一种支持动态调节的最小特权安全策略架构

沈晴霓[1,2], 卿斯汉[1,2], 贺也平[2], 李丽萍[2]
(1. 北京大学软件与微电子学院, 北京 102600; 2. 中国科学院软件研究所, 北京 100080)

摘 要: 最小特权机制可为安全操作系统提供恰当的安全保证级. 本文描述了一种支持动态调节的最小特权安全策略架构, 它结合角色的职责隔离和域的功能隔离特性, 通过一种基于进程上下文—角色、执行域和运行映像的权能控制机制, 将每个进程始终约束在这些上下文允许的最小特权范围内. 本文实例分析了该架构在安胜 OS v4. 0, 一种自主开发的、符合 GB17859-1999 第四级——结构化保护级的安全操作系统中的实现. 结果表明, 它可支持安全操作系统实施动态调节的最小特权控制, 并提供灵活有效的系统.

关键词: 安全操作系统; 安全策略; 最小特权; 权能; 角色; 域
中图分类号: TN309 文献标识码: A 文章编号: 0372-2112 (2006) 10-1803-06

A Framework for Implementing Dynamically Modified Least Privilege Security Policy

SHEN Qing-ni[1,2], QING Si-han[1,2], HE Ye-ping[2], LI Li-ping[2]
(1. *School of Software and Microelectronics, Peking University, Beijing* 102600, *China*;
2. *Institute of Software, Chinese Academy of Sciences, Beijing* 100080, *China*)

Abstract: Least privilege mechanism can provide a reasonable degree of security assurance for secure operating systems. This paper described a framework for implementing dynamically modified least privilege security policy, which combined role's duty separation property and domain's function separation property. Under the control of its new capability mechanism based on a process' s executable image, current role and current domain, it restricted the process to the minimum amount of privileges within these contexts. This paper illustrated its implementation in ANSHENG OS v4.0, a copyrighted secure operating system satisfying all the specified requirements of Criteria class 4, "Structured-Protection", in GB17859-1999 (equally, the B2 level in TCSEC) in China. Thus it demonstrates that this framework can help enforcing dynamically least privilege control on a secure operating system, while still providing a flexible efficient system.

Key words: secure operating system; security policy; least privilege; capability; role; domain

1 引言

特权分离(separation of privilege)和最小特权(least privilege)原则最早由 Saltzer 提出, 它要求系统中每个用户或进程仅拥有执行其授权任务所必需的最小特权集, 而不是拥有所有特权. 在安全操作系统实践中, 管理员为维护系统和安全策略库的正常运行, 经常需要一些超越安全策略检查的行为特权, 这些特权必须以受控的方式使用, 以限制和减小它们被滥用、误用或出错可能导致的系统危害. 因此, TCSEC B2(或我国 GB17859-1999 第四级)以上等级安全操作系统均要求实施最小特权安全策略, 以达到合理的安全保证级[1].

传统 Unix/Linux 系统中实现的是基于用户的特权管理策略, 即: 超级用户和 Setuid 超级用户程序将拥有所有特权, 从而可以执行任何特权行为, 这也是导致系统危害的一个主要根源, 因此大多数安全操作系统不再使用这种方式. 基于 RBAC(Role-Based Access Control)模型[2]角色的最小特权管理策略在安全操作系统中一直受到青睐[3-6], 它可有效实现特权分离和简化管理, 但正如 Hoffman 在文献[7]中指出, 角色仅属于较高的抽象层次, 它的控制粒度和灵活性不足, 不能有效控制每个进程的特权随其使用上下文进行动态调节. 另外, 有一些系统[8,9]尝试完全基于程序来实现, 这可细化特权控制的粒度, 但它主要依赖于对每个程序文件进行恰当的特权指派, 容易导致特权管理的复杂性和风险性增大.

