中国科学 E 辑: 信息科学
2007 年 第 37 卷 第 2 期: 238~253
http://www.scichina.com

《中国科学》杂志社
SCIENCE IN CHINA PRESS

高等级安全操作系统的设计

卿斯汉*

(中国科学院软件研究所, 北京 100080; 北京大学软件与微电子学院, 北京 102600)

沈昌祥

(海军计算技术研究所, 北京 100841)

摘要 众多因特网安全事件的发生表明, 为了对抗现代计算环境中的安全威胁, 来自安全操作系统的支持是必不可少的. 基于国内外相关标准的要求, 结合安胜高等级安全操作系统 v4.0 (以下简称为安胜 OS)的设计与开发实践, 讨论高等级安全操作系统设计中的 3 个关键问题: 安全体系结构、安全模型与隐蔽通道分析. 对安全体系结构与 3 种基本的安全策略模型: 机密性模型、完整性模型和特权控制模型的设计原则分别进行了阐述, 提出了新的安全体系结构与 3 个新的安全模型, 分别介绍它们的主要特色, 以及它们在安胜 OS 中的实现.

隐蔽通道分析是高等级安全操作系统设计中众所周知的难题, 迄今缺乏坚实的理论基础与系统的分析方法. 为了解决隐蔽通道分析中存在的基本问题, 文中提出了隐蔽通道标识完备性的理论基础、一种通用的隐蔽通道标识框架, 以及高效的回溯搜索方法. 这些新方法在安胜高等级安全操作系统中的成功实现表明, 它们可以简化并加快整个隐蔽通道的分析过程.

关键词 高等级安全操作系统 体系结构 安全模型 隐蔽通道分析

一般认为, 符合国标 GB17859 第 3 级(相当于橘皮书 TCSEC B1 级和 CC 标准 EAL4 级)以上的操作系统是安全操作系统. 再高一级, 亦即符合国标 GB17859 第 4 级(相当于 TCSEC B2 级和 CC EAL5 级)以上的操作系统是高等级安全操作系统. 安胜 GB17859 第 4 级安全操作系统 v4.0, 以下简称安胜 OS, 是中国科学院信息安全技术工程研究中心自主研制的, 其总体结构如图 1 所示.

安胜 OS 基于 Linux 内核, 设计与开发时共分析了内核源程序 38 万行, 改造和开发了 4.5

收稿日期: 2006-07-14; 接受日期: 2006-09-29
北京市自然科学基金(批准号: 4052016)、国家自然科学基金(批准号: 60573042)和国家重点基础研究发展规划(批准号: G1999035802)资助项目
* 联系人, E-mail: qsihan@ercist.iscas.ac.cn

万余行. 对系统原有的 221 个系统调用逐次进行了安全性分析, 其中, 未做任何修改的 49 个, 加入审计机制的 130 个, 加入存取控制机制的 95 个, 新开发系统调用 33 个. 新开发用户 shell 命令 69 个, 修改用户 shell 命令 16 个. 在隐蔽通道分析中, 分析了全部内核源代码与系统调用, 以及 365 个全局变量和 75 个可信进程. 技术报告与文档资料超过 150 万字.

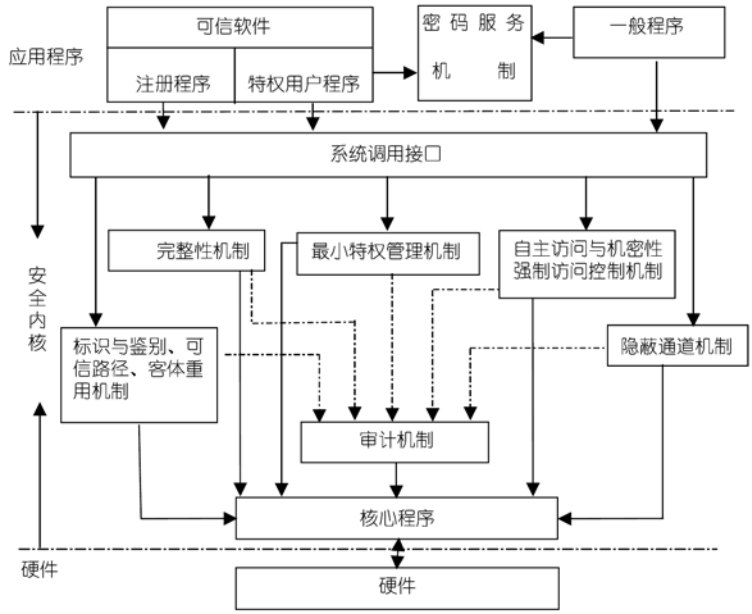

图 1　安胜 OS 的总体结构图

本文的其余部分组织如下: 第 1 节讨论安全体系结构的设计原则, 提出新的架构, 说明它的特点, 以及在安胜 OS 中的实现. 第 2 节讨论安全策略模型, 包括机密性模型、完整性模型与特权控制模型. 分别讨论 3 种重要安全模型的设计原则, 提出新的模型, 并介绍它们的特色, 以及在安胜 OS 中的实现. 第 3 节详尽地讨论隐蔽通道分析, 提出新的理论基础, 通用的标识架构和新的回溯搜索方法, 并介绍它们在安胜 OS 中的实现. 最后, 第 4 节总结全文.

1　安全体系结构

安全体系结构是高等级安全操作系统的基础, 它的设计可以借鉴著名的Flask体系 [1], 它具有当代操作系统支持多策略与动态策略的特征. 设计时我们重点考虑的是, 如何将基于微内核的Flask 体系集成到基于宏内核的Linux之中. 在Linux系统中, LSM[2]是其内核中一个轻量级通用访问控制框架, 可使不同的安全模型以Linux 可加载内核模块的形式实现. 由此, 可以提高Linux访问控制机制的灵活性, 这正是现代安全操作系统追求的目标. 有鉴于此, 安胜OS结合Flask体系与LSM框架, 设计了基于Flask的模块化安全内核, 如图 2 所示. 安全内核可视为由策略引擎与策略实施两部分组成, 其中策略引擎封装在安全服务器之中, 以安全模块的形式出现. 策略实施则由对象管理器具体执行, 对象管理器包括文件安全管理器、网络安全管理器、进程安全管理器、审计管理器等.

Flask 体系虽然支持策略删除, 但是不支持策略增加. 它仅给出了策略删除的抽象原则, 没有给出可操作的规范. 安胜 OS 设计的形式架构做了相应的改进, 并考虑到解决策略等价、

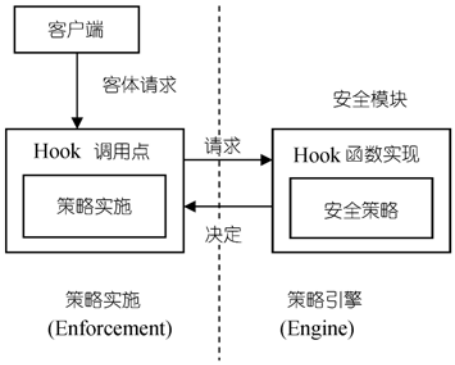

图 2 Flask 与 LSM 的结合

策略冲突消解与策略协作等问题.

在这种架构下, 安全服务器作为安全内核的子系统, 可以实现对安全策略的封装, 并提供通用接口. 通过安全服务器, 可以实现多种安全策略的并存, 从而实现支持动态多策略的整体安全体系结构, 满足高等级安全操作系统的需求.

该架构由模型集合 $\{M_1, M_2,\cdots, M_n\}$、模型接口集合 $\{I_1, I_2,\cdots, I_n\}$ 和模型组合器组成, 如图 3 所示. 因为各种安全策略模型的决策方式可能不同, 所以为它们设计了统一的模型接口. 请求通过接口转送给安全模型, 实际模型做出决策后再返回接口, 作为接口的决策付诸实施.

