一种基于 TE 技术实现 Clark-Wilson 模型的方法

何建波[1], 郭 新[3], 卿斯汉[1,2]
(1. 中国科学院信息安全技术工程研究中心, 北京 100080; 2. 北京大学软件与微电子学院, 北京 102600;
3. 北京城市学院计算机实验教学中心, 北京 100083)

摘 要: 分析了当前 Clark-Wilson 完整性模型实现机制的不足, 提出了一种基于 TE 实现 Clark-Wilson 模型的方法. 首先讨论了 TE 对 Clark-Wilson 模型的支持能力, 然后给出了用 TE 实现 Clark-Wilson 模型的配置规则和约束. 在实现中, 扩展了模型的(userid, TP, list of CDIs) 三元组, 引入了角色层, 以更实用的方式实现了职责隔离. 同时, 实现机制实现了对 TP 的保护, 有效地保障了 TP 功能的正确性, 提高了系统正确控制 TP 操作的可信度, 减少了对 Clark-Wilson 模型验证规则的依赖.

关键词: Clark-Wilson 模型; 完整性; TE 机制; 职责隔离
中图分类号: TP309 文献标识码: A 文章编号: 0372-2112 (2008) 02-0216-08

An Approach to Enforcing Clark-Wilson Model Based on Type Enforcement

HE Jiarr-bo[1], GUO Xin[3], QING Si-Han[1,2]
(1. *Engineering Research Center for Information Security Technology, the Chinese Academy of Sciences, Beijing* 100080, *China*;
2. *School of Software and Microelectronics, Peking University, Beijing* 102600, *China*;
3. *Computer Testing Teaching Center, Beijing City University, Beijing* 100083, *China*)

Abstract: This paper analyzes the limitations of current implementation technologies to Clark-Wilson integrity model, and presents an approach to enforcing Clark-Wilson model based on Type Enforcement. Firstly, the advantages of this approach are discussed. Secondly, the configuration rules and constraints that implement Clark-Wilson model based on TE are given. The triple (userid, TP, list of CDIs) of Clark-Wilson model is extended and replaced by the quarter-tuple (userid, role, TP, list of CDIs), which implement the separation of duty principle more practically. In addition, the protection of TP is covered by the TE, which ensures the correct functionality of TP, enhances the degree of confidence that TP function correctly, and releases the dependency on the certification procedure.

Key words: Clark-Wilson model; integrity; Type Enforcement(TE); separation of duty

1 引言

D D Clark 和 D R Wilson[1] 分析了军用和商务领域安全需求的不同, 提出了"以良构事务(well-formed transaction) 来操作数据和职责隔离(separation of duty)"思想为核心的完整性保护模型, 简称为 Clark-Wilson 模型. Clark-Wilson 模型主要通过完整性验证过程(Integrity Verification Procedures, IVP) 和转换过程(Transformation Procedure, TP) 来实现对数据的完整性保护. 为了便于实现, Clark-Wilson 模型定义了执行规则(enforcement rules) 和验证规则(certification rules). Clark-Wilson 模型被业界认为是完整意义上的完整性目标、策略和机制的起源, 是唯一实现三个完整性保护目标——防止未授权用户修改数据; 保持数据的内在和外在一致性; 防止授权用户以非授权的方式修改数据——的完整性模型[2].

任何一个安全模型的实现都必须依赖于一定的系统安全机制. 对于 Clark-Wilson 模型, 人们提出了不同的实现机制. 文献[3]提出用 Biba 模型的完整性类属和部分可信主体来实现 Clark-Wilson 模型, 类属表示数据的完整级, 每个部分可信主体对应一个 TP. 此外, 文献[3]还提出 8 条管理规则来实现 Clark-Wilson 模型的执行规则和验证规则. 文献[4] 提出了一种结合安全权能体系(SCAP) 和格安全模型的实现方式, 用权能和改进的 ACL 实现 Clark-Wilson 模型的执行规则, 而令牌能力(token capability) 则作为 ACL 的补充被用来实现职责隔离. 文献[5] 提出了用 UNIX 操作系统的 setuid 机制来实现 TP,

收稿日期: 2006-06-31; 修回日期: 2007-07-18
基金项目: 国家 973 重点基础研究发展规划(No. G1999035802); 国家自然科学基金(No. 60573042); 北京市自然科学基金(No. 4052016)

文献[6]提出基于 RBAC 来实现 Clark-Wilson 模型. 这些研究探讨了 Clark-Wilson 模型在安全操作系统中的实现, 促进了 Clark-Wilson 模型在实际开发中的应用. 但是这些实现方式存在如下不足:

(1)实现方法比较复杂, 难以配置和管理. 尽管文献[3]证明了基于格的方式可以实现 Clark-Wilson 模型, 但是实际操作起来仍然难以处理, 同时, 系统中大量的类属也难以管理[2]. 文献[4]的方法是一种灵活地实现 Clark-Wilson 模型的方式, 但是改进的 ACL 比普通系统中实现的 ACL 更加复杂, 因此也难以维护. 文献[4]的作者也认识到, 令牌能力的引入增加了系统开销, 给系统性能造成了难以承担的负担. 而 setuid 程序分布在系统的各个子系统中, 为系统管理员管理这些程序带来了不便.

(2)实现机制自身安全性不足, 没有把TP 的保护纳入实现机制中. 文献[3]把TP 定义为部分可信主体, "可信主体"的概念就默认了对 TP 功能正确性的假设, TP 的保护需要系统额外的机制来保障. 文献[4]采用的能力和 ACL 机制是一种自主访问控制, 在系统实现时要考虑特洛伊木马可能带来的安全问题. 此外, 如何控制能力的传递和吊销, 也是一个难以解决的问题. 在文献[5]的实现中, 必须要考虑 setuid 自身的安全问题[7], 此外, 超级用户 root 可以执行所有的TP, 这违反了极小特权原则. 一旦恶意用户或进程获取了超级用户的身份, 将导致整个系统完整性保护的失效. 文献[6]考虑了用角色来关联TP, 但是没有考虑对 TP 的保护.

