计算机安全中的经典模型*)

周 伟 尹 青 王清贤
(信息工程大学网络工程实验室 郑州450002)

摘 要 安全模型是构造安全计算机系统的基础。到目前为止,已有多种公开发表的安全模型。本文总结了几种重要的早期安全模型,其中包括访问矩阵模型,HRU 模型,BLP 模型、格模型和无干扰模型,并对它们进行了分类和简要评述。本文将它们称为经典安全模型。这些经典安全模型都是开创性的,从各个不同的方面对安全问题进行抽象,模型所定义的安全问题具有典型性,并对后续的研究产生了重要影响。目前,共享计算机系统的安全问题仍然是计算机科学的中心问题之一,研究这些经典模型,对于我们全面理解计算机系统的安全问题,展望未来发展方向,具有重要意义。

关键词 安全模型,访问控制模型,信息流模型,格模型,无干扰模型

Typical Models of Computer Security

ZHOU Wei YIN Qing WANG Qing-Xian
(Network Engineering Laboratory, Information Engineering University, Zhengzhou 450002)

Abstract Security models lay the groundwork on which the secure computer systems are found. In this paper, we review several earlier seminal models, including the Access-Matrix, the HRU model, the Bell and LaPadula model, Denning's Lattice model, J. Goguen and J. Meseguer's non-interference model. We call these models typical because they or their basic concepts serve as key components of computer security. We start by discussing the classification of the typical models, then review each of them, and make brief comments on them. Since security is one of the central problems in computer science, the restudy of these typical models are very important for us to seize the essentials of computer security.

Keywords Security model, Access control model, Lattice model, Non-interference model

1 引言

1972年,James P. Anderson[1,2]指出,要开发安全系统,首先必须建立系统的安全模型。安全模型给出安全系统的形式化定义,正确地综合系统的各类因素,包括系统的使用方式、使用环境类型、授权的定义、受控制的共享等,这些因素构成安全系统的形式化抽象描述。安全模型允许使用数学方法证明系统是否安全,或发现系统的安全缺陷。

安全模型的目的是保护计算机系统的机密性和完整性,其研究可追朔到20世纪60年代,随着支持远程访问和分时多用户系统的使用,共享计算机系统的安全问题引起重视。安全问题一直是计算机科学的中心问题之一。在三十多年的发展中,先后提出了多种安全模型,其中一些主要模型形成于20世纪70年代到80年代初期,如著名的访问矩阵模型[12]、BLP 模型[3~5]、HRU[11]模型、格模型[7]、无干扰模型[9,10]等等。这些早期安全模型从各个不同的方面对安全问题进行抽象,对安全系统的研究与开发具有重要的意义,本文称这些安全模型为经典模型。研究这些模型,对于我们全面理解计算机系统的安全问题具有重要意义。

2 经典模型及分类

2.1 经典模型

最早的形式化安全模型可追朔到 C. Weissman 1969年发表的高水标(High-Water-Mark)模型[14]。迄今已有多种安全模型发表于公开刊物上。1981年,Landwehr 综述了10种安全模型[14],其中重点评述了高水标模型、访问矩阵(Access Matrix)模型、BLP 模型和格(Lattice)模型。Landwehr 将安全模型近似地分为三类:访问控制模型、信息流模型和程序通道模型。程序通道模型所面向的安全问题是信息流的演绎性质,即系统中实体通过观察系统部分行为,从中推断系统的其它行为。这种性质本质仍属信息流的干扰性质。1994年,John McLean 在他的论文中[16]讨论了两类三种具有重要影响的模型,HRU 模型、BLP 模型和无干扰模型,同时还提到了另外9种模型,其中除 Clark-Wilson 模型外,其它8种基本上都是上述三种模型的扩展或限制。Silvana Castano 等人在他们的著作[6]中综述了11种安全模型,将他们分为两类,即自主访问控制和强制访问控制。

表1列出本文认为最重要的几种早期安全模型,并将它们称为经典模型。本文选择这些模型作为经典,基于以下理由:(1)模型是开创性的;(2)模型所定义的安全问题具有典型性;(3)模型对后续的研究具有重要影响;(4)它们的研究基本覆盖了安全研究的各个侧面。

2.2 经典模型分类

安全模型可分为两大类:访问控制模型和信息流模型。按

*)基金项目:国家863计划信息安全技术主题"网络安全及防御技术"(编号:863-104-06-1)。周 伟 博士研究生,研究方向:软件理论与技术,网络安全。尹 青 博士研究生,研究方向:软件理论与技术,网络安全。王清贤 教授,博士导师,研究方向:算法分析与设计,网络安全。

此分类的经典模型见表1。最早期的形式化模型基本上都属于访问控制模型。访问控制模型，也可形象地称为“公文世界模型”，是现实公文世界保密管理机制的数学抽象，直接背景是军事安全的组织模型[14]。访问控制模型在操作系统和数据库系统中都有典型的应用，如 Unix 操作系统中的访问控制机制[20]。

访问控制模型又可分为两种，自主访问控制(DAC)和强制访问控制(MAC)[8]。自主访问控制是最常用的一类模型，它基于客体-主体间的所属关系，根据主体所属的组来限制对客体的访问。所谓自主，是指主体可以根据自己的意愿，将访问控制权限授予其他主体，或从其他主体那里收回访问权限。也就是说，DAC 的基本思想是将用户作为客体的拥有者，他有权自主地决定哪些用户可以访问他的客体。在强制访问控制模型中，主体不能修改访问权，也不能将自己的访问权授予其他主体。MAC 的最初目标是军事领域安全，防止非法获取敏感信息，因此在 MAC 模型中，主体和客体都有固定的安全属性，这些安全属性由系统设置，并只由系统使用，系统按主体和客体的安全级别决定主体是否有权访问客体。