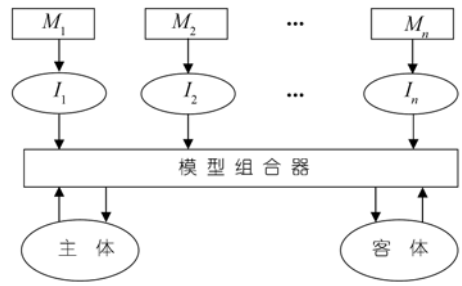

图 3 支持多策略的形式架构

下面进行形式化的定义: 对任意模型 M_i, 令 $V_i=\{v_{ij}\mid j=1,2,\cdots\}$ 表示它的状态空间: R 表示请求: D 表示模型接口的决策集. 定义 M_i 的模型接口为函数 $I_i: V_i\times R\to D$.

如图 4 所示, 模型组合器由冲突关系 Φ, 冲突消解关系 Ψ 与模型协作关系 Σ 组成, 其中, Ω_i 表示冲突类的集合.

称 $\{M_{i_1}, M_{i_2},\vdots, M_{i_j}\}\in\Phi$, 如果我们可以从这些模型的不变量推导出矛盾的不变量, 并且缺少其中任何一个都不能导出这些矛盾的不变量, 因此由定义有: 如果 p 与 $\neg p$ 是属于 $\{M_{i_1}, M_{i_2}\}$ 的矛盾不变量, 则对任意不同于 M_{i_1} 与 M_{i_2} 的模型 M_{i_j}, p 与 $\neg p$ 不是属于 $\{M_{i_1}, M_{i_2}, M_{i_j}\}$ 的矛盾不变量. 并且, 如果 p 与 $\neg p$ 是属于 $\{M_{i_1}, M_{i_2}, M_{i_j}\}$ 的矛盾不变量, 且它们之中有两个模型, 例如 M_{i_1} 与 M_{i_2}, 使得 p 与 $\neg p$ 属于 $\{M_{i_1}, M_{i_2}\}$, 则 $\{M_{i_1}, M_{i_2}, M_{i_j}\}\notin\Phi$.

令 $\{\Omega_1,\Omega_2,\cdots,\Omega_s\}\in\Phi$, 且 $\bigcup_{j=1,\cdots,s}\Omega_j=\{M_1, M_2,\cdots, M_n\}$, 对任意冲突类 $\Omega_i:=\{M_{i_1}, M_{i_2},\cdots, M_{i_j}\}$, 定义冲突消解关系为函数 $\Psi_i: I_{i_1}\times I_{i_2}\times\cdots\times I_{i_j}\to D$. 称 $(\Psi_1,\Psi_2,\cdots,\Psi_s)$ 为系统的冲突消解关系. 协作关系定义为函数 $\Sigma:\Psi(\Omega_1)\times\Psi(\Omega_2)\times\cdots\times\Psi(\Omega_s)\to D$.

为了定义冲突消解关系, 必须求出所有的矛盾不变量. 因此, 在定义分量模型时, 必须充分考虑分量模型可能蕴涵的不变量. 显然, 文献 [3]中提到的冲突矩阵只是一种特殊的消解关系, 不能代表一般情形的 Ψ.

与相关的工作相比较, 上述架构具有以下特点: (1) 与Flask体系兼容, 并改进了Flask体系的局限性, 即新的架构既支持形式化的策略增加与删减, 又给出了策略冲突消解与策略协作的机制. 此外, 通过Flask体系与LSM的结合, 新结构适用于基于Linux的安全操作系统. (2) 新架构的结构是模块化的, 模块之间具有明确的关系. (3) 推广了Kuhnhauser与Ostrowski[3]提

出的策略冲突矩阵方法. 他们讨论了在多策略环境下支持安全策略的形式化架构的一般概念, 但是他们提出的架构过于复杂和一般化, 因此不适于在安全操作系统中实现. (4) Bell[4]也提出了一种支持策略合作与策略冲突的概念性架构, 但是他提出的多策略机制不适于在安全操作系统中分析多安全策略. 新架构视协作为具有不同安全属性的策略之间的关系: 冲突是相同安全属性的不同安全策略之间的关系. 因此, 我们处理的是冲突类之间的协作, 而不是策略之间的协作, 从而清楚地界定了冲突分解与协作的不同.

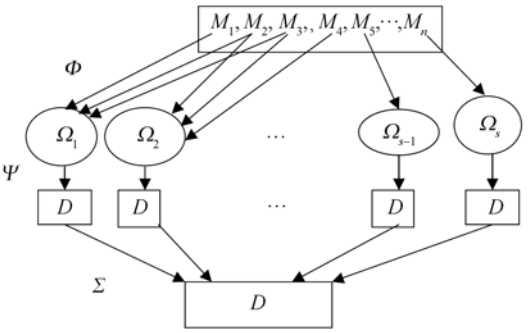

图 4　模型组合器的结构

2　安全策略模型

对于高等级安全操作系统, 标准要求提供形式化的安全策略模型. 安胜 OS 提出了机密性、完整性和特权控制 3 种新的形式化安全策略模型, 并在前一节所述的形式构架中实现.

2.1　机密性模型

2.1.1　背景与相关工作

在详尽讨论我们提出的新模型之前, 首先简要地介绍前人的相关工作. 典型的机密性模型BLP (Bell-LaPadula) [5]对多级安全系统中的每个主体和客体都赋予一个安全级. 主体的安全级规范了主体允许访问的数据的最高安全级, 而客体的安全级表示在客体中存储的数据的最高安全级. 此外, 在BLP模型中可以定义可信主体, 即相信可信主体可以称职地对任何数据建立适当的安全级并进行适当的处理.

BLP模型的主要缺点在于, (1) 可信主体具有无限的特权; (2) 模型的灵活性不够, 并且过于依赖平稳性(tranquility)原则, 主体与客体的安全级在其生命周期内不变. 为此, Ott[6]提出一种可以动态调节当前安全级的机制, 增加了BLP模型的灵活性, 但付出了增加占用资源的代价. Lee[7]引入了部分可信主体的概念, 试图在一定程度上限制可信主体的特权. Bell[8]将主体的当前安全级扩展为二元对 $(a\text{-min}, v\text{-max})$, 称为安全级范围. Mayer[9] 在TMach系统中进一步推广了这种思想, 并将 a-min 和 v-max 不相等的主体定义为可信主体.

2.1.2　新的模型

在我们的模型中, 为主体分配了 3 个安全标签: 最大安全级标签 f_s, 最小可写安全标签 $a\text{-min}_s$ 和最大可读安全标签 $v\text{-max}_s$. 称 $[a\text{-min}_s, v\text{-max}_s]$ 为主体的可写范围, 并记为 ran_s. 为客体分配了一个最大安全级标签 $L\text{-max}_o$ 和一个最小安全级标签 $L\text{-min}_o$. 称 $[L\text{-min}_o, L\text{-max}_o]$ 为客体的安全标签范围, 并记为 ran_o. 此外, 模型中定义了单级客体与多级客体. 注意, 我们定义的多级客体不同于Fluke微内核 [10]中实现的多级客体. 在那里, 多级客体定义为 $L\text{-min}_o \neq L\text{-max}_o$, 只能由可信主体访问.

新模型的顶层安全策略是: (1) 只有可信主体才能对客体进行具有多级属性的访问: 一般主体只能对客体进行具有单级属性的访问. (2) 可信主体对客体进行具有单级属性的访问时, 只能

在安全标签范围内是可信的. (3) 每个客体都分离为两部分: 与内容相关和与内容无关的部分, 这两部分作为不同的客体具有相同的安全标签范围. (4) 对可信客体的创建及删除是具有多级属性的访问, 而对它们的应用性访问是具有单级属性的访问.