(3)所有这些机制都只支持执行规则的实现, 只实现了模型的(userid, TP, list of CDIs)三元组, 而没有对模型的验证规则进行支持.

实际上, 尽管 Clark-Wilson 模型是目前最能反映完整性保护需求的模型, 但是在操作系统中完全实现它是困难的. McLean[7]认为, 形式表达的缺乏和无法依据形式模型进行形式分析是 Clark-Wilson 模型没有被广泛使用的主要原因. 为此, 文献[8]对 Clark-Wilson 模型的形式化进行了探索, 但是如何将形式化分析的结果用于实际应用, 仍需进一步研究. 由于 Clark-Wilson 模型的验证规则需要系统外在的干预, 而从现有的实现机制来看, 多数实现方式都注重于从访问控制的角度来实现 Clark-Wilson 模型的执行规则. 对于验证规则, 这些实现都假设系统TP 的程序逻辑是正确的, 如用可信主体实现TP[3], 因此不需要用验证规则来保障 TP 功能的正确性和系统保持完整性有效状态. 但是, 系统实现机制完全不考虑对 IVP 的支持显然是不恰当的, 一种好的实现机制有助于减少系统实现 Clark-Wilson 模型时对 IVP 过程的依赖. 正如 Thomsen 等[9]所指出的, 实现 Clark-Wilson 模型的系统对 IVP 的依赖程度取决于实现机制的本质, 如果系统安全机制能控制 TP 对数据操作的正确性, 同时能保护 TP 自身的完整性, 那么在实现 Clark-Wilson 模型时没有必要实现 IVP. 因此, 选择一种实现机制来实现 Clark-Wilson 模型时, 应当看它是否能对 TP 的正确执行提供额外的保障.

作为一种强制访问控制机制, TE(Type Enforcement)[10]具有灵活, 实用和可实现细粒度访问控制等优点, 目前已有很多系统把 TE 作为系统级访问控制机制, 如 DTMach[11]和 SELinux[12]. 虽然 TE 得到了越来越广泛的应用, 但是, 目前很少有文献说明或者探讨过如何用它来实现 Clark-Wilson 模型. 文献[9]比较了TE 和 UNIX setuid 机制实现 Clark-Wilson 模型的不同, 但是他们的工作仅限于对两种机制实现 Clark-Wilson 模型能力的比较, 而没有给出配置 TE 实现 Clark-Wilson 模型的通用规则, 尤其是没有指出如何用 TE 支持 Clark-Wilson 模型的职责隔离原则. 随后, Thomsen 在文献[19]中讨论了如何用 TE 实施基于角色应用安全策略, 他认为 Clark-Wison 完整性策略是一种基于角色策略, 并在实现 Clark-Wilson 模型(userid, TP, list of CDIs)三元组时, 引入了角色层, 讨论了"用户-角色"层的职责隔离实现. 但是, 其实现方式仍然缺乏配置 TE 的具体细节, 同时也没有讨论如何配置角色和域的关系来实现职责隔离.

基于 Thomson 等人的工作, 本文首先分析了 TE 对 Clark-Wilson 模型的支持能力, 指出用TE 实现 Clark-Wilson 模型能够克服现有实现机制的不足; 然后研究了如何用 TE 实现 Clark-Wilson 模型的执行规则, 给出了配置 TE 实现 Clark-Wilson 模型的具体配置规则和约束. 在实现职责隔离时, 扩展了模型的"用户/TP/数据"三元组, 引入了角色层, 以更实用的方式实现了职责隔离. 另外, 与前面提到的各种实现机制不同的是, TP 的保护被纳入到 TE 访问控制机制中, 从而更好地保障了 TP 功能的正确性, 提高了系统正确控制 TP 操作的可信度, 减少了实现过程对 IVP 的依赖.

2　Clark-Wilson 模型和 TE 访问控制机制

2.1　Clark-Wilson 模型

Clark-Wilson 模型把系统数据集分成两个不相交的子集, 分别称为受限数据项(constrained data items, CDIs)和非受限数据项(unconstrained data items, UDIs). 一个 CDI 针对某一特定应用属性有完整性保护要求, 如银行存款簿中的信息栏, 而一个 UDI 客体则是系统中的普通数据客体, 没有完整性保护要求, 如普通的用户文本. 系统中的任一数据客体不能既是 UDI, 又是 CDI.

Clark-Wilson 模型通过 IVP 和 TP 实现完整性保护. 前者主要保障 CDI 始终处于满足完整性策略的有效状态中, 而后者则用于保持或者提升系统数据的完整性. 在 Clark-Wilson 模型中, 用户只有通过执行 TP 才能操作

CDI, TP 实现了商务应用中良构事务的概念.

为了便于实现, Clark-Wilson 模型定义了完整性验证规则($C1$~ $C5$)和完整性执行规则($E1$~ $E4$)[1]. 执行规则规定了实现 Clark-Wilson 模型的系统安全机制必须满足的安全需求. 验证规则规定了执行 Clark-Wilson 模型的系统需要外在干预的安全需求. 由于验证规则并不直接影响数据的完整性, 只是以一种监控功能保障 IVP 的功能正确性, 因此也被称为完整性监控规则. 表 1 简要描述了这 9 条规则.

