用程序, 负责检查客体是否存在病毒(如杀毒程序), 则经检查无毒后, 可以适当提升其完整级.

当然, 如果可信主体的判断出现差错, 如杀毒程序不能有效杀毒, 则系统仍存在安全风险. 从这个意义上说, 本模型并不是“绝对安全”的, 而是“条件安全”的——即在可信主体能有效工作的前提下是安全的. 这实际上证明了基于可信根的可信计算模型的安全性, 本模型的好处还在于, 它给出了一个操作系统的安全框架, 同时把安全的核心归结为可信主体π的工作规则的合理性上.

4 结束语

本文提出了一种新的操作系统安全模型, 把防病毒、检查机密性等任务交给专门的可信应用程序, 从而使模型大为简化, 并解决了机密性与完整性二者兼顾的问题.

不过本文只给出了强制安全规则, 还未涉及对用户的授权管理. 同时, 限于篇幅, 本文未对强制规则作为公理的一致性进行分析. 此外, 正如前面提到的, 可信主体应该依据什么样的规则进行工作, 这些都是在本模型的基础需要进一步研究的问题.

参 考 文 献

1 卿斯汉, 刘文清, 刘海峰. 操作系统安全导论. 北京: 科学出版社, 2003
2 南湘浩, 陈 中. 网络安全技术概论. 北京: 国防工业出版社, 2003
3 Bell D E, LaPadula L J. Secure computer systems: A mathematical model. Technical Report M74-244, The MITRE Corporatoin. 1973
4 Biba K J. Integrity considerations for secure computer system. Technical Report, ESD-76-372, Bedford. MA: PSAF Electronic System Division, Hanscom Air Force Base, 1977
5 Denning D. A Lattics model of secure information flow. Commnication of The ACM, 1976, 19(5): 236~250
6 Goguen J A, Meseguer J. Security policy and security models. In: Proceedings of the 1982 IEEE Symposium on Security and Privacy. 11-20, 1982