进程可以通过任意一种进程间通信 (IPC) 的方法彼此通信. 在操作系统中, IPC提供的功能可以帮助信息之间的共享与交换, 以及对共享资源访问的同步, 因此, IPC的重要性不言而喻. 但是, 如果对IPC对象处理不当, 不仅会降低效率, 而且会产生隐蔽通道. 新模型对IPC保护的顶层策略是, 通过IPC对象所属的进程控制IPC对象. 新模型通过两条规则 [11]实现: (1) client进程对server进程的操作只能是单向的. (2) 仅允许下述组合: 非可信client进程与非可信server进程: 非可信client进程与可信server进程: 可信client进程与可信server进程. 并且, 严格禁止非可信进程为可信进程提供服务. (3)通过主动进程与IPC对象一体化, 同时解决了多级应用与隐蔽通道的问题.

新模型包含 6 个不变量、6 个约束条件、14 条状态迁移规则和基本安全定理, 其形式化建模与分析的细节, 请参看文献 [11].

新模型在安胜 OS 中的实现包括两个主要方面. 首先, 安全标签赋值的实现. 对任意主体, 每个进程被赋予一个当前安全标签. 用户安全标签由系统管理员创建用户时通过 useradd 命令设置, 标明用户的安全标签范围及缺省安全标签. 进程安全标签在创建进程时确定, 在它所代表的用户安全标签范围之内. 对于客体, 每个客体被赋予安全标签范围与当前安全标签.

文件、特别文件(表示设备驱动程序)、管道的安全标签为创建该客体进程时的安全标签, 且客体的安全标签等于其父目录的安全标签. 目录与普通文件一样, 在它们的生存周期内具有安全标签. 不同之处在于目录结构满足兼容性, 即进程创建目录时, 目录的安全标签就是它的创建进程的安全标签, 且目录的安全标签不小于其父目录的安全标签. 进程、消息队列、信号量集合和共享存储区是特殊类型的客体, 当 fork(), msgget(), semget(), shmget()系统调用创建这类客体时, 客体的安全标签就是它的创建进程的安全标签.

其次, 系统安全级设置: 安胜 OS 通过安全级将系统划分为 3 个区, 即系统管理区、用户空间区和病毒保护区, 如图 5 所示. 该图可视为一组系统安全标签与用户安全标签, 它们通过模型的控制机制分隔. 其中, “箭头”表示安全级支配关系. 对于系统管理区, 用户没有读写任何数据的权限, 如 TCB 数据、审计数据等. 在用户空间区, 一般用户具有读与写的权限. 对于病毒保护区中的数据与文件, 用户进程具有读权限, 但没有写权限. 因此, 这种访问隔离机制将进入系统的用户分为两类: 不具有特权的普通用户与系统管理用户. 前者在用户工作区中登录, 如 USER_LOGIN, USER_PUBLIC 等: 后者则在系统管理区中登录, 如 SYS_AUDIT, SYS_OPERATOR1, SYS_OPERATOR2, SYS_PRIVATE 等.

2.1.3 新模型的分析、比较与特点

新模型是可以动态调节安全标签范围的形式化机密性模型, 其特点是: (1) 类似于Ott模型 [6], 支持安全级范围的动态变更. 但是, 新模型提出了具有单级与多级属性的操作, 对可信主体进行了形式化规范, 使可信主体是真正部分可信的. (2) 可信主体定义为可信操作有限、常规操作之后主体写范围 ran_s 不变的主体, 与文献中各种可信主体的定义都不同. 可信主体不仅可以执行一般主体的单级访问, 也可以执行为它们专门设计的多级访问. 特别, 可信主体的写范围 ran_s 可以由安全策略进行管理. (3) 新模型首次通过信息流控制的安全策略, 形式化地

规范可信主体的行为. 模型中证明的定 理[11]表明, 如果满足新模型的不变量与约束条件, 不会发生信息由高安全级到低安全级的非法流动. (4) 在Fluke[10]的实现中没有仔细考虑兼容性的问题, 新模型中引入的兼容属性 [11]可以有效地配合 a-min$_s$ 和 v-max$_s$ 的动态变化. (5) 新模型能有效地处理网络安全对象[11]. (6) 新模型实现主动进程与IPC对象一体化, 同时解决了多级应用与隐蔽通道的问题.

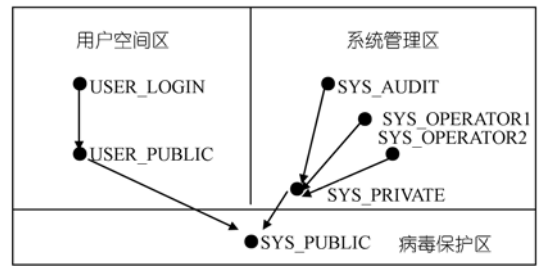

图 5　安胜 OS 系统安全级设置

值得指出的是, 在后续研究中我们发现, 最初提出的机密性模型 [11], 其安全性证明是不完备的. 定义 2.1[11]没有明确指出非可信主体访问 $L\text{-min}_o \neq L\text{-max}_o$ 的单级客体时, 如何确定客体安全级范围内的安全级. 同时在模型约束规则 CT_1 中, 访问监控器仲裁非可信主体对 $L\text{-min}_o \neq L\text{-max}_o$ 的单级客体访问时, 没有考虑客体安全级范围内的安全级, 而是通过单级客体的最大与最小安全级确定主体对客体的访问权限, 这样, 会导致违反安全策略的信息流. 因为虽然可以证明, 非可信主体访问 $L\text{-min}_o = L\text{-max}_o$ 的客体时不会发生违反安全策略的信息流. 但是, 我们无法证明非可信主体访问 $L\text{-min}_o \neq L\text{-max}_o$ 的客体时, 不会出现违反安全策略的信息流. 限于本文的篇幅, 改进后的机密性模型将另文发表.

2.2　完整性模型

2.2.1　背景与相关工作

最早的完整性模型是Biba[12]于 1977 年首先提出的, 可以视为一种基于安全级(level based)的完整性模型. Biba模型的缺点主要是: (1) 完整性标签很难确定. (2) Biba模型没有提供保证数据一致性的机制. (3) Biba模型难以在实际系统中实现.

Clark-Wilson模型 [13]是一种基于封装(encapsulation-based)的完整性模型. 该模型具有里程碑的意义, 涉及计算机系统完整性的所有目标: 用户完整性、数据完整性和过程完整性. 并且, 引入了良构事务(well-formed transaction)的重要概念.

Clark-Wilson模型的局限性在于: (1) 没有形式化, 作为通用模型也难以形式化. (2) 转换过程之间的顺序执行关系不利于从数据项中分离对数据的控制策略. (3) Karger[14]指出, 在实际系统中难以高效与灵活地实现.

域与型实施模型DTE[15]提供一种访问控制机制, 其中一个称之为“域(domain)”的访问控制属性与主体绑定, 另一个称之为“型(type)”的安全属性与客体绑定. 当系统运行时, 可以动态地调节访问请求, 并拒绝未经授权的访问. 此外, DTE是一种灵活的、可配置的、内核级的访问控制机制, 支持最小特权原则, 并且适于进行安全策略的配置. 著名的SELinux[16]当前是一个安全增强的Linux原型系统, 它在安全策略配置时定义了一系列DTE域与型. 但是, 迄今未见在安全系统中实现基于DTE的形式化完整性模型.

总之, 与机密性模型相比, 迄今完整性模型在成熟度、形式化与应用等方面还相距甚远.