表 1 Clark-Wilson 完整性规则

规则	描 述
$C1$	IVPs 必须确保 CDIs 始终处于有效状态中, 即满足完整性保护要求
$C2$	TP 对任一 CDI 的操作结果仍是一个有效的 CDI
$C3$	$E2$ 的执行必须满足职责隔离(Separation of Duty)和极小特权原则(principle of least privileges).
$C4$	TP 的行为必须被日志记录
$C5$	TP 对一个 UDI 操作的结果, 要么是一个 CDI, 要么遗弃该 UDI
$E1$	只有经过验证的 TP 才能操作 CDIs, 即系统必须维持(TP_i, (CDI_1, CDI_2, …))关系列表.
$E2$	用户只能通过授权他们执行的 TP 访问 CDIs, 也就是系统必须维持访问三元组(userid, TP_i, (CDI_1, CDI_2, CDI_3, …)), 这条规则对应于职责隔离要求.
$E3$	每一个请求执行 TP 的用户身份必须是经过系统认证的.
$E4$	只有系统特定的安全管理员才能修改(TP_i, (CDI_1, CDI_2, …))和(userid, TP_i, (CDI_1, CDI_2, CDI_3, …))关系表. 特别地, 具有修改关系列表权限的安全管理员不能执行关系表中任何 TP.

2.2 TE 访问控制机制

TE 把系统看成主动实体(主体)和被动实体(客体)的集合. 根据不同客体的安全属性, 把客体空间分成不同的等价类, 每个等价类称为一个型(type). 型相同的所有客体具有相同的安全保护要求, 如完整性和私密性. 同时, 根据主体在系统中的职能, 把主体空间划分成不同的等价类, 每个等价类被称为一个域(domain). 域定义了主体的功能范畴或者执行环境, 一个给定域中的主体只能执行域所限定的操作. TE 机制通过两张表来控制域对域, 以及域对型的访问. 域定义表(Domain Definition Table, DDT)定义了域对型的访问模式, 域转换表(Domain Transition Table, DTT)定义了域间的访问模式. 在 TE 中, 域和型是系统的访问控制原子单位. 在系统运行时, 通过查表来仲裁主体对客体的访问请求.

一般地, 可以由系统安全官员(SSO)根据具体的安全需求, 先验地配置 DDT 和 DTT 表. 如果 DDT 和 DTT 是静态的, 则系统总能执行 SSO 规定的安全需求.

2.3 TE 对 Clark-Wilson 模型的支持能力分析

相对于前面提到的各种实现机制而言, TE 对 Clark-Wilson 模型支持能力如下:

(1) TP 对数据的操作关系易于用"域–型"关系来实现. Clark-Wilson 模型的实例先验地定义了系统的行为, 行为由定义在数据客体上的操作来决定, "用户-操作-数据"关系模式决定了操作的执行者, 也决定了所表现出来的特定行为. TE 执行访问控制的核心是基于两张静态表: DDT 和 DTT. 只有特定身份的用户, 才能配置这些表, 以满足特定的安全需求. 因此, 可以首先分析 Clark-Wilson 模型元素(TP, CDIs 和 UDI 等)和 TE 机制中域和型的对应关系, 然后定义 TE 实现 Clark-Wilson 模型安全保护的规则和约束.

(2) 如前所述, 实现 Clark-Wilson 模型要考虑如何减少对验证规则的依赖. 在 TE 中, 域限制了主体的操作范围, 可以通过对每个 TP 的可执行程序赋予型属性, 并限制域对这个型的访问来保护 TP 可执行程序的完整性. 把 TP 的保护纳入系统安全机制中, 是对 TP 功能正确性的重要保障. 前面提到的 setuid 机制和能力机制都只实现了 TP 对 CDI 访问的控制, 而没有考虑 TP 自身完整性的保护.

(3) 相对于其他的实现机制, TE 可以更好地满足 Clak-Wilson 模型对极小特权原则(principle of least privileges)的要求. 极小特权原则是安全系统设计必须满足的原则之一[13], 它要求系统只能赋予一个主体完成其任务所必需的权限. Clark-Wilson 模型的规则 C3 要求系统满足极小特权原则, 文献[2]也把极小特权列为实现完整性保护的系统必须满足的准则之一. 在 TE 中, 域被视为进程的逻辑执行空间, 位于特定域中的进程能执行的所有行为都要受到其所属域的限制. 用域来限制进程的执行空间, 可以实现对进程权限的有效控制.

(4) 在现实应用中, 一个任务通常分解成多个子任务, 这些子任务必须以正确的顺序执行才能完成给定的任务. 比如, 在复式簿记中, 借方分录和贷方分录必须保持平衡. 因此, 在计算机系统中, 当一个程序(TP1)创建了一个借方分录的同时, 必须由另一个程序(TP2)根据借方分录创建一个相应的贷方分录, 以保持账目的平衡. 这个过程可以用 TE 中的可信管道(assured pipeline)[10]来实现. 可信管道表示为一系列 TP, 是数据从一个客体流向另一个客体所必经的通路. 在 Clark-Wilson 模型中, 可以用可信管道来实现把一个 UDI 或者 CDI 转换到另一个 CDI 的一系列转换过程. 通过恰当地配置 DTT 和 DDT 来保障可信管道不被旁过(non-bypassed), 并保证实现可信管道的各 TP 以正确的顺序执行, 可以很容易地实现 n 阶段可信管道.

3 配置 TE 实现 Clark-Wilson 模型

3.1 基本数据类型的定义

在 TE 中, CDIs 和 UDIs 都被视为客体. 一个 CDI 是

具有完整性保护要求的客体, 而 UDI 则是系统中的普通客体, 不需要完整性保护. 根据不同的完整性保护要求, 所有的 CDIs 被分成不同的等价类, 每个等价类用一个型来表示. 相应地, 所有的 UDIs 也分成不同的等价类, 每个等价类也用一个型来表示. 比如, 在前面的复式簿记的例子中, 借方分录和贷方分录都是 CDIs, 而从键盘输入的数据, 在没有经过任何 TP 处理前可以视为 UDIs. 因此, 可以定义型 *t_debit* 表示所有的借方记录的完整性属性, 型 *t_credit* 表示所有的贷方记录完整性保护属性, 型 *t_keyboard* 表示从键盘输入的数据型属性 (实际上没有完整性保护要求).