访问控制模型试图控制系统的行为来保证系统的安全性。与访问控制模型不同，信息流模型的目的是控制系统内的信息流，防止未经许可的信息流。因此，信息流模型所研究的是信息在各系统实体的流动以及由此所产生的系统实体间的相互关系。在理论上，信息流模型可以有效地防止隐蔽信息流。与访问控制模型相比，信息流模型在本质上更抽象。

本文讨论了两种信息流模型，格模型和无干扰模型。Dorothy E. Denning[7]提出使用格模型定义信息流的控制结构，将信息流抽象为安全等级间的偏序关系，在这种偏序关系下，安全等级构成一个全有界的格。由于格本身是纯数学结构，所以格模型是一类更为抽象的安全模型。我们知道，BLP 模型经常被批评为没有彻底形式化，因为在 BLP 模型中，作为原子操作的“读”和“写”很难被形式化。

无干扰模型也被称为“完美模型(perfect model)”[16]，是由 Joseph A. Goguen 和 Jose Meseguer 在他们的开创性论文[9,10]中提出的一种形式化信息流模型。直观地说，所谓无干扰，是指系统一个用户在系统中的行为不会影响另一个用户所观察到的系统行为。Goguen 和 Meseguer 的无干扰概念可以表述为：为保证不存在从用户 A 到用户 B 的信息流，只需要保证 A 的行为不会影响 B 的观察视图，即 B 所看到的系统行为与 A 的行为无关。

表1 经典模型及分类

模型分类		模型	作者	年代	说明
访问控制模型	自主访问控制	Access Matrix Model	Butler W. Lampson	1971	机密性
		HRU Model	Michael A. Harrison, Walter L. Ruzzo Jeffrey D. Ullman	1976	机密性
	强制访问控制	BLP Model	D. Elliott Bell, Leonard J. LaPadula	1973	机密性
信息流模型	信息流控制	Lattice Model	Dorothy E. Denning	1975	机密性 完整性
	完美安全模型	Non-interference Mode	Joseph A. Goguen, Jose Meseguer	1982	机密性 完整性

3 访问控制模型

3.1 访问矩阵

B. W. Lampson 使用主体、客体和访问矩阵等概念，第一次对访问控制问题进行形式化抽象[12]。该模型相当简单，而且非常一般，所以被广泛应用。访问矩阵包含三类主要对象：客体集 O，表示访问操作中的被动实体；主体集 S，表示访问操作中的主动实体；访问规则集 R，也称权限集，它规定主体对客体的操作。典型的客体是文件、终端、设备、以及其它一些操作系统实体。主体可能同时也是客体，即 $S\subseteq O$，其它主体可对其执行读操作或其它操作。访问矩阵是一个正交数组，每个矩阵元素表示主体对客体的访问方式。访问矩阵可形式地定义如下：

$M: S\times O\rightarrow 2^R$

典型的访问方式包括读、写、添加、执行以及所属关系。实际上，访问矩阵还定义了系统的当前安全状态，系统的所有安全状态可被表示为一个矩阵阵列。特定的操作引起系统安全状态的转移，并进入下一个安全保护。如文件主删除文件后，与该文件相应的矩阵行也被删除，由此得到的新矩阵表示了后继的安全状态。

形式地，我们以 $Q_i=(S_i,O_i,M_i)$ 表示系统的状态，也称为格局(configuration)。操作引起的系统状态转移可以形式地表示为

$Q_i\Rightarrow_{op}Q_{i+1}$

即执行操作 op 后，系统格局的变化。由于访问矩阵模型是对系统状态的抽象，所以它与安全策略的执行机制无关，因此，该模型实现了策略执行机制与策略本身的分离。

3.2 HRU 模型

1976 年，M. A. Harrison、W. L. Ruzzo 和 J. D. Ullman 发表了他们关于操作系统保护的基本理论[11]，描述了 HRU 模型。模型使用的基本概念包括：主体、客体、访问矩阵、格局、操作、命令和保护系统，其中访问矩阵是 HRU 模型中的最主要的概念之一。保护系统的定义如下：

HRU 模型的一个保护系统由以下两部分构成：

(1)访问规则集 R，在 HRU 模型中称为权限集；

(2)命令集 C，其命令形式为：

```
command C(X1,…,Xk)
        if r1 in M[Xs1,Xo1] ∧ … ∧ rm in M[Xsm,Xom]
        then op1,…,opn
end
```

其中，$X_1,\cdots,X_k$ 是形式参数，$r_1,\cdots,r_m$ 是访问权限，$s_1,\cdots,s_m$ 和 $o_1,\cdots o_m$ 是从1到 k 的正整数。如果 $m=0$，命令没有条件部分，可简单地写成以下非条件命令：

```
command C(X1,…,Xk)
        op1,…,opn
end
```

每个操作 $op_1,\cdots,op_n$ 是表 HRU 中的六个原语操作之一。

原语操作会改保护变系统的格局。格局 (S,O,M) 在操作 op 作用下转移到格局 (S',O',M')，记作 $(S,O,M)\Rightarrow_{op}(S',O',M')$。完成这一转移的规则全部列在表 HRU 中。