本文提出了可实施动态调节最小特权的安全策略架构(A Framework for Implementing Dynamically Modified Least Privilege Security Policy, DMLP-SP), 它结合 RBAC 模型便于管理的职责隔离特性和 DTE(Domain and Type Enforcement)策略[10]的域隔离保护特性, 提出一种新的 POSIX[11]权能机制, 使它通过程序

收稿日期: 2005-08-08; 修回日期: 2006-06-19
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)

执行(或 exec 调用)和动态域转换引起的进程映像、当前角色和当前域的动态变化,为系统动态调节和灵活管理每个进程的特权最小化提供有效方法.本文实例分析了该策略架构在自主开发的、符合国家标准 GB17859-1999 第四级"结构化保护级"要求的安胜安全操作系统(简称安胜 OS v4.0)中的实现,以及它对系统性能的影响.结果表明,该策略架构能够使系统在效率损失不大的情况下,确保最小特权控制的动态灵活性和安全性.

2 DMLP-SP 多策略结合思想

在安全操作系统中,特权分离原则和最小特权原则都要求将超级用户的特权划分成细粒度且明确定义的权能集合,并使每一个用户主体在其进程链生命周期中仅具有其使用上下文允许和完成其任务所必需的最小权能集.

RBAC 模型将用户与角色绑定,角色与权限绑定,描述的是一个角色如何限制一个用户可获得的操作[7],而用户和角色在系统中只是一种较高层次的抽象概念,真实的活动实体是用户进程(本文简称为主体),所以 RBAC 角色的实现需要一个更低层次访问控制机制的支持.DTE 属于一种较低层次的抽象,它将主体与一个域绑定,域与权限绑定,描述的是一个域如何限制一个主体可获得的操作,特别是它可以通过恰当的域及其权限的定义限制对特权进程的访问.因此,DTE 是 RBAC 的有效补充[7,12~14],二者的结合可以描述一个域如何限制承担一个角色的活动主体可获得的操作,在支持域隔离保护的基础上实现抽象的 RBAC 职责隔离概念.但这种结合实现的控制粒度仅在主体域层次,还不足以灵活控制每个进程的特权状态最小化.

分析表明,POSIX[11]提供了相应的权能机制标准,给出了权能定义的准则,并明确规范了实现最小特权原则的基本语义和实现方法:(1)每个主体应该与其可用的一个权能集绑定;(2)对主体权能的控制粒度应该是程序级的,即每个主体的权能状态应该受其所执行程序权能状态的影响;(3)每个主体的权能状态必须控制在其特定使用的上下文中,而不是限制在一个单一的状态.因此,在 RBAC 和 DTE 结合的基础之上通过 POSIX 权能机制的实现,可以确保每个主体在程序文件层次上进行更细粒度的特权控制.再者,主体的域标识决定其访问权[9,10],所以在主体执行入口程序进入一个新域(动态域转换)时,该主体的访问权将发生动态改变,除主体本身和程序外,如果能对其使用角色和域赋予恰当的特权,则 POSIX 权能机制可通过执行(或调用 exec)程序/可执行文件,在进行动态域转换和进程映像改变的同时,经过调用一个有效的权能遗传算法的检查和运算功能,使系统动态地调节和控制每个主体的权能状态仅处于当前上下文允许的最小特权范围之内.

因此,基于 RBAC,DTE 和 POSIX 相结合的特权控制策略思想可满足以下三个特性:

- 实现管理员之间的职责隔离.通过在用户和权能间增加一个抽象层次—角色,使权能与角色而不是直接与用户关联,不仅可简化授权管理,而且可实现特权分割,将系统任务分给多个管理员共同承担,而不是一个超级用户拥有所有特权.

  实现可信功能之间的隔离.通过在角色与权能之间增加一个抽象层次—域,限定一个角色的用户进程只允许进入/动态转入授权的域运行,使同一角色的不同主体能够依据其执行功能的不同运行于不同的域,保证敏感信息的完整性和防止不安全的信息流.

  保证实现动态的最小特权控制.通过提出一种新的权能机制,基于对每一个主体、程序文件、角色和域进行恰当的权能指派,最后由一个基于程序文件、角色和域的权能遗传算法(见 3.3 节),使主体在执行程序时基于其当前安全上下文变化自动地计算新进程映像主体的权能状态,始终控制在角色和域允许范围之内、程序功能所需的最小特权.