为了表示与 *CDIs* 和 *UDIs* 有关的型, 有如下定义,

定义 **1**　*Type_CDIs* 和 *Type_UDIs* 分别表示所有与 CDIs 相关的型和所有与 UDIs 相关的型的集合.

逻辑上, TP 可以视为有两个状态: 静态的 TP 可执行程序和动态的 TP 进程像. 因此, 需要从两个方面来保护TP. 首先, TP 可执行程序的完整性直接影响到 TP 功能的正确性, 所以有必要保护TP 可执行程序的完整性. 其次, 用域来限制TP 可执行进程像的权限. 为了对 TP 可执行程序进行保护, 系统把每个 TP 的可执行程序看成一个特殊的 CDI, 但是其完整性保护与普通的 CDI 不同.

定义 **2**　*TP_Objects* 是所有 TP 对应的可执行程序的集合.

定义 **3**　*Type_TPs* 是所有 TP 的可执行程序相应的型的集合.

定义 *CW_TYPES* 和 *OBJECTS* 分别表示 TE 配置中所有型和客体的集合.

定义 **4**　$CW\_TYPES = Type\_CDIs \cup Type\_UDIs \cup Type\_TPs$.

定义 **5**　$OBJECTS = CDIs \cup UDIs \cup TP\_Objects$.

为了获取 *OBJECTS* 中每个客体的型, 定义函数 *type_of_object* 如下.

定义 **6**　$type\_of\_object: OBJECTS \rightarrow CW\_TYPES$.

由于 TP 的动态性可以看成是一个能可信地执行保持或者引入完整性的操作的进程, 为了描述的方便, 在使用 TP 时, 我们将其视为一个进程. 在实现 TE 的系统中, 每个进程都有一个域属性, 表示了进程的逻辑执行空间. 因此, 对于每个 TP, 定义一个相应的域, 它限制了TP 所能执行的操作. 为了表示 TP 相关的域集, 有如下定义.

定义 **7**　*Domain_TPs* 是系统中所有TP 的域组成的集合, *CW_DOMAINS* 是系统中所有域的集合, $Domain\_TPs \subset CW\_DOMAINS$.

为了实现职责隔离, Clark-Wilson 模型要求系统维持 userid-TP 和 TP-CDIs 关系, 用户直接通过操纵 TP 来操作 CDI 数据, 在 Clark-Wilson 模型中以元组 (userid, TP_i, (CDI_1, CDI_2, CDI_3, …)) 的形式反映出来. 本文采用了文献[19] 的处理方式, 在用户和 TP 之间增加一个新的层次——角色 (role) 层, 从而访问元组扩展为: (userid, role, TP_i, (CDI_1, CDI_2, CDI_3, …)).

定义 **8**　*ROLES* 表示系统角色集.

定义 **9**　$domain\_of\_role: ROLES \rightarrow IP(CW\_DOMAINS)$, 表示具有某一角色身份的进程允许进入的域集. 这里 $P(A)$ 表示 A 的幂集.

定义 *OPERATIONS* 表示系统访问模式.

定义 **10**　*OPERATIONS*: 系统访问模式集, 其中包括 *read*, *write*, *exec*, *auto* 和 *signal* 等操作.

为了简化描述, *DDT* 和 *DTT* 表被定义为三元组的集合:

定义 **11**　$DDT = \{(d, t, op) \mid d \in CW\_DOMAINS, t \in CW\_TYPES, op \subseteq OPERATIONS\}$

定义 **12**　$DTT = \{(d_1, d_2, op) \mid d_1, d_2 \in CW\_DOMAINS, op \in OPERATIONS\}$, $op \subseteq \{auto, exec, signal\}$.

定义函数 *tp_execby_role*, 返回具有某个角色身份的进程可以执行的 TP 可执行程序集合.

定义 **13**　$tp\_execby\_role: ROLES \rightarrow IP(TP\_Objects)$.

$\forall r: ROLES \cdot tp\_execby\_role(r) = \{tp\_o \mid tp\_o \in TP\_Objects \wedge (\exists d \in Domain\_TPs \cdot d \in domain\_of\_role(r) \wedge (d, type\_of\_object(tp\_o), \{exec\}) \in DDT)\}$.

此外, 系统中的任务往往由多个执行完整性操作的子任务(TP) 组成, 定义系统任务集为 *TASK*,

定义 **14**　$TASK = \{tsk \mid tsk \in IP(TP\_Objects) \wedge tsk \neq \varnothing\}$, 由于 TP 的可执行程序与其进程像是一一对应的, 为了简化后面的描述, 这里每个 *tsk* 视为 TP 可执行程序的集合.

为了描述具有职责隔离要求的任务, 定义集合 *SOD_TASK*.

定义 **15**　$SOD\_TASK = \{tsk \mid tsk \in TASK\}$, 是有职责隔离要求的任务集合.

SOD_TASK 中的每一个任务都由多个 TP (实际是 TP 的可执行程序) 组成, 由定义 14 有, $\forall tsk: SOD\_TASK.\ tsk \in IP(TP\_Objects)$

3.2　配置规则和约束

基于基本数据类型的定义, 本节给出配置 TE 实现 Clark-Wilson 模型的规则和约束. 为了与 Clark-Wilson 模型的规则区别开, 我们把配置 TE 实现 Clark-Wilson 模型的规则称为配置规则, 约束 (constraint) 主要指为保证 Clark-Wilson 模型的实现, 对 TE 配置所施加的限制性条件.