每个命令都封装了一组原语操作，因此，命令会引起系统格局的系列变化。如果有一组参数 $x_1,\cdots,x_k$，当系统执行特定的命令 $C(x_1,\cdots,x_k)$ 时，系统格局会产生下列变化，即存在一个格局序列 $Q_1,\cdots,Q_n$，使得

$Q=Q_1\Rightarrow_{op_1}Q_2\Rightarrow_{op_2}\cdots\Rightarrow_{op_n}Q_n=Q'$

则称 $Q\vdash_C Q'$，其中 op_i 表示以实参 $x_1,\cdots,x_k$ 替换形参 $X_1,\cdots,X_k$ 后，命令 C 所执行的原语操作。在下面的定义中，用记号 Q

$\vdash^{*}Q'$ 表示 Q 执行0次或多次命令后转移到格局 Q'。

表 HRU 原语操作及转移规则

原语操作	解释	结构格局(S',O',M')
enter r into $M[s,o]$	授予主体 s 对 O 访问权 r	$S'=S, O'=O, M'[s,o]=M[s,o]\cup\{r\}$当 $s'\neq s, o'\neq o$ 时，$M'[s',o']=M[s',o']$
delete r from $M[s,o]$	收回主体 S 对 O 访问权 r	$S'=S, O'=O, M'[s,o]=M[s,o]-\{r\}$当 $s'\neq s, o'\neq o$ 时，$M'[s',o']=M[s',o']$
create subject s	增加一个新主体 S	$S'=S\cup\{s\}, O'=O\cup\{s\}$对所有 $s\in S, o\in O, M'[s,o]=M[s,o]$否则 $M'[s,o]=\varnothing$
destroy subject s	删除一个旧主体 S	$S'=S-\{s\}, O'=O-\{s\}$对所有 $s\in S', o\in O', M'[s,o]=M[s,o]$
create object o	增加一个新客体 O	$S'=S, O'=O\cup\{o\}$对所有 $s\in S, o\in O, M'[s,o]=M[s,o]$否则 $M'[s,o]=\varnothing$
destroy object o	删除一个旧客体 O	$S'=S, O'=O-\{s\}$对所有 $s\in S', o\in O', M'[s,o]=M[s,o]$

定义(HRU1) 假设给定一个保护系统中的权限 r，如果存在格局 Q 和一个命令 C，系统从初始格局 Q_0 出发，使得：

(1)$Q_0\vdash^{*}Q$，即存在命令串 $C_1,\cdots,C_n$，和格局 $Q_1,\cdots,Q_n$，$Q_0\vdash c_1Q_1\vdash c_2\cdots\vdash c_nQ_n=Q$；

(2)C 在 Q 上泄露权限 r，即 C 中有原语将 r 加入到访问矩阵中原来不含 r 的元素中；则称 Q_0 对 r 是不安全的。如果 Q_0 对 r 不是不安全的，则称 Q_0 对 r 是安全的。

对于由 HRU 定义的保护系统，Harrison、Ruzzo 和 Ullman 证明了三个定理来说明安全问题的复杂性。Harrison、Ruzzo 和 Ullman 提出的安全问题是：在一个给定的保护系统中，一个给定的格局对某个特定的权限是安全的吗？下面这个问题被称为 HRU 安全问题。

定理 HRU1 HRU 安全问题是不可判定的。

定理 HRU2 若保护系统中每个命令只包含一个原语操作，则 HRU 安全问题是可判定的。

定理 HRU3 如果将保护系统限制为没有 create 操作，则 HRU 安全问题是 PSPACE 完全的。

3.3 BLP 模型

BLP 模型是第一个形式化的多级安全模型。BLP 模型在所有的安全级之间定义了一个偏序，主体能否访问客体由两者的当前安全级决定。在 BLP 模型中，使用访问控制矩阵对访问模式进行补充约束，决定一个任意主体对一个任意客体的访问模式。表 BLP1列出了下面讨论中所用到主要元素。表 BLP2列出了 BLP 模型的安全公理(安全特性)。

表 BLP1 BLP 模型主要元素

集合	语义	集合	语义
S	主体集：进程，正在执行的程序	R	请求集：请求/释放访问；授予/撤销访问；请求创建客体/重分级等
O	客体集：数据，文件，主体等		
A	$A=\{a,e,r,w\}$：访问属性集，其中 a 添加；r 读；w 写；e 执行	V	$V=B\times M\times F\times H$；$v=(b,M,f,H)\in V$ 表示系统状态
M	访问矩阵：表示自主访问控制模式，$M[s,o]\subseteq A$ 主体 s 对客体 o 的权限	W	$W\subset R\times D\times V\times V$；$w=(r,d,v,v')$表示在请求 r 下，产生决策 d，系统从状态 v 转移到 v'
H	客体的层次结构		
F	F 是等级函数(f_s,f_o,f_c)的集合，其中：f_s：主体等级函数；f_o：客体等级函数；f_c 当前安全等级函数	X	$X=\{x;T\to R\}$：T 到 R 的函数集，x 表示一个请求序列
B	$B\in\mathbb{P}(S\times O\times A)$：当前访问集合	Y	$Y=\{y;T\to D\}$：T 到 D 的函数集，y 表示一个决策序列
T	正整数集，表示离散时间集合	Z	$Z=\{z;T\to V\}$：T 到 V 的函数集，z 表示一个状态序列
D	$D=\{yes,no,error,?\}$：决策集		