3 DMLP-SP 实施架构描述

安全系统中的基本特权事件主要有两大类[4,13],一类是可操作事件(operational event),是一种受限执行的操作,它需要系统检查主体是否具有所需的操作特权;另一类是超越访问事件(access override event),是一种安全访问事件,它需要系统检查主体是否具有超越所实施访问控制(比如自主访问控制 DAC、强制访问控制 MAC)的访问特权.需要说明的是,每个任务往往需要按一定顺序激发若干个特权事件,所以任务所需的权能与它要激发的事件序列直接相关.在系统中特权事件(记为 E)已知的前提下,本文提出了一种支持 DMLP-SP 安全策略的实施架构.

3.1 授权实体及相互映射关系

DMLP-SP 策略有关的授权实体除了包括特权事件集合 E 之外,还包括权能 C、用户 U、主体 S、角色 R 和任务 T 等.下面介绍这些授权实体及相互之间的映射关系.

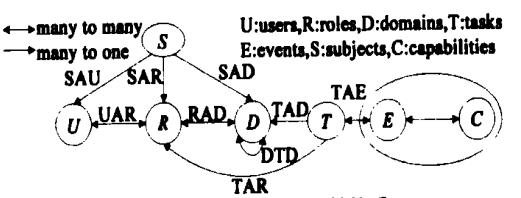

图 1 授权实体之间的映射关系

C 表示系统所定义的权能集.

T 表示系统中用户和管理员使用的所有可信程序(Trusted program)集,每个程序必须拥有正确完成其功能所需的恰当特权(记为 Program-CapState: $I_f \times P_f \times E_f$),其中 I_f, P_f, E_f: 2^C 的含义分别为:

- I_f 可继承权能集(Inheritable set):表示该程序的进程映像能够遗传给其后续进程,并且其前驱进程也拥有的权能.
- P_f 许可权能集(Permitted set):表示该程序的进程映像正确地完成其功能所需的权能,与其前驱进程是否具有这些权能无关.

® E_f 有效权能集(Effective set): 表示该程序的进程像将拥有的有效进程权能集.

一个任务往往需要按一定顺序激发若干特权事件来完成, 同样, 不同的任务可能需要激发同一个事件完成各自的任务. 因此, 任务和事件的映射关系是多对多的(记为 $TAE \subseteq T \times E$). 其中任务所需特权事件集表示为 $task\_required\_events: T \rightarrow 2^E$.

U 表示系统中建立的有效用户帐户集, 每个用户用 *uid* 唯一标识, 并赋予一个允许承担的角色集合. 本架构允许每个用户承担多个角色, 所以用户和角色之间属于多对多的映射关系(记为 $UAR \subseteq U \times R$), 其中用户允许承担的角色集表示为 $user\_assigned\_roles: U \rightarrow 2^R$, 授予指定角色的用户集为 $role\_authorized\_users: R \rightarrow 2^U$.

S 表示用户创建的活动主体集. 每个主体必须拥有一个用户标识、一个角色标识和一个域标识和一个权能状态(记为 $Sulj\_CapState: I_p \times P_p \times E_p$), 其中 I_p, P_p, E_p: 2^C 含义分别为:

® I_p 可继承权能集(Inheritable set): 表示当前进程可遗传给其后续进程的权能集.

® P_p 许可权能集(Permitted set): 表示当前进程允许生效的最大权能集.

® E_p 有效权能集(Effective set): 表示当前进程生效的权能集.

主体和用户之间的映射关系始终是多对一的(记为 $SAU \subseteq S \times U$), 其中指定主体的用户表示为 $subject\_user: S \rightarrow U$. 一个主体表示以一个角色身份运行的用户进程. 为避免角色组合或动态角色转换带来的管理复杂性, 本架构规定: 在一次会话过程中, 每一个主体只允许承担一个角色的职责. 因此, 主体和角色之间的映射关系是多对一的(记为 $SAR \subseteq S \times R$), 其中指定主体的当前角色表示为 $subject\_current\_role: S \rightarrow R$, 当前承担同一指定角色的不同主体集表示为 $role\_activated\_subjects: R \rightarrow 2^S$.