2.2.2　新的模型

新的形式化完整性模型基于 DTE, 它应用到以下 4 个概念: (1) 域: 可以视为从逻辑上界

定的进程的受限执行环境. 进程进入一个域之后, 其活动就限定在这个域内. 域包含 3 个重要的组成部分. 第 1 是入口守护进程, 构成进入域的惟一通道; 第 2 是域的管理空间与权限; 第 3 是域的关系域列表及其相关的关系. (2) 型: 可以视为受完整性策略保护的数据的一种特殊类型名. (3) 完整性校验过程 IVP: 在主体操作受保护数据之前, 校验该主体的身份与访问权限的过程. (4) 良构事务 WFT.

新模型的顶层安全策略是: (1) 定义与应用管理用户域和非管理用户域. (2) 根据完整性控制粒度, 定义数据类型. (3) 域间必要的信息流动必须通过 WFT 实现, 保证域间信息流动的完整性. (4) 将 WFT 标识为特殊的数据类型, 防止非法的使用与篡改. 对 WFT 的应用进行审计, 防止授权用户的不恰当修改. (5) 将 IVP 标识为特殊的数据模型, 对创建和操作完整性保护数据的行为实现 IVP. (6) 仔细设计对域-域关系表 DDT 和域-型关系表 DTT 的维护, 保证配置的正确性.

新模型有两个主要的组成部分: 保障规则与状态迁移模型. 保障规则的目的是指出如何设置域与型才能实现安全不变量, 采用什么保证措施才能实现完整性保护目标. 状态迁移模型是一种特殊的状态机模型, 负责定义系统状态、状态迁移规则和安全不变量. 在新模型中, 设置了 10 个不变量, 包括新提出的处理信息流的不变量: 提出了 13 个具有原子性的迁移规则. 此外, 提出并证明了一个基本安全定理. 关于其他细节, 可参看文献 [17].

在安胜 OS 中, 基本的域与型分别有 6 个和 27 个. 根据模型不变量, 为域与型给出了非控制关系、非依赖关系、可信管道与角色等, 并验证这些设置满足不变量. 例如, 当进程试图访问文件时, 内核将进行完整性校验. 当域中的进程访问型的对象时, 访问模式包括: 文件读(r)、文件写(w)、文件执行(x)、文件附加(a)、目录读(l)、目录写(c)和目录执行(d).

系统的每个 inode 节点包含 3 个指针, 分别表示 etype 值, rtype 值和 utype 值. etype 表示一个目录或文件的型. rtype 表示一个目录下(递归地)所有文件或子目录的型(包括这个目录本身). utype 表示一个目录下(递归地)所有文件或子目录的型, 但不包括这个目录.

当执行域的入口点文件时, 共有 3 种可能的域转化: 自动域转化、自愿域转化和空的域转化, 亦即域保持不变, 如图 6 所示.

2.2.3 新模型的分析、比较与特点

新模型是首次提出的基于 DTE 的形式化完整性模型, 其目标是通过域隔离最大限度地实现数据完整性与系统完整性.

新模型具有以下特点: (1) 结合 DTE 模型与良构事务的概念, 实现域与域之间的受限信息交换. (2) 仅将良构事务分配给角色, 与相关的主体无关, 比 Clark-Wilson 模型的解决方案更为有效. (3) 通过增加策略执行的保障规则实现多域策略, 确保受保护数据的内在一致性, 从而改进了 Biba 模型.

2.3 特权控制模型

2.3.1 背景与相关工作

最小特权设计原则是Saltzer[18]首先提出的, 要求对系统中的每一个进程只赋予完成任务所需的最小特权. 基于角色的访问控制模型RBAC[19]是实施最小特权的主流模型, 但是RBAC的策略具有较高的抽象层次, 它的控制粒度较粗且灵活性不够, 难以在安全操作系统中实现基于内容的最小特权控制.

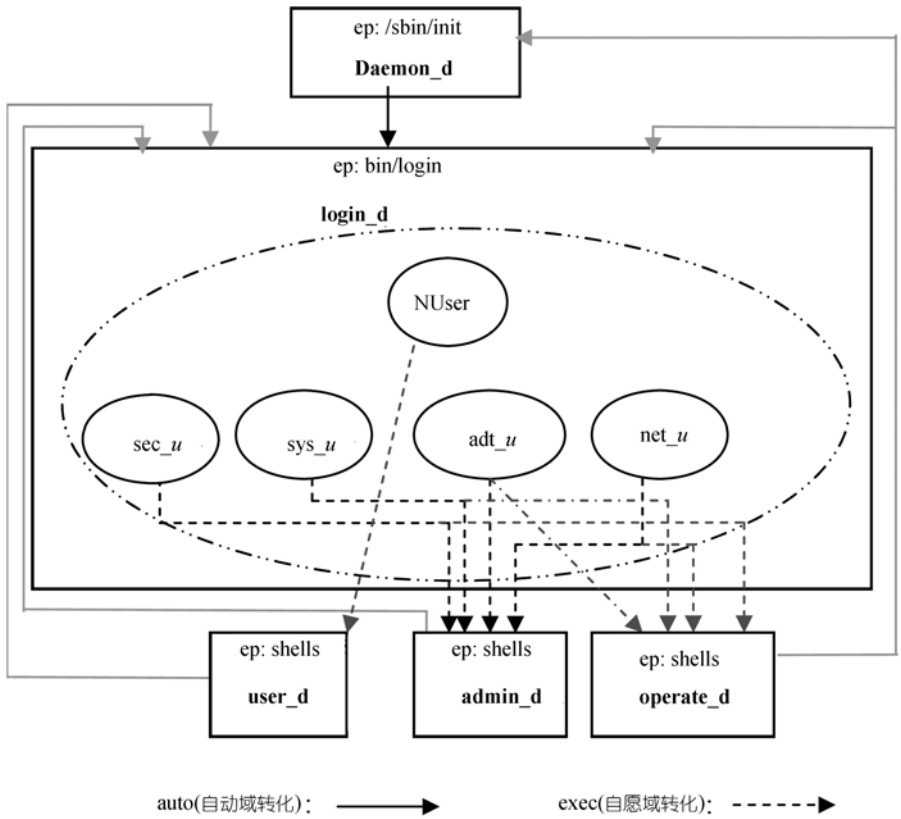

图 6　安胜 OS 中完整性策略的域关系图举例

在安全操作系统Lock6 的实现中，Hoffman[20]首次提出结合RBAC与DTE的思想．但是，Hoffman的实现是粗粒度的，没有细化到系统中的可操作事件(operational events)与超越访问事件(access override events)．此外，没有给出特权控制机制．Chandramouli[21]也提出一种结合RBAC与DTE的架构，但是这种架构适用于应用系统而不是操作系统，因此仅关心多授权类型而不是执行与安全相关的操作时所需的特权．SELinux[16]的安全模型结合RBAC，TE(Type Enforcement)与用户身份(User Identity)模型，但是仅通过TE控制一般访问模式，并没有应用POSIX 标准 [22]实施权能控制．尽管POSIX权能机制可以在细粒度下实现最小特权原理，但它只定义了权能机制的接口，并没有规范最小特权机制，也没有给出权能遗传算法．

2.3.2　新的模型

新模型是一种结合 RBAC, DTE 和 POSIX 权能机制的形式化特权控制模型，它具有实现最小特权原理所需的由 3 个层次组成的结构．其中，顶层称为管理层，用于控制主体所能拥有的最大特权，从管理角度限制进程的活动范围．中层称为功能控制层，用于控制实现某些特定功能所需的功能特权，从功能实现的角度实现功能隔离．底层称为执行层，用于控制进程完成一项任务时所需的进程特权，从完成任务的角度限制进程的活动能力．

新模型实现了下述目标：(1) 静态角色职责隔离(SRSD)：如果用户被指派角色 r，且角色 r' 与角色 r 为授权互斥关系，则不能将角色 r' 指派给该用户．(2) 动态角色职责隔离(DRSD):

如果主体进程以用户允许的角色 r 运行，角色 r' 与 r 为运行互斥关系，则不能存在以同一用户且角色为 r' 运行的主体进程. (3) 动态域职责隔离(DDSD): 如果用户通过一个角色创建的主体进程运行于域 d 中，且域 d' 与 d 为运行互斥关系，则不能存在具有相同用户与角色，但运行于域 d' 的主体进程.

