安全操作系统中的安全管理增强功能*)

郭 玮 茅 兵 谢 立
(南京大学计算机软件新技术国家重点实验室 南京210093)

摘 要 针对一般操作系统在自保护和管理控制机制薄弱方面的缺陷,本文介绍了由笔者等设计开发的一个基于特权分化模型PDM(Privilege-Divided Model),该模型依据最小特权原则,将原有系统中超级用户的特权集合加以细分,由多个管理员取代单个超级用户,同时采用了Capability机制和引进了Voting机制。最后介绍了在安全增强操作系统SoftOS上从管理层到核心层的特权细化实现工作。

关键词 安全操作系统,安全管理,分权,Capability

Enhancement of Security Management in Security Operating System

GUO-Wei MAO-Bing XIE-Li
(State Key Laboratory of Novel Technology of Computer Software, Nanjing University)

Abstract The common operating systems have some shortcomings in self-protecting and control of management. In this paper, a Privilege-Divided Model (PDM) designed by the author and his collegues is introduced in detail. According as the Least-privilege principle, PDM divides the privilege set of the former super-user into fractions and substitute the single super-user as several manager. PDM also adopts the mechanism of Capability and Voting. The inplementation of privilege-divided from management layer to kernel layer in a security enhanced operating system SoftOS will also be presented.

Keywords Operating system, Security management, Privilege-divided, Capability

1. 引言

网络技术和应用的飞速发展正对计算机信息系统产生着显著的影响,而信息安全是影响计算机信息系统发展和应用的重要因素。操作系统是计算机信息系统正常运行的软件基础,它为应用程序提供可靠的运行环境和相应的安全机制。操作系统在计算机系统安全中扮演着极为重要的角色,一方面它直接为用户数据提供各种保护机制,如实现用户数据之间的隔离;另一方面为应用程序提供可靠的运行环境,保证应用程序的各种安全机制正常发挥作用,如禁止数据管理系统之外的应用程序直接操作数据库文件,以防数据库系统的安全保护机制被绕过。

Linux操作系统近年来得到广泛的重视和应用,原因在于其性能稳定、源码公开等优点,目前很多网络服务器都以Linux作为系统平台。但是Linux在安全性方面还存在一定的不足,具体表现为:(1)对自身的完整性保护很弱,容易遭到恶意程序或病毒的破坏;(2)应用程序拥有所代表用户的全部权限,难以防止服务器应用受到攻击控制后滥用权限,破坏系统。(3)自保护和管理控制机制薄弱,违背最小特权原则,给系统带来一定的安全隐患。(4)审计功能简单,只记录系统启动、用户登录等少数事件,缺乏用户资源访问等安全事件的审计处理。

安全管理(Security Management)在操作系统安全中占有非常重要的地位,很多安全事件的发生都存在一定的管理根源,这其中既有操作系统管理机制上的因素,也有用户管理配置上的因素。具体表现在三方面:(1)管理权限过于强大,对保证系统正常运行的关键资源(如系统核心、各类系统工具等)提供不必要的修改权限;(2)提供直接修改系统配置的操作权限,如直接修改(不通过系统管理工具)系统配置文件,使管理工具中实现的安全检查和控制机制落空;(3)管理权限过于集中,单一管理员全权管理系统。

美国国家计算机安全中心NCSC(National Computer Security Center)在其安全标准TCSEC(Trusted Computer Security Evaluation Criteria)中的B级安全操作系统标准中明确提出可信设施管理(Trusted Facility Management)的要求,即TCB应支持操作员和管理员职能的分离(The TCB shall support separate operator and administrator functions)。

本文给出在由笔者等开发的B2级安全操作系统SoftOS上有关安全管理增强功能的基于特权分化模型PDM(Privilege-Divided Model)的设计与实现。

2. 问题分析

2.1 资源的划分

PDM把系统中存在的数据资源根据与系统管理的关系、稳定程度分为三种类型,分别实施不同的控制:

·固定的系统资源(Stable Resource,SR),包括:系统核心文件、系统管理工具等。PDM原则上认为任何用户(包括超级用户root)都不需要也不应该改变这种资源,为避免用户无意