在一次会话期间, 一个主体允许依次运行于不同的域, 仅当它满足条件: (1)主体的当前角色 r 被指派允许进入这些域 D', 即 $D' \subseteq role\_assigned\_domains(r)$(见后); (2)这些域之间的转换关系是策略所允许的, 即 $\exists d, d' \in D', t: T, (d, t, d') \in DTD$(见后). 因此, 主体和域之间的映射关系是多对多的(记为 $SAD \subseteq S \times D$). 但每个主体在某个时刻只允许运行于一个域中, 其中主体当前运行域表示为 $subject\_current\_domain: S \rightarrow D$, 当前运行于指定域的主体集表示为 $domain\_activated\_subjects: D \rightarrow 2^S$.

R 表示策略配置的可信角色(Trusted role)集, 每个角色表示管理用户的特定职责, 负责系统一个方面的管理任务, 须赋予一个允许进入的域集合, 以及拥有恰当的特权(记为 $Role\_CapState: 2^C$). 为保证可信角色职责隔离, 一项任务只允许指派给一个特定的可信角色来负责. 因此, 角色和任务之间的映射关系是多对一的(记为 $TAR \subseteq T \times R$), 其中承担指定任务的角色表示为 $task\_authorized\_role: T \rightarrow R$, 指定角色承担的任务集表示为 $role\_assigned\_tasks: R \rightarrow 2^T$.

此外, 某个时刻, 一个特定角色的主体仅允许在一个特定域运行, 但不同时刻, 它允许运行于不同的域. 这样, 角色和域之间的映射关系是多对多的(记为 $RAD \subseteq R \times D$), 其中角色允许进入的域表示为 $role\_assigned\_domains: R \rightarrow 2^D$, 授予运行于指定域的角色集表示为 $role\_authorized\_domains: R \rightarrow 2^D$.

D 表示策略配置的可信域(Trusted domain)集, 每个域限定一类主体的行为能力, 拥有恰当的特权(记为 $Domain\_CapState: 2^C$), 同时须赋予一个允许的域转换关系集合. 本架构利用 *DTE* 支持的域转换机制, 细化域的功能定义, 并决定每个域允许自动(*auto*)或执行(*exec*)转换进入的域关系. 因此, 域和域之间的这种映射关系是多对多的(记为 $DTD \subseteq D \times T \times D$), 其中 $domains\_transition: D \times T \rightarrow 2^D$ 映射关系表示的是: 运行于一个域中的主体在执行一个入口程序后将允许自动或执行进入的域的集合. 如有域定义:

$domain\ login\_d = (/bin/login), (nwxd \rightarrow base\_t), (exec \rightarrow admin\_d, op\_d, audit\_d)$;

其中域与域之间的映射关系有:

$(login\_d, /bin/login, admin\_d), (login\_d, /bin/login, op\_d), (login\_d, /bin/login, audit\_d)$.

为确保功能之间的隔离, 一项任务只允许在一个特定的域中正确执行. 因此, 任务和域之间的映射关系是多对一的(记为 $TAD \subseteq T \times D$), 其中允许执行指定任务的域表示为 $task\_authorized\_domain: T \rightarrow D$, 指定域的授权任务集表示为 $domain\_assigned\_tasks: D \rightarrow 2^T$.

3.2 限制关系

授权实体之间的映射关系需要进行恰当的限制, 比如在 RBAC 模型中, 限制属于一种重要的特性, 以表达职责隔离的安全需求. 本架构引入了文献[15]中的静态互斥/职责隔离(Static Separation of Duty, SSD)和动态互斥/职责隔离(Dynamic Separation of Duty, DSD)限制关系, 同时也引入了新的域间互斥限制关系, 具体描述为:

动态域互斥/功能隔离 DSF(Dynamic Separation of Function): 若用户 u 以角色 r 创建的主体 s 运行于域 d 中, 而域 d' 与域 d 为运行互斥关系, 则此时不应该有任何以同一用户 u、同一角色 r、但运行于域 d' 的主体 s' 存在, 表达如下:

$$\forall d, d': D, s, s': S,$$
$$d = subject\_current\_domain(s) \wedge d' = subject\_current\_domain(s') \wedge (d, d') \in ED_r$$
$$\Rightarrow subject\_user(s) \neq subject\_user(s') \wedge subject\_current\_role(s) \neq subject\_current\_role(s')$$

其中 $ED_r \subseteq D \times D$ 是一个域运行互斥关系集合.