考虑到下述两个因素: (1) DTE 域可以限制进程的执行范围，因此扩展它可以限制权能的生效范围. (2) 单纯依靠主体标识符标识主体的权能集难以实现职责隔离. 因此，新模型中引入了两个新的权能集，一个称为域权能集，记为 B_d，由域标识符惟一地确定；另一个称为角色权能集，记为 B_r，由角色标识符惟一地确定，与主体标识符无关. 在此基础上，新模型给出下述新的权能遗传算法:

$$I_1 = I_0 \wedge I_f,$$
$$P_1 = (P_f \vee (I_1 \wedge P_0)) \wedge B_r \wedge B_d,$$
$$E_1 = P_1 \wedge E_f.$$

新模型设置了 14 个不变量，提出相应的约束条件与迁移规则，并证明了基本安全定理. 更多的细节，可参看文献 [23].

在新模型的安胜 OS 实现中，初始的授权实体为: 3 个管理用户、4 个可信角色和 2 个可信域. 超级用户的权限分解为 57 个特权. 安全管理用户 sec_*u* 负责安全策略数据库的配置与管理，被赋予安全员角色 sec_*r*. 系统管理用户 sys_*u* 负责系统与网络的配置与管理，被赋予系统员角色 sys_*r* 和网络员角色 net_*r*. 审计管理用户 adt_*u* 负责审计事件与审计记录的配置与管理，被赋予审计员角色 adt_*r*. 代表用户的不同角色进入不同的 DTE 域就具有不同的特权，因此，事实上在初始设置时定义了 8 个执行不同任务的角色.

安全员角色 sec_*r* 与审计员角色 adt_*r* 分别负责安全策略数据库与审计记录的管理，因此应当实施静态职责隔离，如表 1 中的 SRSD 所示. 系统管理员用户 sys_*u* 允许承担系统员角色 sys_*r* 和网络员角色 net_*r*，但不允许 sys_*u* 用户同时创建这两个角色的主体进程，防止可能发生的网络安全威胁. 例如可信用户以 net_*r* 角色从网上下载一个非可信模块，并通过 sys_*r* 角色将它加载到系统，可能导致病毒或木马的侵害，因此，对这两个角色实施动态职责隔离，如表 1 中的 DRSD 所示.

管理域 admin_*d* 拥有写或修改系统配置与安全策略库的权限，而操作域 operate_*d* 仅能执行与安全相关的操作，所以, admin_*d* 域的可信度高于 operate_*d* 域. 4 个可信角色都允许进入这两个域，但在 DTE 策略配置时，只允许进入 operate_*d* 域的主体进程执行入口程序/sbin/dt 自动转换到 admin_*d* 域. 审计员 adt_*r* 角色则不具备执行/sbin/dt 的权限，因此审计管理用户每次只能进入一个域中运行，如果要进入另一个域，必须退出当前会话后重新登录. 参看表 1 中的 DDSD.

非可信域 user_*d* 不具有任何特权. 系统作出如下规定: (1) 任意可信用户都允许进入这个域，但不允许由一个可信域动态转换到这个域: (2) 为实现可信功能与非可信功能的隔离，不允许同时存在由同一个可信用户以相同角色创建的两个主体进程，分别在可信域与 user_*d* 域中运行，如表 1 中的 DDSD 所示.

此外，我们在内核函数 compute_creds()中实现了权能遗传算法，该函数总是通过系统调用 exec 调用. 同时，我们在内核中建立了默认角色、域和程序文件的权能状态的索引列表，并

且在内核中对安全函数增加了必要的权能校验.

表 1　不同隔离类型的实现举例

Constrains Types	Exclusive Relationships
SRSD	(ser_*r*, sys_*r*) (sec_*r*, net_*r*) (sec_*r*, adt_*r*) (sys_*r*, adt_*r*) (net_*r*, adt_*r*)
DRSD	(net_*r*, sys_*r*)
DDSD	(admin_*d*, operate_*d*) (admin_*d*, user_*d*) (operate_*d*, user_*d*)

2.3.3　新模型的分析、比较与特点

新模型通过对进程特权的控制, 有效地实现了最小特权原理. 该模型具有以下特点: (1) 首次有效地结合 RBAC 和 DTE 与 POSIX 权能机制, 支持用户、角色、域与权能之间的层次映射关系. 通过角色与域的结合简化授权管理: 通过细化的域定义实现角色职责中可信功能与非可信功能的隔离: 通过 POSIX 权能机制与 DTE 支持的域转换, 进程的特权状态可以动态调节. (2) 提出实施最小特权的 3 层实现机制: 管理层、功能控制层与执行层. 新机制推广了子域控制机制, 实现了子域控制机制的动态化. (3) 提出了新颖的“功能隔离原理”概念, 并进行形式化规范. (4) 实现了角色的职责隔离与域的功能隔离. (5) 提出了新的权能机制、两个新的权能集: 域权能集 B_d 与角色权能集 B_r, 以及新的权能遗传公式.

3　隐蔽通道分析

隐蔽通道对操作系统的安全构成严重威胁, 对于高等级安全操作系统, 所有的评测标准都要求进行隐蔽通道的分析与处理. 然而, 隐蔽通道分析是众所周知的难题, 迄今缺乏坚实的理论基础与系统的分析方法. 我们重点讨论两个关键问题: (1) 如何保证隐蔽通道标识的完备性, 即彻底搜索隐蔽通道的问题; (2) 如何提高隐蔽通道标识的效率. 在此基础上, 我们提出: (1) 新的隐蔽通道标识完备性的理论基础; (2) 新的通用隐蔽通道标识架构; (3) 新的隐蔽通道标识方法－回溯搜索法. 最后, 阐述这些新方法在安胜 OS 中的实现.

3.1　背景与相关工作

1973 年, Lampson[24]首先提出了隐蔽通道的概念. 他关于隐蔽通道的定义是: “如果一个通道既不是设计用于通信, 也不是用于传递信息, 则称该通道为隐蔽通道”. 1990 年, Tsai等 [25]提出了一种更为确切的定义: “给定一个强制安全策略模型M和它在一个操作系统中的解释$I(M)$, $I(M)$中两个主体$I(S_i)$和$I(S_j)$之间的任何潜在通信都是隐蔽的, 当且仅当模型M中的相应主体S_i和S_j之间的任何通信在M中都是非法的”. 我们认为, 任何定义都至少应当反映下述内涵: (1) 隐蔽通道分析仅与强制安全策略模型有关; (2) 隐蔽通道分析与机密性模型和完整性模型相关; (3) 隐蔽通道分析与TCB规范有关.

30年来, 文献中提出的隐蔽通道标识方法共有以下4类: (1) 语法信息流法; (2) 无干扰法; (3) 共享资源矩阵法; (4) 语义信息流法. 其中, 最有影响的是后两种方法.