*)本文研究得到国家863高科技项目基金(课题编号2001A144010,课题名称:基于Linux的操作系统安全增强技术的研究与开发)资助。郭 玮 硕士研究生,主要研究领域为安全操作系统;茅 兵 副教授,主要研究领域为分布式计算,安全操作系统;谢 立 教授,博士生导师,主要研究领域为分布式计算,并行系统,安全操作系统。

(或恶意)破坏该类资源,PDM 禁止任何主体修改这些资源。

·动态的系统资源(Dynamic Resource,DR),包括:系统配置、系统管理方面的数据资源,如用户信息文件(/etc/passwd)等。对这类资源的操作实现了绝大多数的系统管理工作,因此对这类资源的读写进行访问控制是 PDM 管理增强的重点。

·一般用户资源(User Resource,UR),包括:一般用户的数据资源。除特别指明为上两类资源外,都默认为一般用户资源。PDM 的系统保护与管理控制模块不对这类资源的访问进行控制。

2.2 安全管理功能增强的解决办法

从近年来的研究来看,操作系统安全管理方面的增强主要针对这两方面:开发自动化或半自动化的辅助管理技术或工具和改进原有操作系统中管理方面的缺陷。

自动化或半自动化的辅助管理技术与工具的作用包括:(1)自动化配置能够提高安全系统对用户错误或疏忽的免疫力;(2)智能化访问控制策略的出现,如基于程序可信度、基于操作危险级别等权限控制策略,这些权限控制策略能够尽量避免用户直接进行权限管理配置,试图以相关推理为基础进行访问控制。(3)漏洞扫描是发现和消除管理配置漏洞的重要措施;(4)审计信息自动化分析可以协助管理员发现和消除系统管理方面存在的安全隐患。

PDM 主要解决的是对原有操作系统中管理方面的缺陷的改进工作。

2.3 最小特权原则

基于 BLP(Bell-La Padula)模型构建的系统通常将超级用户独立于强制访问控制的约束体系之外,并由它来完成系统属性的调整,即在自主访问控制允许的情况下,允许超级用户的所有动作请求。这实际上给强制访问控制系统埋下了一个安全隐患:一旦系统的超级用户被攻破,整个系统将完全暴露在攻击者面前。同理这样也会使安全审计成为一种摆设,入侵者完全可以篡改审计数据,使它的入侵行为隐于无形。现实中还存在这样一种情况:由于系统管理员(通常就是超级用户)对系统并不熟悉而造成误动作从而给系统危害。另外,系统中的一些权限对正常的用户并不常用,尤其对没有计算机专业人员的机构更是如此,这些权限的存在可能成为外界攻击的着手点。

因此需要对超级用户所拥有的特权(privilege)做一定的调整:细化超级用户的权限粒度,并按照权限操纵客体的不同将它们划分为若干类,同时创建若干个子超级用户(sub root),将分类权限分配给各个子 root,同时取消原有系统中的超级用户。这样,既保留了原有系统中的操纵管理功能,又将安全隐患降至最低点。

这里需要引入一个重要的概念——最小特权原则。所谓最小特权原则(Least privilege principle)即每一个用户和进程只拥有最小的必须访问权的集合。进程只应在为完成其任务所必需的那些权限所组成的最小保护域 SPD(smallest protection domain)内执行。当进程的访问需求变化时,该进程就转换其保护域。一个程序无法损坏它不能访问的对象。这一概念较为形式的表述为:

设 P_{root} 为原有超级用户所拥有的全部特权集合;ξ 为 P_{root} 上的一个划分,其秩为 n;P_i 为在划分 ξ 下第 i 个子 root 所拥有的权限集合(i=1..n);那么:

1)$P_{root}=\bigcup_{i=1}^{n} P_i$

2)$\forall$ i,j,0<i,j≤n,i≠j,有 $P_i \cap P_j=\Phi$

3)$\forall$ i,P_i 足以满足某个操作 op_i 的需要,且若存在某个特权集合 P_k 也能满足操作 op_i 的需要,$P_k \neq P_i (i=1..n)$ 那么必有:$P_i \subseteq P_k$

图1给出了系统分权前后的权限示意图。

a. 原有系统的权限系统示意图　　b.PDM 的权限系统示意图

(图中的数字表示部分即为左图中阴影部分的垂直划分)