3.3 基于程序文件、角色和域的权能机制

POSIX 认为, 主体在系统中是一个动态特征很强的对象, 仅为它建立一个固定的、进程链生命周期内一直生效的权能集是不合适的. 实际上, 一个主体在请求执行一个客体(程序文件)时, 最终将调用 exec 系统调用, 在此过程中, 首先检查主体对客体的访问权, 如果访问允许, 下一步将为其创建一个新的进程空间, 并在真正运行之前依据其使用的安全上下文

是否变化决定是否重新计算该主体的权能状态. 因此权能机制必须解决两个问题: 一是确定主体的安全上下文及其权能表示, 二是提出一种有效的权能遗传算法, 保证最小特权控制的有效实施.

® 安全上下文的权能状态

一个主体的安全上下文包括它所执行的程序文件、当前角色和当前运行的域, 依据 3.1, 系统由每个可信角色定义的职责来确定它所要完成的一个任务集合, 由主体域所定义的功能来确定它允许完成的一个特定的任务集合, 因此: 角色和域的权能状态取决于完成这些任务所需的权能, 实际上是这些任务所激发特权事件所需权能之和.

一个任务(或可信程序)的权能状态除了与其所激发的事件序列相关, 还与承担此任务的角色和主体域关联. 因此: 任务可能激发的所有事件所需的权能之和决定该任务的有效集 E_f 和可继承集 I_f, 承担任务的角色和域都允许的权能决定该任务的许可集 P_f.

® 基于程序文件、角色和域的权能遗传算法

算法 **1** 基于程序文件、角色和域的权能遗传算法

```
void compute_creds(s: S, t: T)
{
  ∃ r: R, d: D, r = subject_current_role(s), d = subject_current_domain(s)
  // a check to see whether the domain transition is permitted.
  if d' = domains_transition(d, t) ∧ d' ∈ roles_assigned_domains(r)
  then d = d'   // s will enter domain d'
  // compute the new capability state of s based on the capability state of t, r and d.
01 s.Subj_CapState.I_p = s.Subj_CapState.I_p ∩ t.Program_CapState.I_f
02 s.Subj_CapState.P_p = (t.Program_CapState P_f ∪ s.Subj_CapState.I_p) ∩ r.Role_CapState ∩ d.Domain_CapState
03 s.Subj_CapState.E_p = t.Program_CapState.E_f ∩ s.Subj_CapState.P_p
}
```

Linux 为支持 POSIX 权能机制提供了一种参考的权能遗传算法, 它依据主体的用户标识为 0 或非 0 决定赋予其所有特权或没有任何特权, 不能有效实施最小特权控制. 本架构提出一种新的权能遗传算法(见算法 1), 它支持主体的权能状态随其安全上下文动态变化, 而不再直接与用户标识关联. 本算法基本语义为: (1) 新的可继承集, 是程序文件允许其后续进程继承、执行它的主体也允许其后续进程继承的那些权能; (2) 新的许可集, 包括程序功能所需的最大权能和当前主体允许继承的权能, 但最大不能超出当前角色和当前域允许的权能范围; (3) 新的有效集, 是正确执行程序功能必须生效的权能, 它必须是当前进程许可权能集的子集.

需要说明的是, 当一个主体依据策略执行一个入口程序发生域转换时, 新进程映像的权能状态将取决于新域和入口程序的权能状态, 但是, 当一个主体执行同一程序而策略不允许进行域转换时, 其权能状态仅取决于该程序文件的权能状态.

4 DMLP-SP 在安胜 OS v4.0 中的实现

安胜 OS v4.0 是自主开发并实现的一种高安全等级操作系统, 其安全功能符合 GB17859-1999 第四级(结构化保护级), 安全保证达到 GB/T18336-2001 EAL5 以上的相关要求. DMLP-SP 架构在该系统中的实现不仅要满足职责隔离和功能隔离需求, 而且要确保每个主体动态地建立一个权能上下文环境, 实现有效的最小特权控制, 为系统提供合理的安全保证级. 本节将举例说明该架构如何保证本系统达到以上目标.

4.1 策略配置和授权机制

在安胜 OS v4.0 系统中, DMLP-SP-SP 策略架构的主要授权实体体现为 3 个管理用户、4 个可信角色和 3 个可信域, 它们之间的映射关系和限制, 见表 1 和表 2.