Kemmere[26]提出的共享资源矩阵法, 又称SRM方法, 由于其简单与直观的特点, 应用最为广泛. SRM方法还有下述具有代表性的推广: 隐蔽流树方法CFT[27]: McHugh[28]提出的改进SRM方法, 以及Kemmerer与Taylor[29]提出的模块化SRM方法. 但是, SRM类方法具有下述先天性的不足: 构造初始SRM矩阵的工作量巨大, 并且没有有效的构造工具: 不能证明单个的TCB原语(或原语对)是安全隔离的, 不利于增量分析新的TCB原语. 此外, SRM方法是一种“保守”

方法，当构造SRM矩阵时，不要求显式地区分初始直接流，并且在传递闭包操作时不控制信息流的迁移．所有的分析工作都留到最后一步完成，因此，将会导致信息流的急剧增长，不但增加了分析工作量，而且被标识的通道往往不是真实隐蔽通道．

Tsai等人[25]提出了语义信息流法：通过分析编程语言的语义、代码和内核中使用的数据结构，发现变量的可见性/可修改性，确定内核变量的间接可修改性．对源代码进行信息流分析，从而确定内核变量的间接可见性．与SRM方法相比较，Tsai方法具有搜索更彻底、可以发现大量伪非法流、可以确定安置审计代码与时间延迟变量的位置等优点．然而，Tsai对新引入的概念没有给出准确的定义，并推导出一系列不准确的结论．在实际应用时，从原语出发构造函数依赖关系集合的过程没有退出机制，作了许多“无用功”，并容易产生状态爆炸．

其他有关隐蔽通道的内容，请参看文献 [30].

3.2 隐蔽通道标识完备性的理论基础

所有的国内外评测标准都要求“彻底”搜索隐蔽通道，但是都没有给出“彻底性”的显式规范．为此，本节建立隐蔽通道标识完备性的理论基础．首先，引入以下记法与概念．

从 TCB 接口看来，信息流是由一系列 TCB 原语操作引起的．四元组

$$(s, op, vs_1, vs_2) \in S \times OP \times VS \times VS$$

表示一个基本的信息流单元，其中 S 表示所有主体的集合：OP 表示 TCB 原语的集合，VS 表示共享变量集合．该四元组说明，主体 S 调用原语 OP 之后，产生由变量 vs_1 到变量 vs_2 的信息流．序列 $vs_0\ s_1(op_1, vs_1)\cdots s_n(op_n, vs_n)$ 表示主体 s_1 调用原语 $op_1,\cdots$，主体 s_n 调用原语 op_n 之后，产生源于 vs_0 的信息流 vs_0:$vs_0 \to vs_1 \to \cdots \to vs_n$.

通过一个原语操作产生的信息流称为直接流，可以表示为 $vs_0\ s(op, vs_1)$. 相反，通过多个原语操作产生的信息流称为间接流．可见，直接流仅有一个迁移步：间接流则有多个迁移步．内部流是在内部 TCB 变量之间流动的信息流．相反，进程流是穿越 TCB 接口，并返回 TCB 外部的信息流．如果用变量 *In* 和 *Out* 分别表示原语的输入与输出，则进程流可以表示为

$$vs_0\ s_1(op_1, vs_1)\cdots s_n(op_n, Out),\ vs_0 \in VS \bigcup \{In\}.$$

令进程 P_S 和 P_R 分别表示发送与接收主体，此处 P_S 的安全级别不被 P_R 的安全级别支配．显然，任何从 P_S 到 P_R 的信息流都构成隐蔽通道．

McHugh[28]认为，隐蔽通道一定始于“用户输入 *In* ”，并终止于“用户输出 *Out* ”．事实上，TCB外部的输入不是产生隐蔽通道的必要条件．只要隐蔽通道的第一个操作可以修改一个共享TCB变量，就可以开始传输信息．对这些与输入无关的通道，在它们的信息流序列中，可以忽略初始变量．

基于以上分析，隐蔽通道一般可以表示为

$$P_S(op_1, vs_1)\cdots P_S(op_m, vs_m)\ P_R(op_{m+1}, vs_{m+1})\cdots P_R(op_n, Out).$$

当然，也可以将与输入相关的隐蔽通道显式地表示为:

$$In\ \ P_S(op_1, vs_1)\cdots P_S(op_m, vs_m)\ P_R(op_{m+1}, vs_{m+1})\cdots P_R(op_n, Out).$$

我们将隐蔽通道分为以下 6 类：直接通道、间接通道、原子通道、单通道、复通道和 SR^+ 通道．由一个发送操作和一个接收操作组成的隐蔽通道称为直接通道(DCC)，可以表示为

$P_S(op_1,vs_1)\ P_R(op_2,Out)$. 发送序列或接收序列由多于 1 个操作组成的隐蔽通道称为间接通道(ICC). 假设 C 和 C_S 是两个不同的隐蔽通道. 不考虑目的变量 Out, 如果 C_S 的信息流序列是 C 的信息流序列的子序列, 则称 C_S 是 C 的子通道, 且记为 $C_S \subset C$. 因此如果 C 的信息流序列形如 $P_S(op_1,vs_1)\cdots P_S(op_m,vs_m)\ P_R(op_{m+1},vs_{m+1})\cdots P_R(op_n,Out)$, 则 C_S 一定形如

$$P_S(op_i,vs_i)\cdots P_S(op_m,vs_m)\ P_R(op_{m+1},vs_{m+1})\cdots P_R(op_j,Out),$$
$$1\leqslant i\leqslant m\wedge m<j\leqslant n\wedge C_S\neq C.$$

假设 C 和 C_S 是两个不同的隐蔽通道. 不考虑主体与目的变量, 如果 C_S 的“操作-变量”序列是 C 的“操作-变量”序列的子序列, 则称 C_S 是 C 的广义子通道, 且记为 $C_S\angle C$. 即 C_S 形如 $P_S(op_i,vs_i)\cdots P_R(op_j,Out),\ 1\leqslant i<j\leqslant n\wedge C_S\neq C$. 没有广义子通道的隐蔽通道称为原子通道(ACC). 由定义可知, 原子通道的信息流序列不包含任何其他通道的操作序列. 此外, 原子通道的发送序列由一个发送操作组成. 否则, 从原始信息流序列中去掉第一个发送操作, 我们就可以获得另一个隐蔽通道, 即原始通道的广义子通道. 令 CC 为所有隐蔽通道的集合, 则原子通道可以表示为

$$acc: P_S(op_1,vs_1)\ P_R(op_2,vs_2)\cdots P_R(op_n,Out),\neg\exists(cc)\left(cc\in CC\wedge cc\angle acc\right).$$

没有子通道的隐蔽通道称为单通道(SCC), 相反, 具有子通道的隐蔽通道则称为复通道(PCC). 类似于原子通道, 单通道只有一个发送操作, 可以表示为

$$scc: P_S(op_1,vs_1)\ P_R(op_2,vs_2)\cdots P_R(op_n,Out),\neg\exists(cc)\left(cc\in CC\wedge cc\subset scc\right).$$

直接通道、原子通道和单通道的共同特点是仅有一个发送操作. 只有一个发送操作的隐蔽通道称为 SR^+ 通道(SR^+CC), 可以表示为

$$P_S(op_1,vs_1)\ P_R(op_2,vs_2)\cdots P_R(op_n,Out).$$

上述 6 种隐蔽通道的关系可以表示如下, 它们是衡量隐蔽通道标识完备性的基础. 首先, 所有隐蔽通道的集合 CC 可以进行两类不同的分划: (1)分划为 DCC 与 ICC; (2)分划为 SCC 与 PCC. 亦即, $CC=DCC\cup ICC$ 且 $DCC\cap ICC=\varnothing$, 其中 $\varnothing$ 表示空集; 以及 $CC=SCC\cup PCC$ 且 $SCC\cap PCC=\varnothing$. 此外, 不难验证存在下述关系:

$$DCC\subseteq ACC\subseteq SCC\subseteq SR^+CC.$$

需要着重说明的是, $ACC\subseteq DCC$ 不成立. 换句话说, Tsai 的观点“原子通道一定是直接通道”是错误的. 事实上, 仅当对主体的操作没有约束时, 原子通道才是直接通道. 这一事实最好通过一个实例进行说明. 例如, 间接通道 $C_1: P_S(op_1,vs_1)\ P_R(op_2,vs_2)\ P_R(op_3,Out)$ 有一个广义子通道 $C_2: P_S(op_2,vs_2)\ P_R(op_3,Out)$. 这时, 如果对主体的操作具有约束, 使 P_S 与 P_R 之间的安全级支配关系没有通过安全检查, 则直接通道 C_2 不再存在. 因此, 间接通道 C_1 是一个不包含直接通道的原子通道.

