9.3.6 安全系统的形式化模型

诸如图9-5的保护矩阵并不是静态的。它们通常随着创建新的对象，销毁旧的对象而改变，而且所有者决定对象的用户集的增加或限制。人们把大量的精力花费在建立安全系统模型，这种模型中的保护矩阵处于不断的变化之中。在本节的稍后部分，我们将简单介绍这方面的工作原理。

几十年前，Harrison等人（1976）在保护矩阵上确定了6种最基本的操作，这些操作可用于任何安全系统模型的基准。这些最基本的操作是create object，delete object，create domain，delete domain，insert right和remove right。最后的两种插入和删除权限操作来自于特定的矩阵单元，如赋予域1读文件6的许可权。

上述6种操作可以合并为保护命令。用户程序可以运行这些命令来改变保护矩阵。它们不可以直接执行最原始的操作。例如，系统可能有一个创建新文件的命令，该命令首先查看该文件是否已存在，如果不存在就创建新的对象并赋予所有者相应的权限。当然也可能有一个命令允许所有者赋予系统中所有用户读取该文件的权限。实际上，只要把“读”权限插入到每个域中该文件的登录项即可。

此刻，保护矩阵决定了在任何域中的一个进程可以执行哪些操作，而不是被授权执行哪些操作。矩阵是由系统来强制的；而授权与管理策略有关。为了说明其差别，我们看一看图9-12域与用户相对应的例子。在图9-12a中，我们看到了既定的保护策略：Henry可以读写 mailbox7，Robert可以读写secret，所有的用户可以读和运行compiler。

对象

	Compiler	Mailbox 7	Secret
Eric	读/运行		
Henry	读/运行	读/写	
Robert	读/运行		读/写

a)

对象

	Compiler	Mailbox 7	Secret
Eric	读/运行		
Henry	读/运行	读/写	
Robert	读/运行	读	读/写

b)

图9-12 a) 授权后的状态；b) 未授权的状态

现在假设Robert非常聪明，并找到了一种方法发出命令把保护矩阵改为如图9-12b所示。现在他就可以访问mailbox7了，这是他本来未被授权的。如果他想读文件，操作系统就可以执行他的请求，因为操作系统并不知道图9-12b的状态是未被授权的。

很明显，所有可能的矩阵被划分为两个独立的集合：所有处于授权状态的集合和所有未授权的集合。经过大量理论上的研究后会有这样一个问题：给定一个最原始的授权状态和命令集，是否能证明系统永远不能达到未授权的状态？

实际上，我们是在询问可行的安全机制（保护命令）是否足以强制某些安全策略。给定了这些安全策略、最初的矩阵状态和改变这些矩阵的命令集，我们希望可以找到建立安全系统的方法。这样的证明过程是非常困难的：许多一般用途的系统在理论上是不安全的。Harrison等人（1976）曾经证明在一个不定的保护系统的不定配置中，其安全性从理论上来说是不确定的。但是对特定系统来说，有可能证明系统可以从授权状态转移到未授权状态。要获得更多的信息请看Landwehr（1981）。

9.3.7 多级安全

大多数操作系统允许个人用户来决定谁可以读写他们的文件和其他对象。这一策略称为可自由支配的访问控制（discretionary access control）。在许多环境下，这种模式工作很稳定，但也有些环境需要更高级的安全，如军方、企业专利部门和医院。在这类环境里，机构定义了有关谁可以看什么的规则，这些规则是不能被士兵、律师或医生改变的，至少没有老板的许可是不允许的。这类环境需要强制性的访问控制（mandatory access control）来确保所阐明的安全策略被系统强制执行，而不是可自由支配的访问控制。这些强制性的访问控制管理整个信息流，确保不会泄漏那些不应该泄漏的信息。

1. Bell-La Padula模型

最广泛使用的多级安全模型是Bell-La Padula模型，我们将看看它是如何工作的（Bell La和Padula，