安全管理用户 *sec_u* 负责安全策略库的配置和管理, 承担 *sec_r* 角色; 系统管理用户 *sys_u* 负责系统(包括网络)的配置和管理, 承担 *sys_r* 角色和 *net_r* 角色; 审计管理用户 *adt_u* 负责审计事件的配置和审计记录的管理, 承担 *adt_r* 角色(见表 1). 这样系统特权事件分给了四个可信角色来共同承担, 而不是由一个超级用户来承担所有的职责.

表 **1** 安胜 OS v**4.0** 系统策略配置

user	roles	domains	DTD
sec_u	*sec_r*	*admin_d*, *operate_d*	(*operate_d*, /*sbin*/*dt*, *admin_d*)
sys_u	*sys_r*	*admin_d*, *operate_d*	(*operate_d*, /*sbin*/*dt*, *admin_d*)
	net_r	*admin_d*, *operate_d*	(*operate_d*, /*sbin*/*dt*, *admin_d*)
adt_u	*adt_r*	*audit_d*, *operate_d*	*null*

由于 *sec_r* 和 *adt_r* 分别负责安全策略库和审计日志的维护和管理, 这两个角色的用户都不允许承担其他可信角色的职责, 确保静态职责隔离 SSD(见表 2). *sys_u* 用户允许承担 *sys_r* 和 *net_r* 角色, 但不允许它同时创建这两个角色的主体, 防止可能的网络危害, 比如可信用户以 *net_r* 从网上下载一个非可信模块, 并通过 *sys_r* 将它加载到系统, 可能导致病毒或木马的侵害, 因此它们需实施动态职责隔离 DSD 限制(见表 2).

表 **2** 角色和域之间的限制关系

constrains types	exclusive relationships
SSD	(*sec_r*, *sys_r*), (*sec_r*, *net_r*), (*sec_r*, *adt_r*) (*sys_r*, *adt_r*), (*net_r*, *adt_r*)
DSD	(*net_r*, *sys_r*)
DSF	(*admin_d*, *operate_d*), (*operate_d*, *audit_d*)

admin_d, *audit_d* 域拥有写敏感数据的特权, *operate_d* 域仅拥有执行受限操作的特权, 所以 *admin_d*, *audit_d* 域比 *operate_d* 的可信度高. 本系统的 *sec_r*, *sys_r* 和 *net_r* 授权可进入 *admin_d* 和 *operate_d* 域(见表 1), 但只允许进入 *operate_d* 域的主体执行入口程序 */sbin/dt* 自动转换到 *admin_d* 域, 反之不行, 可防止可信域之间产生不安全的信息流; *adt_r* 授权可进入 *audit_d* 和 *operate_d* 域, 但不允许动态转换, 确保审计行为的可信性及审计记录的完整性(见表 2 的 DSF 限制).

权能机制是 *DMLP_SP* 架构具体实施的关键. 为此, 本系统提供了授权安全上下文权能状态的核心功能和接口, 具体实现方法是: (1) 在 *exec* 系统调用的核心功能 *compute_creds*( ) 中通过该算法(见算法 1) 计算新的进程映像的权能状态; (2) 在内核建立角色、域和程序文件的权能索引; (3) 在相关的核心功能中增加必要的权能检查; (4) 提供有效的接口授权或修改进程本身及其安全上下文的权能状态.

此外, 权能机制的正确实施依赖于权能的恰当指派(即授权管理). 对于一个可信程序, 它的权能状态与其激发的事件序列、指派的角色和运行的域关联. 比如, 本系统设置主客体安全级的命令 *setlevd*, 由 *sec_r* 角色负责执行, 并只能在 *admin_d* 域正确运行. 由于它的执行需调用可操作事件 *setlevel* (所需权能 *CAP_SEC_CONFIG*) 和超越访问事件 *read* 和 *write* (所需权能 *CAP_OVERRIDE_READ* 和 *CAP_OVERRIDE_WRITE*), 所以, 授予 *setlevel* 的有效集 E_f 和可继承集 I_f 为 (*CAP_SEC_CONFIG*, *CAP_OVERRIDE_READ*, *CAP_OVERRIDE_WRITE*), *sec_r* 和 *admin_d* 的权能集合的交授予其许可集 P_f.