注意, 消除所有的原子通道并不保证隐蔽通道标识的完备性. 为了彻底地搜索隐蔽通道, 可以递归地标识与处理原子通道, 直到不能再发现原子通道时为止. 此外, 一个隐蔽通道至少包含一个单通道, 且一个复通道至少有一个子单通道. 因此, 应当标识所有的单通道, 或者递归地标识和处理原子通道. 值得指出的是, 在标识隐蔽通道时, 也应当同时考虑隐蔽通道的处理策略. 例如, 消除隐蔽通道与对隐蔽通道实施带宽限制是不同的. Tsai 认为, 复通道的带宽

不会高于它的子通道带宽, 因此, 如果将一个复通道的所有子通道的带宽限制为低于一个阈值, 则复通道的带宽也可以低于该阈值. 事实上, 上述结论并不永远正确, 并可能导致错误的带宽估计. 此外, 递归地处理原子通道也不能保证恰当的带宽限制.

为了保证隐蔽通道标识的完备性, 除系统调用之外, 我们认为还需要扩大 TCB 原语的范围. 当在内核系统中标识隐蔽通道时, 通常假设系统调用是 TCB 的惟一的接口, 并被选择为 TCB 原语. 但是, 中断与异常(exception)也可能使系统转换为内核模式, 且内核线程也可能产生隐蔽通道. 在第 3.5 节中, 我们将通过安胜 OS 的实例进一步予以说明.

3.3 一种通用的隐蔽通道标识架构

本节提出一种新的通用隐蔽通道标识架构, 试图弥补文献中的空白.

我们认为, 可将隐蔽通道视为两个主体(发送进程与接收进程)之间的信息流. 信息流形成隐蔽通道时, 一定产生由发送进程到接收进程的信息流, 与经过哪些变量无关, 因此, 应当检查具有某些特征的主体之间的、违反安全策略的信息流. 通过信息流主体的安全级检测, 以及主体与客体的约束条件, 可以校验信息流的合法性.

新提出的通用隐蔽通道标识架构共分 3 个阶段: (1)分析 TCB 原语, 标识由每一个原语产生的直接流; (2)根据特定的规则, 由直接流构造间接流; (3)分析可疑信息流, 标识潜在的隐蔽通道. SRM 等类似的方法均仅为表示和组合信息流的技术, 仅包含阶段(2). Tsai 等人的方法是搜索直接流和标识直接通道的技术, 仅包含阶段(1)和(3). 因此, 新方法是一种完备的通用架构.

在应用 SRM 方法时, 信息流通过传递闭包任意地组合. 仅当所有可能的组合都完成之后, 才开始进行分析. 相反在新的架构下, 在隐蔽通道标识的全过程中都进行信息流分析. 更具体地说, 初始分析阶段寻找直接流; 中间分析阶段组合信息流; 最后分析阶段标识潜在的隐蔽通道.

3.4 隐蔽通道的回溯搜索方法

我们新提出的隐蔽通道回溯搜索法, 对 SRM 方法和语义信息流分析方法都进行了重大的改进. 在原始的 SRM 方法中, 信息流由起点开始向前迁移, 且在迁移过程中确定共享变量之间的间接引用关系. 新的基于 SRM 方法的回溯搜索法可以视为一种“反向”的 SRM 方法: 信息流由终点开始回溯到起点, 并发现间接的修改关系. 如果某个原语 op_i 修改了引用变量 vs_x 的变量 vs_y, 产生信息流: $vs_x \rightarrow vs_y$, 且另一个原语 op_j 修改了变量 vs_x, 则我们称原语 op_j 可以间接地修改变量 vs_y. 反向的传递闭包操作从可见变量开始它的第 1 个迁移轮. 所有生成的信息流都是进程流, 防止了无效的内部迁移. 为使 SRM 矩阵能够记录全部信息流路径, 在传递闭包的操作过程中, 当信息流迁移到一个新的变量时, 我们不仅标记相应的矩阵项, 而且记录先前的原语和变量序列. 此外, 我们通过“轮(round)”执行传递闭包操作. 信息流有序地扩展, 且每一轮中每个信息流只前进 1 步, 与下一个直接流相连接. 其次, 对每一个矩阵列都增加主体对原语的限制, 由此判断信息流迁移的有效性. 我们提出了一系列迁移约束规则, 信息流的迁移必须满足特定的迁移条件. 最后, 在每一个迁移轮之后, 分析新扩展的信息流, 并排除伪信息流与无效信息流.

以原语“*Lock*”为例, 新方法对矩阵的反向传递闭包操作仅需运行 3 轮, 一共生成了 30 个

信息流. 其中, 在迁移过程中已经发现了 20 个伪信息流, 仅剩下 10 个信息流供进一步分析. 在 2 个迁移轮之内, 26 个信息流停止了迁移. 相反, 当采用基本的 SRM 方法时, 不算内部流, 总共得到 57 组需要进一步分析的可疑信息流. 更糟的是, 信息流的数量随信息流长度的增加急剧增长. 例如, 在 3 个迁移步之内, 可疑信息流的数量大约为 100 个. 尽管可以搜索到所有的单通道, 但是分析工作量十分沉重.

与语义信息流分析方法相比较, 我们的新方法具有以下特点: (1) 扩大了系统调用的选择范围, 分析全部系统调用. (2) 扩大了要分析的代码范围, 对内联汇编代码也进行分析. (3) 现有的方法从系统调用的入口函数着手, 然后分析该系统调用能够读写的共享变量. 新的回溯搜索法从共享变量着手, 然后找出能够读写该共享变量的所有的系统调用. (4) 当处理已经标识的隐蔽通道时, 新方法易于选择控制共享变量的最佳位置. (5) 原方法需要更多的分析步骤, 因而对自动工具更为依赖. (6) 新方法与现有方法最显著的区别是, 在原方法的第 3 步, 才由所得到的信息流路径中找到涉及共享变量的信息流. 应用 FCD(函数调用依赖关系)分析信息流的过程缺乏退出机制, 做大量“无效”工作, 且容易产生状态爆炸. 相反, 回溯搜索法只分析涉及共享变量的函数, 因为系统调用读写共享变量必然通过这些中间函数实现. 由于具有退出机制, 回溯搜索法有效地避免了状态爆炸.

有关基于语义信息流分析的回溯搜索方法的标识过程及其他细节, 请参看文献 [31].

3.5　安胜 OS 的隐蔽通道标识

我们的方法在安胜 OS 中获得成功的应用, 包括顶层规范级和源代码级的分析. 共标识出 20 条隐蔽存储通道, 其中, 进程子系统 1 个, 文件子系统 13 个, IPC 子系统 5 个, 网络子系统 1 个. 如果根据编码特征分类, 其中事件计数型通道 8 个, 资源耗尽型通道 12 个. 如果根据噪音特征分类, 其中噪音通道 16 个, 无噪通道 4 个.

上述结果是首次报道的基于 Linux 内核的安全操作系统的隐蔽通道分析结果, 包括下述从未见诸于其他文献的新型通道: 子目录耗尽通道; 同名子目录通道; 最近访问时间通道; 索引节点号通道; IPC 标识符重用通道和 IPC 名称通道.

在源代码分析之前, 我们应用基于 SRM 方法的回溯搜索法对安胜 OS 的顶层规范标识隐蔽通道, 总共发现 10 个隐蔽通道. 应用基于语义信息流方法的回溯搜索方法在代码级分析之后, 再次发现了它们.