当为数据客体配置型属性时,必须满足如下约束条件:

约束 **1** $\forall cdi \in CDIs \cdot type\_of\_object(cdi) \in Type\_CDIs$

约束 **2** $\forall udi \in UDIs \cdot type\_of\_object(udi) \in Type\_UDIs$

约束 **3** $Type\_UDIs \cap Type\_CDIs \cap Type\_TPs = \varnothing$

约束 1, 2 和 3 保障了数据客体的型要么属于 *Type_CDIs*, 要么属于 *Type_UDIs*, 但是不能同时属于两者. CDIs 中的数据项的型必须在集合 *Type_CDIs* 中, 而 UDIs 中数据项的型必须在集合 *Type_UDIs* 中.

如前所述, 对TP的保护涉及两个方面: 对 TP 可执行程序的保护和对TP 执行域的限制. 假设每个TP 的可执行程序的程序逻辑是正确的, 为了保证系统运行时 TP 功能的正确性, TP 的可执行程序必须受到系统合理的保护, 因为任何对TP 可执行程序的不恰当修改都将影响TP 进程像的完整性, 从而影响 TP 功能的正确性. 我们为每个TP 的可执行程序定义一个型属性, 且规定 *Type_TPs* 中的每个型与 TP 可执行程序之间是一一对应的关系. 只有可信度最高的系统安全官员(SSO) 才能修改TP 的可执行程序.

约束 **4** 每个 TP 可执行程序的型属性与 TP 可执行程序之间是一一对应的关系,

$\forall tp\_o: TP\_Objects \cdot \exists t\_tp \in Type\_TPs \cdot t\_tp = type\_of\_object(tp\_o)$

$\forall tp\_o_1, tp\_o_2: TP\_Objects \neq tp\_o_1 \neq tp\_o_2 \Rightarrow type\_of\_object(tp\_o_1) \neq type\_of\_object(tp\_o_2)$

约束 **5** 令系统安全官员的角色是 *r_syssec_officer*, 则有

$\forall r \in ROLES, t\_tp \in Type\_TPs \cdot r \neq r\_syssec\_officer \Rightarrow (\forall d \in domain\_of\_role(r) \cdot (d, t\_tp, \{write\}) \notin DDT)$

当一个TP 的可执行程序被执行时, 相应的进程TP 被创建. TP 可以访问的数据项完全由 TP 所在的域决定, 而这又取决于 Clark-Wilson 模型中的元组(TPi, (CDI1, CDI2, …)). 通常, 在考虑数据完整性时, 可以认为TP 对数据客体的操作主要是 read 和 write. 因此, 定义配置规则 1, 给出 TE 对 TP-CDIs 关系的实现.

配置规则 **1** (TP-CDIs 关系) 对于 Clark-Wilson 模型的元组 $(TP, (CDI_1, \ldots, CDI_n))$, 如果 TP 的域属性是 d_tp, 受限数据项 CDI_i 的型为 t_cdi_i(可能不同的 *CDI* 有相同的型), $i = 1, \ldots, n$, 则 $(d\_tp, t\_cdi_i, op) \in DDT$, $i = 1, \ldots, n$, 其中 $op \subseteq \{read, write\}$, 由具体的TP 程序逻辑决定.

例 **1** 一家银行需要统计某一客户在最近三次提取的存款的总额. 令 $deposit_1$, $deposit_2$ 和 $deposit_3$ 分别是最近三次取款的记录项, *sum* 是存放统计结果的记录项, 程序 *add_sum* 是统计程序(TP), 则统计事务可以表示为 $(add\_sum, (deposit_1, deposit_2, deposit_3))$, (add_sum, sum). 令 $type\_of\_object(depositi) = t\_dep$, $i = 1, 2, 3$, $type\_of\_object(sum) = t\_sum$, 良构事务 *add_sum* 的域是 *d_addsum*, 则有

$(d\_addsum, t\_dep, \{read\}) \in DDT$, $(d\_addsum, t\_sum, \{read, write\}) \in DDT$

在 Clark-Wilson 模型中, TP 可以保持或提高数据项的完整性. 一般来说, 只有 write 操作直接影响数据的完整性. Clark-Wilson 模型的规则 C3 要求, TP 对 UDI 的操作结果只有两种, 要么读取 UDI 中的信息, 并把操作结果写入一个 CDI 中, 要么遗弃 UDI. 无论哪种情况, TP 都不能修改 UDI. 这样做主要有两个好处, (1) UDI 数据项通常可信度较低, 不允许TP 修改UDI 可以防止 TP 无意地泄漏商务秘密; (2) 确保了TP 职能的单一性, 简化 TP 的设计.

配置规则 **2** (TP-UDIs 关系)

$\forall t\_udi: Type\_UDIs, d\_tp: Domain\_TP, op \subseteq OPERATIONS \cdot (d\_tp, t\_udi, op) \in DDT \Rightarrow write \notin op$

在现实中, 一个任务通常由多个以正确的次序执行的子任务组成. 比如 TCSEC[14] 对高安全等级操作系统的一个安全要求是, 所有的文档在打印前必须被标记程序(labeler) 标上恰当的安全级. 这个过程可以用标记程序可信管道来实现, 如图 1 所示. 标记程序可信管道包括三类数据项: 用户文档(*user_file*), 被标记的文档(*labeled_file*) 和打印缓冲区中的文档(*printer_buffers*). 通常普通用户文档是一个 UDI, 而 *labeled_file* 和 *printer_buffers* 则是 CDI. 在 *user_file* 被打印输出前, 必须先经过两个 TP 的处理: 标记程序 labeler 和打印假脱机程序 printer_spooler. 可以用可信管道来实现这一信息处理过程. 当实现 Clark-Wilson 模型时, 可信管道用于实现把一个 UDI 或者 CDI 转换到另一个 CDI 的一系列转换过程, 因此, 可信管道是良构事务的一个特例. 不失一般性, 本文描述从一个 CDI 流向另一 CDI 的可信管道.