表 BLP2 BLP 模型的安全公理

状态的安全特性	条件：对于 $v=(b,M,f,H)$和所有$(s,o,x)\in b$
ss-安全特性	简单安全特性：v 是 ss-安全的$\Leftrightarrow$(1)$x=a$ 或 $x=e$；(2)$x=r$ 或 $x=w\Rightarrow f_s(s)\geqslant f_o(o)$
$*$-安全特性	$*$-安全特性：v 是 $*$-安全的$\Leftrightarrow$(1)$x=a\Rightarrow f_s(s)\leqslant f_o(o)$；(2)$x=w\Rightarrow f_s(s)=f_o(o)$；(3)$x=r\Rightarrow f_s(s)\geqslant f_o(o)$
ds-安全特性	自主安全特性：v 是 ds-安全的$\Leftrightarrow x\in M[s,o]$
$(*,S')$-安全特性	对于 $S'\subset s$，v 关于 S' 是 $*$-安全的，当 $s\in S'$时(1)$x=a\Rightarrow f_s(s)\leqslant f_o(o)$；(2)$x=w\Rightarrow f_s(s)=f_o(o)$；(3)$x=r\Rightarrow f_s(s)\geqslant f_o(o)$
(ss,l)-安全特性	对于 $l=(l_s,l_o,l_c)\in F$，v 关于 l 是 ss-安全的$\Leftrightarrow$(1)$x=a$ 或 $x=e$；(2)$x=r$ 或 $x=w\Rightarrow l_s(s)\geqslant l_o(o)$

在 BLP 模型中，一个 BLP 系统 $\Sigma(R,D,W,z_0)\subset X\times Y\times Z$ 定义如下：

$(x,y,z)\in\Sigma(R,D,W,z_0)$当且仅当对任意 $t\in T$，$(x_t,y_t,z_t,z_{t-1})\in W$。其中

(1)z_0称为系统的初始状态；

(2)每个三元组$(x,y,z)\in\Sigma(R,D,W,z_0)$称为系统的表象；

(3)每个四元组(x_t,y_t,z_t,z_{t-1})称为系统的动作。

系统表象$(x,y,z)\in\Sigma(R,D,W,z_0)$满足 ss-特性，如果状态序列 $z=(z_0,z_1,\cdots)$中的每个状态都满足该特性。系统满足 ss-特性，如果它的每个表象都满足 ss-特性。类似地，可以定义系统满足 $*$-特性和 ds-特性。

BLP 系统的安全性，由 BLP 的四条安全定理来保证，这四条安全定理如下：

简单安全性定理 当 z_0满足 ss-特性时，$\Sigma(R,D,W,z_0)$满足 ss-特性的充分必要条件是，W 满足以下条件，即对任意 $w=(R,D,(b^*,M^*,f^*,H^*),(b,M,f,H))$：

(1)每个$(s,o,x)\in b^*-b$ 满足(ss,f^*)-特性。

(2)如果$(s,o,x)\in b$ 但不满足(ss,f^*)-特性，则$(s,o,x)\notin b^*$。

$*$安全性定理 设 $S'\subset S$，初始状态 z_0满足$(*,S')$-特性，那么 $\Sigma(R,D,W,z_0)$满足$(*,S')$-特性和充分必要条件是 W 满足以下条件，即对任意 $w=(R,D,(b^*,M^*,f^*,H^*),$

$(b,M,f,H))$:

(1)对每个 $s\in S'$,$(s,o,x)\in b^*-b$ 满足$(*,S')$-特性。

(2)对每个 $s\in S'$,如果$(s,o,x)\in b$ 不满足$(*,S')$-特性,则$(s,o,x)\notin b^*$。

自主安全性定理 $\Sigma(R,D,W,z_0)$满足 ds-特性,当且仅当,z_0满足 ds-特性,且对于每个活动 $w=(R,D,(b^*,M^*,f^*,H^*),(b,M,f,H))$,$W$ 满足:

(1)如果$(s,o,x)\in b^*-b$,则 $x\in M^*[s,o]$。

(2)如果$(s,o,x)\in b$ 且 $x\notin M^*[s,o]$,则$(s,o,x)\notin b^*$。

基本安全定理 $\Sigma(R,D,W,z_0)$是安全系统,当且仅当,z_0是安全状态,且 W 的活动满足简单安全性定理、* 安全性定理和自主安全性定理。

3.4 评述

访问控制模型依据的基本原理是:如果在系统执行的每一步,系统中实体的行为都符合指定的规范,则系统是安全的。但 HRU 的结果表明,当系统以访问矩阵作为访问控制机制时,一个进程能否取得对客体访问权这一问题,是不可判定的。因为 HRU 模型是一类简单的模型,并部分将授权的自由交给用户,所以,HRU 的结果还表明,安全必须以牺牲自由为代价。

与 HRU 不可判定定理相关的另一个问题是所谓"特洛伊木马"问题。因为用户可以自主地将访问权授予其它用户,所以客体的访问权可能在属主不知道的情况下被授予了其它用户。自主访问控制不能防止"特洛伊木马"。对"特洛伊木马"问题的处理,将访问控制区分为自主访问控制和强制访问控制。强制访问控制模型可以阻止使用直接信息通道的"特洛伊木马",但不能防止隐蔽信息通道,即访问权限某种特殊组合产生的隐蔽信息流。McLean 指出,访问控制模型来自"公文世界",因此,隐蔽通道问题是不可避免的[16]。