此外, 本系统在用户会话期间需要恰当地设置初始进程的权能状态. 比如, 用户在执行 *login* 程序登陆系统时, 须根据该用户将要承担的角色和进入的域, 为其将要创建的主体设置一个恰当的权能状态. 由于角色的职责约束作用, 本系统将角色的权能集合赋予初始进程的许可集 P_p 和可继承集 I_p, 将角色及其要进入域的权能的交集赋予有效集 E_p.

4.2 性能分析

DMLP_SP 对系统性能的影响可以通过相应的系统调用进行测试. 具体方法是: 设计一些程序分别调用每个被测试的系统调用 100,000 次, 并运行该程序 11 次, 为保证数据的准确性, 先去掉第一次测试结果, 然后对余下的 10 次测试结果求平均值.

表 3 为测试结果, 其中 getpid( ) 是最简单的系统调用, DMLP_SP 对它几乎没有影响, 选用它作为比较基数; fork( ) 时权能直接遗传, 除检查它的执行权限外, 性能影响不大; exec( ) 涉及权能运算, 它的影响比 fork( ) 大; read( ) 属于系统中受影响最大的一类系统调用, 除权能检查外, 它需要执行所有访问策略的检查. 测试结果表明, 不同类型的系统调用均有不同程度的效率损失, 但不超过 30%, 属于增加安全之后系统可接受的范围.

表 3　四种典型系统调用的执行效率比较(毫秒)

system calls	standard cost	DMLP_SP cost	overheard
getpid( )	46	46	0%
fork( )	107	116	8%
exec( )	106	129	22%
read( )	311	298	28%

5 相关工作

通过 RBAC 和 DTE 相结合的新机制是由 Hoffman[7] 在 LOCK6 目标系统中引入的, Hoffman 认为在一个安全操作系统中实施 RBAC 必须由 TE(DTE 技术的前身) 支持, 但 Hoffman 只做了较为粗略的描述, 没有在可操作事件和超越访问事件层次上, 明确指出系统中角色的职能与域的职能之间的关系, 也没有明确给出这种控制机制的限制关系, 更没有考虑如何区分特权和常规的访问模式(读, 写和执行). 此后, 文献[12, 14] 引用了 Hoffman 提出的思想, 表达了各自的授权机制, 但它们仅适用于特定的应用系统环境, 与安全操作系统要求不兼容. SELinux 是一个经典安全增强原型系统, 它也采用了 RBAC、TE 和用户身份模型(identity model) 的组合[5, 16], 将 RBAC 提供的管理的便利性与 TE 提供的细粒度的保护很好地结合了起来, 但它对特权行的控制试图完全由恰当的域型指派来保证, 事实证明拥有超越权限的特权[4, 8, 13] 在实际安全系统中是不可避免的问题, 因此它并不实用.

本文将 RBAC、DTE 和 POSIX 权能机制进行有效的结合, 在高安全等级操作系统中实施特权的控制, 不仅支持每个主体级粒度的控制, 而且利用主体域转换及进程映像动态变化的特征, 通过新的权能机制可实现动态调节的最小特权控制, 并提供灵活有效的系统.

6 结束语

本文依据 GB17859-1999 第四级"结构化保护级"及以上级别操作系统的安全需求, 给出了一种支持动态调节的最小特权安全策略架构 DMLP_SP, 它基于 RBAC 模型, 并结合 DTE 策略和 POSIX 权能机制. DMLP_SP 通过 RBAC 角色实现职责隔离, 使每个管理员只能具有其角色职责范围内的特权, 而不是超级用户的所有特权; 借助 DTE 支持的动态域转换机制, 以及 POSIX 权能机制支持的程序级权能控制, 通过一个基于程序文件、角色和域的权能机制, 使系统可动态控制主体的权能状态, 有效地实施最小特权原则. 通过对 DMLP_SP 在安胜 OS v4.0 系统中的实现及其对系统性能的影响分析, 表明该策略架构的灵活性和有效性. 进一步的工作是要对特权与常规访问模式之间的冲突问题进行研究.