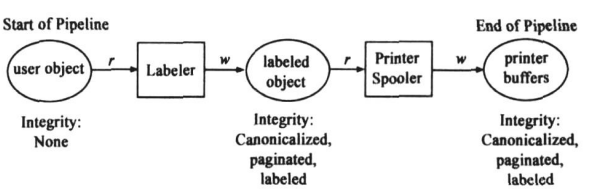

图 1 标记程序可信管道

配置规则 **3** (可信管道) 假设信息从 cdi_0 流到 cdi_n 经过的转换过程为 $tp_1, \ldots, tp_n$, 即,

$$cdi_0 \rightarrow TP_1 \rightarrow cdi_1 \rightarrow TP_2 \rightarrow \cdots \rightarrow TP_n \rightarrow cdi_n$$

假设 TP_i 所处的域为 d_i, $i = 1, \cdots, n$, cdi_j 的型为 t_j, $j = 0, \cdots, n$, 则可信管道可以用域和型表示为:

$$t_0 \rightarrow d_1 \rightarrow t_1 \rightarrow d_2 \rightarrow \cdots \rightarrow d_n \rightarrow t_n \quad (1)$$

其中"$t_i \rightarrow d_{i+1}$"表示域属性为 d_{i+1} 的进程TP允许读型属性为 t_i 的客体, "$d_i \rightarrow t_i$"表示域属性为 d_i 中的进程TP允许写型属性为 t_i 的客体. 则可以如下配置DDT和DTT来实现可信管道:

$\forall 1 \leqslant i \leqslant n, (d_i, t_{i-1}, \{read\}) \in DDT, (d_i, t_i, \{read, write\}) \in DDT$

$\forall 1 \leqslant i \leqslant n-1, (d_i, d_{i+1}, \{signal\}) \in DTT$

其中$(d_i, d_{i+1}, \{signal\})$表示域 d_i 中的进程可以向 d_{i+1} 中的进程发送信号, 以协调管道的执行序列.

可信管道的一个重要性质是不可旁过性[15]. 假设可信管道的形式如(1)所示, 则不可旁过性要求, 信息从型属性为 t_{i-1} 的数据项流到型属性为 t_i 的数据项必须经过域属性为 d_i 的TP. 当配置TE时, 必须保证所有的可信管道是不可旁过的.

约束6　(可信管道不可旁过性)令可信管道如(1)所示, 其中域和型, 以及域间关系满足配置规则3. 为了满足可信管道的不可旁过性, 有:

$\forall d: Domain\_TPs \bullet (d, t_{i-1}, \{read\}) \in DDT \wedge (d, t_i, \{write\}) \in DDT \Rightarrow d = d_i$

例2　在上面的标记程序可信管道的例子中, 数据客体 *user_file*, *labeled_file* 和 *printer_buffer* 的型分别是 *t_usefile*, *t_labeledfile* 和 *t_printerbuffer*. 标记程序 *labeler* 和打印假脱机程序 *print_spooler* 的域分别是 *d_labeler* 和 *d_spooler*, 则标记程序可信管道可以描述为:

$$t\_usefile \rightarrow d\_labeler \rightarrow t\_labeledfile \rightarrow d\_spooler \rightarrow t\_printerbuffer$$

表2和表3给出了相应的DDT和DTT表. 在表2中, *d_user* 中的主体不是一个TP, 因此它可以读, 写和执行型为 *t_usefile* 的UDI文件. 为了保持不可旁过性, 域 *d_spooler* 中的进程不能直接读型为 *t_usefile* 中的文件, 而只能读已被 *d_labeler* 中的进程正确地标记的用户文件. 其中 *r*, *w* 和 *x* 分别是 *read*, *write* 和 *exec* 的缩写, 这一缩写同样适用于后面的情况.

表2　标记管道的DDT表

domains \ types	t_usefile	t_labeledfile	t_printerbuffer
d_user	rwx	—	—
d_labeler	r—	rw—	—
d_spooler	—	r—	rw—

表3　标记管道的DTT表

domains \ domains	d_user	d_labeler	d_spooler
d_user	signal	signal	-
d_labeler	—	—	signal
d_spooler	—	—	—

Clark-Wilson模型实现的另一个完整性保护概念是职责隔离. 职责隔离原则要求一项给定的任务分成多个子任务, 这些子任务必须由几个不同的用户(或者代表不同用户执行的进程)完成. 这保证了把执行一项任务的责任分给多个不同的人, 以防止出现诈骗和共谋的情况.

要注意的是, 虽然可信管道和职责隔离都涉及到多个TP, 但是它们本质上是不同的. 一个可信管道表示的是一个信息流通道, 组成可信管道的多个TP必须以预定的顺序执行, 以便保持或者提高数据的完整性. 因此, 可信管道反映了数据内在一致性保护要求. 而职责隔离主要是防止出现危害商务安全的欺骗和共谋行为, 它反映了数据外在一致性的保护要求.

在用户和TP之间引入角色层后, 元组(*userid*, TP, $(CDI_1, CDI_2, CDI_3, \cdots)$)被扩展成(*userid*, *cw_role*, *TP*, $(CDI_1, CDI_2, CDI_3, \cdots)$), 其中 *cw_role* 是用户 *userid* 执行 *TP* 时需承担的角色. 这样, 系统要维持 *user-role*, *role-TP* 和 *TP-CDIs* 关系. 由于职责隔离更多地强调了通过系统管理功能来实现完整性保护, 而TE只是一种底层强制访问控制机制, 因此, 仅通过TE难以完全地实现职责隔离原则, 需要某些高层抽象机制, 如基于角色的管理[16]来辅助实现职责隔离. 假设在系统管理层正确地设置了 *user-role* 关系, 使得一个用户不能同时承担违反职责隔离要求的所有角色, 那么应当正确配置角色和TP之间的关系, 以保障具有某个角色身份的进程不能执行 *SOD_TASK* 中任一任务的所有子任务(TP). 这里配置TE实现的职责隔离是一种操作职责隔离[17].