因为有 BLP 基本安全定理,所以我们相信 BLP 模型是安全的。但 McLean 却指出[16],在证明系统安全性方面,BLP 基本安全定理并没有任何作为。如果系统状态的索引支持归纳证明,那么不论如何定义"安全状态",类似的定理都成立。如此一来,基本安全定理证明的只不过是状态索引的性质,而非安全性。为此,McLean 构造了一个明显不安全的系统,并在这样的系统中证明了基本安全定理。这说明,对于 BLP 这类的模型,安全是一种平凡性质,从数学上证明是安全的系统,不一定表示实际系统也是安全的。

4 信息流模型

隐蔽通道的概念,是 Butler W. Lampson 在研究禁闭问题时提出的[13]。因为访问控制模型仅考虑主体对客体的访问行为,而不考虑信息的流动问题,所以此类模型不能检查信息的间接流动,因此也就不能消除隐蔽信息通道。例如对于 if a=0 then b=c 这样的语句,信息流关系 c→b 是显式的,而 a→b 则是隐式的,访问控制只能检查到前者。信息流模型可以处理这类间接的信息流动。

4.1 格模型

格模型由 Dorothy E. Denning 提出。一个格信息流模型 FM 定义为[7]

$$FM=(N,P,SC,\oplus,\rightarrow)$$

其中:$N=\{a,b,\cdots\}$是客体集,即逻辑存储对象或信息贮藏器的集合。N 中的元素,根据模型设计详细程度,可以是文件,也可以是程序的变量。系统中的每个用户也可以视为客体。$P=\{p,q,\cdots\}$,是进程的集合。进程是引起信息流动的主动主体。$SC=\{A,B,\cdots\}$是安全级的集合,是信息的不相交分类的集合。定义这一集合的目的,是要把"安全保密级别"、"保密等级"、"需知权"等概念包容到信息流模型中来。每个客体 a 属于一个安全级,标记为 $\underline{a}$,表明存储于 a 中信息的安全等级。有两种把客体绑定到安全级上方法:静态绑定和动态绑定。对于静态绑定,客体的安全级别是恒定不变的。动态绑定时,客体的安全级别可随其内容变化。用户被静态地绑定到称为"安全级别"的安全等级上。进程 p 也要绑定到安全级上,标记为 $\underline{p}$。$\underline{p}$ 可能取决于 p 的属主用户的安全等级,或 p 所访问过的客体和安全等级。等级联合算子⊕是可结合和可交换的二元算子。→是一个自反、传递和非对称的二元关系,称为流关系,$A\rightarrow B$ 表示允许从 A 到 B 的信息流。

模型的主要概念是全有界格。一个全有界格是指这样一种代数结构:一个有最小上界算子和最大下界算子的有限偏序集。在下面的假设之下,$(SC,\rightarrow\oplus)$是一个全有界格。

(1)$(SC,\rightarrow)$是偏序集。

(2)SC 是有限集。

(3)SC 有下界 L,且对所有 $A\in SC$,$L\rightarrow A$。

(4)⊕是 SC 上的最小上界算子。

在一般系统中。上述假设是合理性。可以证明,上述假设蕴含这样一个结果:SC 有最大下界 L 和最小上界 H。最小上界算子⊕可以扩展到子集上。若 X 是 SC 的子集,即 $X\subseteq SC$,如果 $X=\varnothing$则令$\oplus X=L$,否则令$\oplus X$ 表示 X 中所有安全级的最小上界,即对任意 $n>1$和 $X=\{A_1,\cdots,A_n\}$,令$\oplus X=A_1\oplus\cdots\oplus A_n$。

因为$(SC,\rightarrow,\oplus)$有最大下界,所以可以引入最大下界算子⊗,它定义为 $A\otimes B=\oplus\{C\mid C\rightarrow A \text{ and } C\rightarrow B\}$。⊗也可扩展到集合上去。如果 $X=\varnothing$,则$\otimes X=H$,否则,令$\otimes X$ 表示 X 中所有安全级的最大下界,即对任意 $n>1$和 $X=\{B_1,\cdots,B_n\}$,令$\otimes X=B_1\otimes\cdots\otimes B_n$。

这样,$(SC,\rightarrow,\oplus,\otimes)$是一个有最小上界和最大下界的全有界格。最小上界算子和最大下界算子还有下面的性质:

(1)$A_i\rightarrow B(1\leqslant i\leqslant n)$当且仅当$\oplus X\rightarrow B$,或 $A_1\oplus\cdots\oplus A_n\rightarrow B$。意思是说,$a_1,\cdots,a_n$ 客体可分别独立地流向客体 b,当且仅当 $\underline{a_1}\oplus\cdots\oplus\underline{a_n}\rightarrow\underline{b}$,即 $a_1,\cdots,a_n$ 的组合所对应的安全级,可以流向 b。

(2)$A\rightarrow B_i(1\leqslant i\leqslant n)$,当且仅当 $A\rightarrow\otimes X$,或 $A\rightarrow B_1\otimes\cdots\otimes B_n$,意思是说,客体 a 能够流到客体 $b_1\cdots,b_n$,当且仅当 $\underline{a}\rightarrow\underline{b_1}\otimes\cdots\otimes\underline{b_n}$。

有了以上概念,就可以定义什么是安全信息流了:信息流模型 FM 是安全的,如果操作序列不会产生违反"→"关系的信息流。

使用格模型,我们可以构造 if c then S_1的安全模型,更一般地,可以构造 case c of $S_1,\cdots,S_n$ 的安全模型,因为前者是后者的特殊情形。以 $c:S_1,\cdots,S_n$ 表示上述语句。首先归纳地定义语句 S 的结构:

(1)赋值语句 $S\triangleq b=f(a_1,\cdots,a_n)$是原子语句,其中 $a_1,\cdots,a_n,b$ 是客体(变量),f 是求值函数。所有原子语句都是语句。

(2)如果 S_1和 S_2是语句,则 $S\triangleq S_1;S_2$也是语句;

(3)如果 $S_1,\cdots,S_n$ 是语句,c 是变量,则 $S\triangleq c:S_1,\cdots,S_n$ 也是语句。

每个 S 都包含信息的流动。令 $In(S)$表示 S 中信息流入

的客体安全级的集合，$Out(S)$表示信息流出的客体安全级的集合，那么，归纳地定义 $In(S)$和 $Out(S)$定义如下：

(1)如果 S≙$b=f(a_1,\cdots,a_n)$是赋值语句，那么 S 是安全的，当且仅当则 $In(S)=\{\underline{b}\}$，$Out(S)=\{\underline{a_1},\cdots,\underline{a_n}\}$；

(2)如果 S≙$S_1;S_2$，则 $In(S)=In(S_1)\cup In(S_2)$，$Out(S)=Out(S_1)\cup Out(S_2)$；

(3)如果 S≙c：$S_1,\cdots,S_n$，则 $In(S)=\bigcup_{i=1}^{n}In(S_i)$，$Out(S)=\bigcup_{i=1}^{n}Out(S_1)\cup\{C\}$；

现在可以定义语句 S 的安全性如下：语句 S 是安全的，当且仅当$\oplus Out(S)\rightarrow\otimes In(S)$。显然按照这一关于语句 S 的安全模型，语句 if a=0 the b=c 是安全的，当且仅当 $\underline{a}\oplus\underline{c}\rightarrow\underline{b}$。

4.2 无干扰模型

无干扰模型是 Goguen 和 Meseguer 针对多级安全提出的信息流安全模型[9,10]，在本文中，我们将定义无干扰性质使用的自动机模型称为 GM 自动机，其定义如下：

定义(NI1) 一个 GM 自动机 M 由以下诸要素构成：

(1-1)集合 U，其中的元素称为"用户"。

(1-2)集合 S，其中的元素称为"状态"。它实际是对用户程序、数据、消息等信息的完整刻画。

(1-3)集合 C，表示导致状态改变的操作的集合，其中的元素称为"状态操作"。

(1-4)集合 O，其中的元素称为"输出"。一个输出串实际上表示用户所观察到的一次系统行为。

(1-5)集合 R，读操作的集合，其中的元素称为"读操作"。以及

(2-1)函数 $do:S\times U\times C\rightarrow S$，称为状态转移函数，表示用户执行状态操作后所发生的状态变化。

(2-2)函数 $out:S\times U\times R\rightarrow O$，称为输出函数，表示用户执行读操作的可见结果。

表 GM1的左边列出了 GM 状态自动机的基本元素，右边列出了状态自动机的状态转移函数与输出函数。对于从初始状态 s_0出发，经历操作序列 $w\in(U\times C)^*$后，所达到的状态，Goguen 和 Meseguer 使用了一个简单记号 $[\![w]\!]$，即 $[\![w]\!]=do^*(s_0,w)$。因此，$Out([\![w]\!],v,r)$表示用户 v 在状态 $s=[\![w]\!]=do^*(s_0,w)$执行读 r 的结果。

对于操作序列 w，我们定义 w 上的过滤函数 $P_{G,A}(w)$为从 w 中删除 $u\in G$ 且 $c\in A$ 的序对(u,c)所得到的子序列，即：

当 $u\in G$ 并且 $c\in A$ 时，$P_{G,A}((u,c)w)=P_G(w)$。

当 $u\notin G$ 或者 $c\notin A$ 时，$P_{G,A}((u,c)w)=(u,c)P_{G,A}(w)$。

且分别在 $A=C$ 或 $G=U$ 或 $G=\{u\}$，$A=C$ 时，我们将 $P_{G,A}$分别简记成 P_G,P_A,P_*。

我们可以这样的观点来考察 GM 自动机：系统中有两个我们感兴趣的用户组 G 和 G'，其中 G 在系统中执行状态操作子集 A 中操作行为，G'希望观察 G 在系统中的行为，即 G 执行了哪些状态操作。G'所能使用的方法是执行读操作子集 B 中的读操作，然后输出结果。我们称 G 在 A 上的操作 G'对 B 在上观察无干扰，如果不论 G 执行什么样的操作，G'所观察到的结果都是一样的，就像 G 从来没有执行过任何 A 中的状态操作。形式地，无干扰性质定义如下：