此外, 在安胜 OS 的隐蔽通道标识中, 原语扩大到包括下述例程 (routine): (1) 通过非特权指令可以调用的 3 个异常事件: breakpoint (#BP), overflow (#OF) 与 bound range (#BR). (2) 不能从内核外直接调用, 但可以故意通过执行出错代码调用的 8 个异常事件. 例如, 除法错误 (#DE), 页错误 (#PF) 等. (3) 可被其他操作调用或定期发生的中断, 例如时钟中断. (4) 遵循某些约定或执行可被其他操作干扰的内核线程, 例如 swap daemon thread 和某些文件系统的 journal thread 等.

3.6　隐蔽通道分析的主要成果

在高等级安全操作系统的隐蔽通道分析中, 我们取得以下成果: (1) 提出隐蔽通道标识完备性的理论基础. (2) 提出一种通用的隐蔽通道标识架构. (3) 提出新的隐蔽通道回溯搜索方法, 包括基于共享资源矩阵方法的回溯搜索法与基于语义信息流方法的回溯搜索法. (4) 首次报道

基于 Linux 内核的安全操作系统的隐蔽通道分析结果. (5) 发现了一些从未见诸于其他文献的真实隐蔽通道.

值得指出的是, 虽然我们这里主要研究的是存储隐蔽通道, 但是许多讨论也适用于定时隐蔽通道.

4 结束语

众多因特网安全事件的发生表明, 为了对抗现代计算环境中的安全威胁, 来自安全操作系统的支持是必不可少的. 本文总结了在安胜 OS 的设计与开发实践中, 我们在高等级安全操作系统设计中的 3 个关键课题: 体系结构、安全模型与隐蔽通道分析中取得的具体成果. 我们阐述了新提出的安全体系结构, 新的机密性模型、完整性模型和特权控制模型, 分别介绍了它们的特色, 并与现有的技术做了分析和比较. 最后, 介绍了它们在安胜 OS 中的实现.

为了解决隐蔽通道分析中存在的基本问题, 我们提出了隐蔽通道标识完备性的理论基础、一种通用的隐蔽通道标识框架, 以及高效的回溯搜索方法. 这些新方法在安胜 OS 中的成功实现表明, 它们可以简化并加快整个隐蔽通道的分析过程.

本文成果中的一些早期设想与技术路线, 可以参看文献 [32].

致谢 感谢中国科学院知识创新工程重要方向性项目“结构化保护级”安全操作系统设计项目组的全体成员. 特别感谢: 季庆光、朱继锋、沈晴霓、李丽萍、沈建军、何建波等为本文所做的贡献.

参 考 文 献

1 Spencer R, Smalley S, Loscocco P, et al. The flask security architecture: System support for diverse security policies. In: Proceedings of the 8th USENIX Security Symposium. 1999. 123—139
2 Wright C, Cowan C, Morris J, et al. Linux security modules: General security support for the Linux kernel. Usenix Security Symp, Usenix Assoc, 2002. 17—31
3 Kuhnhauser W, Ostrowski M, A framework to support multiple security policies. In: Proceedings of the 7th Canadian Computer Security Symposium. Ottawa, Canada. 1995
4 Bell D E. Modeling the multipolicy machine. In: Proc of the New Security Paradigm Workshop, 1994. 2—9
5 Bell D E, La Padula L J. Secure computer system: Unified exposition and multics interpretation. Mitre Report, MTR-2997 Rev. 1, 1976
6 Ott A. Regel-Basierte zugriffskontrolle nach dem Generalized framework for access controlansatz am beispiel Linux. Diplomarbeit Universitat Hamburg, 1997
7 Lee T. Using mandatory integrity to enforce “commercial” security, Proceedings of IEEE Symposium on Security and Privacy. Oakland: IEEE Computer Society Press, 1988. 140—146
8 Bell D E. Security policy modeling for the next-generation packet switch. In: Proceedings of IEEE Symposium on Security and Privacy. Oakland: IEEE Computer Society Press, 1988. 212—216
9 Mayer FL. An interpretation of refined Bell-La padula model for the Tmach kernel. In: Proc of the 4th Aerospace Computer Security Application. Orlando: IEEE Computer Society Press, 1988. 368—378
10 Secure Computing Corporation. Assurance in the Fluke microkernel: Formal top-level specification. CDRL A004. Technical Report, Secure Computing Corporation, 1999
11 季庆光, 卿斯汉, 贺也平. 一个改进的可动态调节的机密性策略模型. 软件学报, 2004, 15(10): 1547—1557
12 Biba K. Integrity Consideration for Secure Computer Systems, Technical Report, MITRE TR-3153, Hanscom Air Force Base MITRE Corporation, 1977
13 Clark D, Wilson D. A comparison of commercial and military computer security policies. In: IEEE Symposium on Security and Privacy. Oakland: IEEE, 1987. 184—194
14 Karger P. Implementing commercial data integrity with secure capabilities. IEEE Symposium on Security and Privacy. Oak-

land: IEEE, 1988. 130—139
15 O'Brien R, Rogers C. Developing Applications on LOCK. In: Proc 14th National Computer Security Conference. Washington D C, 1991. 147—156
16 National Security Agency. Security Enhanced Linux (SELinux). http://www .nsa.gov/selinux. 2001
17 Ji Q G, Qing S H, He Y P. A formal model for integrity protection based on DTE technique. Sci China Ser F-Inf Sci, 2006, 49(5): 545—565
18 Saltzer J H, Schroeder M D. The protection of information in computer systems. Proc IEEE, 1975, 63(9): 1278—1308
19 Ferraiolo D, Cugini J, Kuhn D. Role based access control (RBAC): Features and motivations. In: Proceedings of 11th Annual Computer Security Applications Conference, New Orleans:IEEE Computer Society Press, 1995, 241—48
20 Hoffman J, Implementing RBAC on a type enforced system. In: Proceedings of 13th Annual Computer Security Applications Conference, 1997. 158—163
21 Chandramouli R. A framework for multiple authorization types in a healthcare application system. In: Proceedings of the 17th Annual Computer Security Application Conference, Los Alamitos: IEEE Computer Society Press. 2001, 137—148
22 Portable Applications Standards. Committee of IEEE Computer Society. Standards Project, Draft Standard for Information Technology—Portable Operating System Interface (POSIX), PSSG Draft 17. New York: IEEE, Inc, 1997
23 Ji Q G, Qing S H, He Y P. A new formal model for privilege control with supporting POSIX capability mechanism. Sci China Ser F-Inf Sci, 2005, 48(1): 46—66
24 Lampson B. A note on the confinement problem. Comm ACM, 1973, 16(10): 613—615
25 Tsai C, Gligor V, Chandersekaran C. On the identification of covert storage channels in secure systems. IEEE Trans Softw Eng, 1990, 16(6): 569—580 [DOI]
26 Kemmerer R. Shared resource matrix methodology: An approach to identifying storage and timing channels. ACM Trans Comput Syst, 1983, 1(3): 256—277 [DOI]
27 Kemmerer R. Covert flow trees: A visual approach to analyzing covert storage channels. IEEE Trans Softw Eng, 1991, 17(11): 1166—1185 [DOI]
28 McHugh J. Handbook for the Computer Security Certification of Trusted Systems-Covert Channel Analysis. Technical Report, Naval Research Laboratory, 1996
29 Kemmerer R, Taylor T. Modular covert channel analysis methodology for trusted DG/UX. IEEE Trans Softw Eng, 1996, 22: 224—235
30 卿斯汉. 高安全等级安全操作系统的隐蔽通道分析. 软件学报, 2004, 15(12): 1837—1849
31 卿斯汉, 朱继峰. 安胜安全操作系统的隐蔽通道分析. 软件学报, 2004, 15(9): 1385—1392
32 Qing SH, Ji Q G. Formal model design for secure operating system (invited paper). In: Proceedings of ITI First International Conference on Information & Communications Technology (ICICT2003), Egypt. 2003. 27—47