配置规则4　Clark-Wilson模型中的每个元组(*User*, TP, $(CDI_1, CDI_2, \cdots)$)被扩展为(*userid*, *cw_role*, TP, $(CDI_1, CDI_2, \cdots)$), 其中 *cw_role* 是用户为了执行TP必须担任的角色. 假设系统管理层机制正确地设置了 *user-role* 关系, *TP-CDIs* 关系用配置规则1设置. 则对于 *role-TP* 关系, 令 *TP* 的可执行程序为 *tp_ob*, 且 $d\_role \in domain\_of\_role(cw\_role)$, $t\_tp = type\_of\_object(tp\_ob)$, 则有$(d_role, t_tp, \{read, exec\}) \in DDT$.

约束7　(操作职责隔离)　具有某个角色身份的进程不能执行有职责隔离要求的任务的所有子任务,

$\forall sod\_task: SOD\_TASK, r: ROLES \bullet d\_role \in domain\_of\_role(r) \wedge \neg(sod\_task \subseteq tp\_execby\_role(d\_role))$

另外, 为了确保TE配置的功能正确性, 除了以上规则和约束外, 还必须满足如下约束

约束8　只有SSO能修改DDT和DTT表, 且具有SSO角色身份的进程不能执行任何TP:

$\forall tp: Type\_TPs, d: Domain\_TPs \bullet d \in domain\_of\_role(sso) \wedge (d, tp, op) \in DDT \Rightarrow exec \notin op$

严格来说, TE 机制的强制访问特性就保证了只有安全管理员能修改 DDT 和 DIT 表. 约束 8 主要是保障 SSO 不能执行任何TP, 这也是为了满足职责隔离的要求[1].

4 对应性分析

4.1 执行规则分析

约束 1– 4 给出了为UDIs, CDIs 和 TP 可执行程序设置型属性时必须满足的条件, 它们构成了整个 TE 配置的基础. 作为一种访问控制机制, TE 主要实现了 Clark-Wilson 模型的执行规则, 其对应性分析如下:

(1) *E*1: 配置规则 1 给出了通过"域-型"关系实现 $(TP_i, (CDI_1, CDI_2, \cdots))$ 关系. 尽管 TE 不实现 IVP, 但是我们假设TP 可执行程序的程序逻辑被正确地设计. 一旦TP 被正式安装, 约束 5 确保TP 可执行程序不会被非授权用户篡改, 从而可以保证系统运行时 TP 的功能正确性. 可信管道扩展了良构事务的概念, 反映了需要多个TP 共同完成一项事务的情况. 根据配置规则 3 和约束 6, TE 配置正确地实现了可信管道.

(2) *E*2: 规则 *E*2 要求系统实现 $(userid, TP, (CDI_1, CDI_2, \cdots))$ 关系, 其本质是保证系统实现职责隔离. 在 TE 实现的过程中, 我们把这一关系扩展为 $(userid, role, TP, (CDI_1, CDI_2, \cdots))$ 关系, 支持角色和 TP 的关联. 在 *TE* 配置中, 假设系统"用户-角色"关系满足职责隔离要求, 那么配置规则 4 和约束 7, 确保了具有某个角色身份的进程不能同时执行具有职责隔离要求的所有 TP.

(3) *E*3: 规则 *E*3 要求执行 *TP* 的用户必须是经过认证的. 本质上, *E*3 的实现必须依赖于系统的标识和鉴别(identification and authentication)机制. 实际上, 任何访问控制机制的正确性都必须基于系统平台的标识与鉴别机制[17], 因此, 在 *TE* 配置中, 不实现这条规则.

(4) *E*4: 规则 *E*4 指出了 Clark-Wilson 模型本质是一种强制访问控制模型. TE 的强制特性规定了只有系统安全管理员才能修改 DDT 和 DIT 表, 而约束 8 使得安全管理员不能执行任何TP.

4.2 验证性规则分析

尽管 TE 并不直接实现验证规则, 但是正如在 2.3 节分析指出, TE 机制本身的特点有助于减少系统在实现 Clark-Wilson 模型时实现验证性规则的必要性, 增强用户对系统正确地实现了 Clark-Wilson 模型的信心. 对于 Clark-Wilson 的规则 *C*1 和 *C*2, 如果每个TP 的可执行程序的程序逻辑被正确地设计且经过了严格的程序测试, 同时约束 5 保证了 TP 的可执行程序在系统运行中不被篡改, 这相应地保证了 TP 在系统运行过程中功能的正确性, 因此不需要在每次 TP 执行时都由一个 IVP 进行操作有效性验证. 系统角色管理机制, 配置规则 4 和约束 7 部分支持了职责隔离的实现. 用域来限制 TP 的执行空间, 满足了 Clark-Wilson 对极小特权原则的要求. 配置规则 2 规定 TP 不能写 UDIs, 从而支持了 *C*5 的实现. 对于 *C*4, 与 *E*3 一样, 需要额外安全机制的保障, 这里的额外机制是指系统审计机制[18], 因此在 TE 的实现中不作相应考虑.