定义(NI2) 执行状态转移命令子集 A 的用户组 G，对另一个执行输出命令子集 B 的用户组 G'无干扰，记为：

$G,A:|G',B$

其表达式为：对任意 $w\in(U\times C)^*$，$v\in G'$，$r\in B$

$$out([\![w]\!],v,r)=out([\![P_{G,A}(w)]\!],v,r)$$

其中 p 是过滤函数。

表 GM1 GM 自动机的基本元素及动作函数

基本元素	操作
U：用户集合	$w=(u_1,c_1)\cdots(u_n,c_n)\in(U\times C)^*$：系统经历的状态操作序列，$empty$ 表示空序列。$w=w_1w_2$表示两个序列的连接。
$G\subseteq U$：用户组	
$G'\subseteq U$：用户组	$do:S\times U\times C\rightarrow S$：状态转移函数。 $do(s,u,c)=s'$：用户 u 在状态 s 下执行状态操作 c，系统转移到状态 s'。
C：状态操作集	
$A\subseteq C$：状态操作子集	
R：读操作集	$do^*:S\times(U\times C)^*\rightarrow S$：广义状态转移函数。 $do^*(s,empty)=s$，$do^*(s,w'(u,c))=do(do^*(s,w),(u,c))$
$B\subseteq R$：读操作子集	
S：系统状态集合	$out:S\times U\times R\rightarrow O$：输出函数。 $out(s,u,r)=0$：用户 u 在状态 s 下执行读操作 r，并将结果 o 输出。
$s_0\in S$：初始状态	

根据上述定义，我们可以构造多级安全(MLS)系统。设 L 是安全等级的有序集，并设函数

$level:U\rightarrow L$

定义用户的安全等级。那么一个 MLS 系统是满足以下(MLNI)性质的系统：

$$(MLNI)\ out([\![w]\!],v,r)\neq out([\![P_*(w)]\!],v,r)\Rightarrow level(u)\leqslant level(v)$$

意思是说，只要 u 的安全级比 v 高，则 u 对 v 无干扰。

4.3 评述

不论是 Denning 的格模型，还是 Goguen 和 Meseguer 的无干扰模型，都使用了纯数学概念，因此，可以说信息流模型在形式化方面迈出了重要一步。在理论上，信息流模型可以消除隐蔽通道，对于格模型，这一点可直接从4.1节中看到。对于无干扰模型，隐蔽通道实际上已经对低安全级用户的观察造成了影响，因此将被(MLNI)禁止。

Ryan 等人认为，无干扰模型对于定义什么是安全来说更为本质，它将安全问题与计算机科学中的另一个中心问题，即系统行为的等价性联系到一起。安全问题实质上与系统行为的等价问题是一致的[19]，而后者可以使用纯数学的方法描述。所谓两个系统的行为等价，是指这两个系统的行为对观察者来说是一样的。对于确定性的系统，定义等价性是很简单的，如 Goguen 和 Mesaguer 所考察的系统。对于非确定性系统，目前不存在一致公认的等价性定义[17]，因此，也就不存在一致公认的无干扰性定义[18,19]。

结束语 访问控制是比较成熟的系统安全技术，但是，从访问控制的角度寻找一种安全访问控制策略，并保证系统中不存在信息泄漏，却不是一件易事。访问控制模型所要面对的主要问题是如何防止"特洛伊木马"。C2级操作系统如 Unix 和 Windows NT，都以自主访问控制作为 TCB 主要安全机制，因而不能防止内部攻击。如 Unix 的文件管理系统允许用户决定文件的访问权，这几乎可以肯定要导致系统的安全漏洞，假如某个用户 A 执行了一个能修改其文件访问权限的"特洛伊木马"程序，就可能将 A 的文件的访问权限授予任意用户。

对于强制访问控制，同样的"特洛伊木马"将被禁止，因为在 MAC 模型中，规则是由系统强制实施的。按照 BLP 模型，"特洛伊木马"程序不能将高安全级的信息泄漏到低安全级用户，因为它违反"禁止向上读"或"禁止向下写"原则。但如果"特洛伊木马"利用系统的功能，间接地传递秘密信息，则 MAC 不能阻止这种信息泄漏。例如两个不同安全级的进程

共享一块有限大小的存储资源，这块存储资源既可能构成一个隐蔽存储通道，也可能构成一个隐蔽时间通道。因此，TCSEC 要求，B2级以上安全的操作系统必须有隐蔽通道分析[8]。

访问控制模型，特别是 BLP 模型，是一类最实用的安全模型[20]。但随着信息处理系统越来越复杂，不论军事部门还是商业部门，都需要更加灵活的安全策略，需要更加细致地划分控制粒度，需要更加灵敏的信息流控制；而新的软件范型，如面向对象、超文本、虚拟机、移动代码、智能主体等，却引入了大粒度的实体；还有一些更加复杂的环境，很难区分可以独立分级标记的实体。因此，模仿"公文世界"的访问控制模型已经不能适应新的安全需求。现在已经明确，安全问题可归结为信息流的无干扰性质，而无干扰性质又可归结为系统的等价性问题[19]，后者是计算机科学中的中心问题之一。

参 考 文 献

1 Anderson J P. Computer Security Technology Planning Study Volume I:[ESD-TR-73-51]. Vol. I, Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Bedford, MA, USA, Oct. 1972

2 Anderson J P. Computer Security Technology Planning Study Volume Ⅱ: [ESD-TR-73-51]. Vol Ⅱ, Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Bedford, MA, USA, Oct. 1972

3 Bell D E, LaPadula L J. Secure Computer Systems: Mathematical Foundations: [ESD-TR-73-278]. Vol. I, AD 770 768, Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Bedford, MA, USA, Nov. 1973

4 Bell D E, LaPadula L J. Secure Computer Systems: A Mathematical Model: [ESD-TR-73-278]. Vol. Ⅱ, AD 771 543, Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Bedford, MA, USA, Nov. 1973

5 Bell D E, LaPadula L J. Secure Computer System: Unified Exposition and MULTICS Interpretation: [Technical Report ESD-TR-75-306]. MTR-2997 Rev. 1, The MITRE Corporation, Bedford, MA, USA, Mar. 1976