图1　分权前后的系统权限示意图

3. 系统的设计

3.1 PDM 设计的目标和原则

根据最小特权原则的要求,PDM 所要实现的目标就是使传统 Unix/Linux 系统中的超级用户不再存在,在系统中另外设立若干个子管理员,这些子管理员之间将互不干扰、互不管辖,履行各自的管理职能且各自的管理职能应尽可能的小。这样即使系统的某个子管理员被攻破,入侵者获得的特权集合只是原有超级用户特权集合的一个很小的子集,其破坏能力也就可以被限定在一个较小的范围内。

为了获得有效而完整的划分,需要对特权的范围以及主客体可信程序、配置文件所构成的集合加以考察。PDM 划分可信程序/配置集合的原则是:①完备性:利用这些可信程序可以实现系统的完全管理;②可信集合要尽可能小,尽量避免功能重复的可信程序;③各个管理员的权限集合的大小尽量平衡;④对内核的改动越小越好;⑤同时考虑系统日后的升级及产品化的问题。

划分可信程序/配置集合的方法是:①以影响系统运行的配置文件为中心,这些配置文件全部归入可信集合;②任何使这些配置文件发挥作用的程序归入可信集合;③修改这些配置文件的程序归入到可信集合;④通过一些特殊的系统调用完成了管理功能的程序归入到可信集合。

3.2 各子管理员的权限划分

根据上述目标和原则,PDM 通过以管理工具为中心的系统管理方案来实现系统管理增强,系统设置三个管理员:系统管理员、安全管理员、审计管理员。每个管理员只能运行特定管理工具,特定的资源只能被特定管理工具所操作。该模块在判断操作是否许可时用到的安全上下文(security context)主要保存在统一的列表结构中:

```
PDM_struct {
```

```
    资源/工具标识;
    资源类别:{SR、DR、UR}
    可执行该工具的管理员;
    可操作该资源的管理工具的标识符数组;
}
```

系统资源的安全上下文在系统安装时就已经确定,在系统运行中生成的数据资源(主要指文件资源)默认为 UR 类,不需要管理员进行配置。

各管理员的职能划分如下:

·审计管理员(sysaud):启动、关闭后台日志/审计进程(syslogd/klogd/auditd);浏览日志、审计文件;日志/审计的日常维护;对日志/审计的检查和备份。

·安全管理员(syssec):从系统中添加、删除用户账号;从系统中添加、删除组账号;设置用户/用户组的口令;修改用户账号属性和组属性;设置系统强制访问控制等安全策略的开关;添加用户的安全标记,获得、改变原有用户的安全标记;覆盖强制访问控制检查,将需要逆向传递的文件复制到公共目录。

·系统管理员(sysadm):设备及模块管理程序功能;添加、卸载设备驱动程序和文件系统模块,网络设备模块;设置设备属性;启动、停止设备;对普通用户使用设备的授权;系统核心功能维护,维护系统核心的镜像文件,如编译系统核心、更改系统核心,维护系统引导扇区等。

需要注意的是:

1)各管理员的行动仍应受安全操作系统访问控制的约束,但须根据需要对访问控制模块进行相应的调整。

2)除上述管理员之外的其它用户身份为一般用户,它们不具备任何特权,只拥有对少数必备工具的使用和在访问控制约束下使用资源的权力。

3)通过不同的操作环境和操作界面实现对各管理员及一般用户权限的限定。

3.3 Capability

"Capability"简单地说就是一个标识(Token),用于表示一个主体是否可以对一个客体实施某种动作。比如一个文件描述符(File Descriptor)就是一种 Capability:在打开文件创建一个文件描述符时,就确定了请求的某些权力,如读、写和执行。当执行一个请求时,操作系统内核就将文件描述符作为一个索引在某个权限表中查找相应的字段,以确定请求是否被通过。这是一种十分有效的权限检查手段。Linux 系统中,POSIX Capability 的概念有些不同,它实际上是对超级用户的特权(Privilege)进行划分,形成若干个特权子集。这个概念首次出现在 POSIX 1003.1e 的标准草案中。因此在 PDM 中,可以使用 Capability 机制来协助特权分化完成安全管理增强。

一般说来,Capability 上的操作有拷贝、Capability 授权、修改 Capability 以及收回 Capability 等几种。Capability 通常体现在属主对某个客体的权限管理。在当今的各种操作系统中,都以某种形式表示了 Capability 的存在。

3.3.1 *进程的 Capability* 一个进程的 Capability 属性集可以分为3类:

·可继承集合 I(Inheritable):指由当前进程执行的程序所继承下来的 Capability,当执行 fork()和 clone()而产生新的进程时,子进程/线程会完全继承父进程的 Capability 集合;

·允许集合 P(Permitted):表示该进程可使用的 Capability 有哪些,P Capability Set 可以在 exec()系统调用执行时刻被 I Capability Set 所屏蔽;

·有效集合 E(Effective):标识允许集合 P 中的 Capability 中有哪些当前是有效的。

一个进程可以动态地修改其 E Capability Set,以适应不同情况下的需求。一个进程在 E Capability Set 中存在某个 Capability,则该 Capability 必须同时出现在 P Capability Set;而出现在 P Capability Set 中却不出现在 E Capability Set 中的 Capabitlity 则是当前暂时被禁止的 Capability。

每一种 Capability 都以位(Bit)的形式记录下来。当一个进程试图请求特权操作时,系统内核都会检查该进程的相应 Capability 位,而不是通常的检查 UID/EUID 是否为0。例如一个进程准备执行重启(reboot)指令时,Linux 内核就会检查 CAP_SYS_BOOT 位是否出现在 E Capability Set 中。

3.3.2 *文件的 Capability* 文件也有 Capability 属性,但考虑到只有可执行文件才可能成为行为的主体,所以才具备这种属性。

为了避免将这些属性与进程的概念相混淆,标准又定义了另外三种名称:

·允许集 A Set(Allowed Set):表明可执行程序可以从执行进程处获得哪些 Capability。这意味着在执行 exec()系统调用执行程序时,先屏蔽 I Set,然后将新执行程序允许接受的 Capability 屏蔽。

·强制集 F Set(Forced Set):是程序在执行时刻被设置的 Capability,即将属主所对应的 Capability 这个完整的权力集细分,只设置自己所需要的部分。这种思想恰巧是特权最小原则的具体体现,同时也是我们选择 Capability 作为分权系统的基础设施的主要原因。

·有效集 E Set(Effective Set):表明了在进程 P Set 中的哪些 Capability 必须被转移至新进程的 E Set 中。

3.3.3 *规则* 在进程运行的过程中,所有涉及 Capability 的变化都应该遵循如下几条规则,以保证权限不被滥用。

I. pI' = pI

II. pP' = fP | (fI & pI)

III. pE' = pP' & fE

(I=Inheritable P=Permitted E=Effective
p=Process f=File)

(' 表示执行之后的状态结果,以上的操作是集合操作)

3.4 表决(Voting)机制

由于在 PDM 的设计过程中取消了原有的超级用户,将其特权分散到几个管理员手中去,这就必然引起一些问题:1)一旦某管理员因特殊原因(如口令遗忘、突然辞职等)无法实施系统管理,整个系统的正常管理都要受到影响;2)分权后的管理员在使用他的某些特权时有可能有机会越权,从而使分权的目的落空。譬如说系统管理员有权使用 insmod 这个命令,但由此命令完全可以插入新的模块(module)以绕过 PDM 模块,从而获得所想拥有的权力。为此我们引入表决机制和特权管理模式。

在特权管理模式下,能够对管理员进行日常管理,如增加和删除管理用户,修改管理的口令。为安全起见,在管理模式下,并不能对系统进行一般性管理,只能通过添加相应的管理简介完成。

特权管理模式对应一个可信度高的管理工具,系统赋予该管理工具一定的管理特权,用户要启动该管理工具,需要同

时认证多个管理员，只有所有或大多数管理员(取决于具体策略)都认证通过，才能进行特权管理，这类似于现实生活中的表决系统。

4. 系统的实现

安全操作系统 SoftOS 的 PDM 模块主要通过两个方面实现安全管理功能的增强：一方面是上层的管理程序，对不同的管理员我们启动不同的管理程序，由管理程序去控制对资源的配置访问(包括 Daemon 进程的启停等)。另一方面在核心层，利用 Capability 机制和修改有关系统调用来实现。