参考文献:

[1] M Gasser. Building a Secure Computer System [M]. New York: Van Nostrand Reinhold Co., 1988.

[2] D Ferraiolo, J Cugini, D R Kuhn. Role based access control (RBAC): Features and motivations[A]. Proc of 11th Annual Computer Security Applications Conference[C]. Los Alamitos: IEEE Computer Society Press, 1995. 241 - 248.

[3] Sun. System Administration Guide: Security Services[R]. Part No: 806-4-78-10, Santa Clara: Sun Microsystems Inc., 2002.

[4] Data General. Managing security on DG/UX system[R]. manual 093-701138-09, Westboro: Data General, 2001.

[5] Stephen Smalley, et al. A Security Policy Configuration for the Security Enhanced Linux[R]. Washington: NAI Labs, 2001.

[6] 卿斯汉, 刘文清, 刘海峰. 操作系统安全[M]. 北京: 清华大学出版社, 2004.
Qing S H, Liu W Q, Weng H Z, Liu H F. Operation System Security[M]. Beijing: Tsinghua University Press, 2004. (in Chi-

nese)

[7] J Hoffman. Implementing RBAC on a type enforced system [A], In Proceedings of 13th Annual Computer Security Applications Conference[C]. Los Alamitos: IEEE Computer Society Press, 1997. 158- 163.

[8] National Computer Security Center. Final Evaluation Report TIS Trusted XENIX Version 4. 0[R]. Report No. CSC-EPL-92/001. A, Maryland, : Trusted Information Systems, Inc, 1994.

[9] Cowan C, Beattie S, Kroach-Hartman G, et al. Subdomain: parsimonious server security[A]. In USENIX 14th Systems Administration Conference(LISA)[C]. Berkeley: USENIX Association, 2000. 335- 367.

[10] L Badger, D F Sterne, D L Sherman, K M Walker. A domain and type enforcement UNIX prototype[J]. USENIX Computing Systems, 1996, 9(1): 47- 83.

[11] Draft Standard for Information Technology-Portable Operating System Interface(POSIX)[S]. New York: IEEE, Inc, 1997.

[12] Chandramouli R. A framework for multiple authorization types in a healthcare application system[A]. In Proceedings of the 17th Annual Computer Security Application Conference[C]. Los Alamitos: IEEE Computer Society Press, 2001. 137- 148.

[13] 卿斯汉, 沈晴霓, 等. 一种安全操作系统实现最小特权控制的策略和方法[P]. 中国: 专利受理号 200510011645. 6, 2005.
Qing S H, Shen Q N, et al. The Policy and Method for Implementing Least Privilege Control on Secure Operating Systems [P]. State Intellectual Property Office of China, Accepted No. 200510011645. 6, 2005. (in Chinese)

[14] Qingfeng He, Privacy Enforcement with an Extended Role-Based Access Control Model[R]. TR-2003-09, Raleigh: NCSU Computer Science, 2003.

[15] Kuhn D R. Mutual exclusion as a means of implementing separation of duty requirements in role access control systems [A]. In: Proceedings of the 2nd ACM Workshop on Role-Based Access Control[C]. New York: ACM Press, 1998. 81- 90.

[16] G Zanin, L V Mancini. Towards a formal model for security policies specification and validation in the SELinux system [A]. In Proceedings of the ninth ACM symposium on Access control models and technologies[C]. New York: ACM Press, 2004. 136- 145.

作者简介:

沈晴霓 女, 1970 年出生于江西宜春, 北京大学软件与微电子学院讲师, 中国科学院软件研究所博士, 主要研究领域为信息系统安全和可信计算技术. E-mail: qingnishen@ss.pku.edu.cn

卿斯汉 男, 1939 年出生于湖南邵阳, 中国科学院软件研究所研究员, 博士生导师, 主要研究领域为信息系统安全、安全协议和可信计算技术.

贺也平 男, 1962 年出生于甘肃兰州, 中国科学院软件研究所研究员, 博士生导师, 主要研究领域为密码学、安全协议和信息安全.

李丽萍 女, 1976 年出生于河南济源, 中国科学院软件研究所博士生, 工程师, 主要研究领域为操作系统和网络安全理论和技术.