6 Castano S, Fugini M G, Martella G, Samarati P. Database Security, Addison-Wesley Publicating Company, 1995

7 Denning D E. A Lattice Model of Secure Information Flow. Communications of the ACM, 1976, 19(5): 236～243

8 Department of Defense: DOD Trusted Computer Security System Evaluation Criteria (The Orange Book), DOD 5200. 28-STD, 1985

9 Goguen J A, Meseguer J. Security policies and security models. In: Proc. of the 1982 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, April, 1982. 11～20

10 Goguen J A, Meseguer J. Unwinding and Inference Control. In: Proc. of the 1984 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, May 1984. 75～86

11 Harrison M A, Ruzzo W L, Ullman J D. Protection in Operating Systems Communications of the ACM, 1976, 19(8): 461～471

12 Lampson B W. Protection. ACM Operating Systems Reviews, 1974, 8(1): 18～24

13 Lampson B W. A Note on the Confinement Problem. Communications of the ACM, 1973, 16(10): 613～615

14 Landwehr C B. Formal Models for Computer Security. ACM Computing Surveys, 1981, 13(3): 247～278

15 McLean J. A Comment on the 'Basic Security Theorem' of Bell and LaPadula. Information Processing Letters, 1985, 20(2): 67～70

16 McLean J. Security models. in the Encyclopedia of Software Engineering (ed. John Marciniak), Wiley & Sons, Inc., 1994

17 Roscoe A W. CSP and determinism in security modelling. In: Proc. of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, 1995. 114～127

18 Roscoe A W. The theory and practice of concurrency. Prentice-Hall 1997

19 Ryan P Y A, Schneider S A. Process Algebra and Non-interference. Journal of Computer Security, 2001, 9(1-2): 75～103

20 Tanenbaum A S. Moden Operating Systems (second edition). China Machine Press, 2002

(上接第182页)

具[10]，它实现了图形用户界面、支持远程分布调试、支持多种断点以及提供了强大的变量跟踪和显示支持。但是 TotalView 的可移植性较差，而且是一个商业产品，使用起来的代价是一个不得不考虑的问题。清华大学实现了一个可移植的并行源级调试工具 Buster[11]，它支持多进程程序的源级交互调试，实现了图形用户界面，并能支持多种典型的工作站机群系统，但 Buster 不能支持远程调试。国防科学技术大学实现了一个远程并行调试工具 Pdebug[12]，但是他们采用的是自行设计的通信协议，功能有限，而且可移植性较差。

小结 并行调试对并行程序的开发非常重要，然而过去并行调试的主要方式是采用文本的命令行界面，远程用户一般通过 telnet 命令模拟终端来进行并行调试。

本文介绍了我们设计实现的远程并行调试器 RPB 的工作。RPB 实现了完全并行调试等强大的并行调试功能。同时 RPB 提供了友好的图形用户界面，并且使用 Java 语言和 Swing 图形工具包实现，具有良好的可移植性。RPB 通过 CORBA 技术实现了用户的远程调试，所有通信细节对用户完全透明，并且 RPB 的客户端嵌入到 Web 浏览器中，用户无需安装任何客户端程序，随时随地通过任何一个支持 Java 的浏览器就可以使用，极大地方便了用户使用并行调试工具的方式，使用户感觉不到平台的差异和并行机地理位置的差异，大大提高了并行机的好用性。RPB 的实现也较好地验证了为超级计算机在客户端提供支持远程使用的图形用户界面的可行性，让我们看到了未来超级计算机的使用方式将在新技术的支持下发生根本的变化。

参 考 文 献

1 吴明，陈国良，孙凝晖. 并行计算机用户环境的设计与实现. 计算机学报，2000，23(10)：1021～1027

2 安虹，陈国良，李宏，陈志辉. 曙光并行机客户端集成环境的设计. 计算机研究与发展，2001，38(增刊)：181～186

3 Lumetta S S, Culler D E. The Mantis Parallel Debugger. In: Proc. of the First Symposium on Parallel and Distributed Tools. Philadelphia, PA, 1996. 118～126

4 Wang Feng, Zheng Qilong, An Hong, Chen Guoliang. A Parallel and Distributed Debugger Implemented with Java. In: Proc. of the 31th Intl. Conf. on Technology of Object Oriented Languages and Systems (TOOLS Asia'99). IEEE Computer Society Press, 1999. 342～348

5 Netzer R H B, Miller B P. Optimal Tracing and Replay for Debugging Message-Passing Parallel Programs. In: Proc. of Supercomputing 92. Minneapolis, MN, 1992. 502～511

6 Wang Feng, An Hong, Chen Zhihui, Chen Guoliang. Completely Debugging Indeterminate MPI/PVM Programs. 软件学报，2001，12(3)

7 Sun Microsystems. Java Language—A White Paper. Sun Microsystems Computer Company, 1996

8 Geary D M 著，李建森等译. Java 2 图形设计卷Ⅰ：SWING. 北京：机械工业出版社，2000

9 Otte R, Patrick P, Roy M 著，李师贤等译. CORBA 教程：公共对象请求代理体系结构. 北京：清华大学出版社，1999

10 TotalView: http://www.etnus.com/

11 杜术，陈晓鹏，汪东升，郑纬民. Buster：一个可移植的并行调试器. 小型微型计算机系统，2001，22(10)

12 黄瑞芳，朱敏，张卫民. 远程调试的设计与实现. 计算机工程与应用，2001，27(1)