5 结论

尽管 Clark-Wilson 模型实现了完整性保护的三个目标, 但是, 如何在系统中有效地实现它, 一直是安全研究人员探讨的问题. 由于 Clark-Wilson 模型的 IVP 需要外在的干预, 而访问控制机制往往只能直接实现 Clark-Wilson 模型的执行规则, 因此通过访问控制机制完全实现 Clark-Wilson 模型比较困难. 但是, 不同的实现机制实现 Clark-Wilson 模型的灵活性和粒度都不一样, 从而对 IVP 过程的依赖程度也不一样. 因此, 选择哪种机制实现 Clark-Wilson 模型, 不仅取决于对 Clark-Wilson 的支持粒度, 也取决于采用这种机制以后是否有利于增强系统有效控制 TP 功能的可信度, 减少对 IVP 过程的依赖. 本文给出了配置 TE 实现 Clark-Wilson 模型所遵循的规则和约束; 建立了域、型和 Clark-Wilson 模型元素的关系, 并通过配置 DDT 表和 DIT 表来实现 Clark-Wilson 模型的执行规则. 这种基于表的实现方式简化了系统的管理. 把TP 的保护纳入到TE 机制中, 提高了TP 功能的正确性的可信度; 此外, 用域来限制 TP 的权限有利于实现最小特权原则.

在本文的分析中, 我们指出 TE 方式实现 Clark-Wilson 模型的职责隔离时, 必须依赖于系统高层管理机制正确配置了角色和用户的关系, 使得同一个用户承担的角色不能违反职责隔离原则. 本文中, 我们只研究了在系统正确地配置"用户– 角色"关系这一假设的基础上, 配置 TE 实现操作职责隔离. 研究如何组合 RBAC 和 TE 机制实现其他类型的职责隔离, 将是我们下一步的工作.

参考文献:

[1] D D Clark, D R Wilson, A comparison of commercial and military computer security policies[A]. In Proceedings of 1987 IEEE Symposium on Security and Privacy[C]. Oakland, CA: IEEE press, 1987. 184– 194.

[2] T Mayfield, J E Roskos, S R Welke, J M Boone. Integrity in Automated information Systems[R]. U. S. National Computer Security Center, 1991, 79– 91.

[3] T Lee. Using mandatory integrity to enforce 'commercial' security[A]. In Proceedings of 1988 IEEE Symposium on Security and Privacy[C]. Oakland, CA: IEEE press, 1988. 140– 146.

[4] P K Karger. Implementing commercial data integrity with secure capabilities[A]. In Proceedings of 1988 IEEE Symposium on Security and Privacy[C]. Oakland, CA: IEEE press, 1988, 130- 139.

[5] W Polk. Approximating clark wilson' access triples' with basic UNIX controls[A]. In Proceedings of the 4th USENIX UNIX Security Symposium[C]. Santa Clara, CA: USENIX press, 1993. 145- 154.

[6] LIANG B, SHI W C, SUN Y F, SUN B. An approach to enforcing clark-wilson model in role-based access control[J]. Chinese Journal of Electronics, 2004, 13(4): 596- 599.

[7] J Mclean, Security Models. In: J. Marciniak, eds., Encyclopedia of Software Engineering[M]. Wiley Press, 1994.

[8] WEN Hongzi. Research on Commercial Security Policy and its Formal Analysis[D]. Beijing: Institute of Software, the Chinese Academy of Sciences. 2004. (in Chinese with English abstract)

[9] D J Thomsen, J T Haigh. A comparison of type enforcement and unix setuid implementation of well-formed transactions [A]. In Proceedings of 6th Annual Computer Security Applications Conference[C]. Tucson, Arizona: IEEE press, 1990. 304 - 312.

[10] W E Boebert, R Y Kain. A practical alternative to hierarchical integrity policies[A]. In Proceedings of 8th National Computer Security Conference[C]. Gaithersburg, MD, 1985. 18- 27.

[11] T Fine, S E Minear. Assuring distributed trusted mach[A]. In Proceedings of 1993 IEEE Symposium on Security and Privacy [C]. Oakland, CA: IEEE press, 1993. 206- 218.

[12] SMALLEY, S. 2002. Configuring the SELinux policy[R]. NAI Labs Rep. 02-007, available at www.nsa.gov/selinux. June 2002.

[13] M Bishop. Computer Security: Art and Science[M]. Chapter 13, New York: Addisorr Wisley, 2002.

[14] National Computer Security Center. Trusted Computer System Evaluation Criteria[S]. Dept of Defense, No. DOD 5200. 28-STD.

[15] T Fraser, L Badger. Ensuring continuity during dynamic security policy reconfiguration in DTE[A]. In Proceedings of 1998 IEEE Symposium on Security and Privacy[C]. Oakland, CA: IEEE press, 1998. 15- 26.

[16] R Simon, M E Zurko. Separation of duty in role-based environments[A]. In Proceedings of 1997 IEEE Computer Security Foundations Workshop[C]. Washington, DC: IEEE press, 1997. 183- 194.

[17] Jonathon E, Tidswell, Trent Jaeger. Integrated constraints and inheritance in DTAC[A]. In Proceedings of the 5th ACM Symposium on Access Control Models and Technologies[C]. Berlin, Germany: ACM press, 2000, 93- 102.

[18] R S Sandhu, P Samarati. Access control: Principle and practice [J]. IEEE Communication Magazine, 1994, 32(9): 40- 48.

[19] D J Thomsen. Role-Based Application Design and Enforcement[M]. In Database Security, IV: Status and Prospects (eds. Jajodia, S., Landwehr, Carl, E.), Amsterdam: Northr-Holland, 1991. 151- 168.

作者简介:

何建波　男, 1978年生于湖南新田, 博士, 主要研究领域为信息系统安全理论和技术.
E-mail: superhjb@gmail.com

郭　新　女, 1978年生于山东诸城, 硕士, 主要研究领域为网络信息系统安全.

卿斯汉　男, 1939年生于湖南隆回, 研究员, 教授, 博士生导师, 主要研究领域为信息系统安全理论和技术.