4.1 对 shell 的修改

为各管理员(审计管理员 sysaud、安全管理员 syssec、系统管理员 sysadm)和一般用户设计各自的管理程序 shell，通过修改/etc/passwd 文件将不同身份的用户指向不同的 shell。这样在不同身份的用户 login 后就可以为他启动相应的 shell 以限制他的权力。同时为各不同身份的用户设置符合其身份的安全上下文结构列表，各 shell 在执行操作时，首先根据该操作是否在相应用户的安全上下文结构列表中，若否，拒绝该操作；若在列表中，还需进一步根据安全操作系统本身的访问控制策略(DAC、MAC、ACL 等)进行进一步的判断。

管理程序 shell 是我们根据具体的管理员的职责和相应的被控客体的集合开发的，假定它是可信的。具体流程如图2。

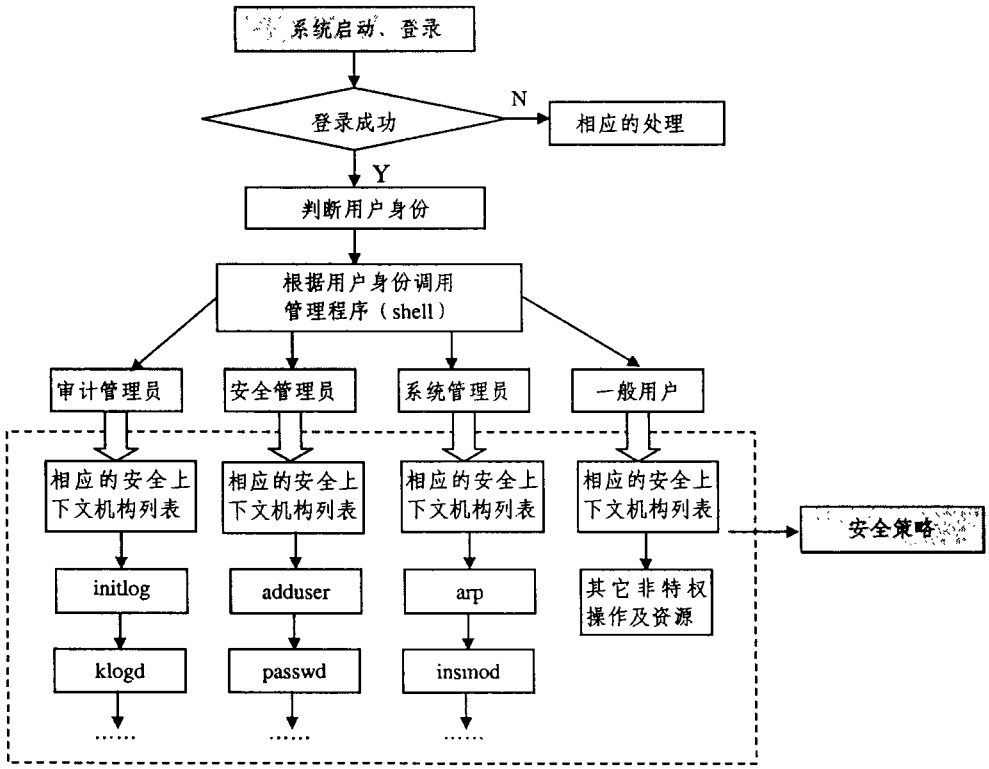

图2 PDM 的管理层(shell)控制

4.2 对核心的修改

在核心层，PDM 的工作是修改系统调用，建立有关系统访问控制规则表；利用核心提供的 capability 机制。

·对系统调用的修改：进程对配置文件的访问主要体现在对 open()以及 chown()、chmod()调用族的调用上；管理命令的调用以及后台进程的启动主要体现在 exec()系统调用族的调用上；对于一般的系统调用，将按照系统原有的控制策略进行控制；对设备的访问控制主要体现在系统调用 ioctl()的各种参数的控制，这种控制主要由相应的规则来描述；/proc 文件系统是由核心实现的一种"虚拟"文件系统，主要是用来使应用程序以访问文件的方式访问核心数据。

·对于 Capability 的授权管理，需要借用系统调用 setcap()；但这需要该进程具备 CAP_SETPCAP 的 Capability。所以在内核中，我们要针对不同的管理员进行不同的 capability 授权，然后在涉及访问控制的时候调用 capable()函数进行 capability 判断，仅当与访问控制策略相符时才允许相应的访问。

由于对核心的修改涉及较广，比较容易产生不可预计的后果，并且过多的核心操作会增加系统开销，因此 PDM 尽量减少对核心的修改(前文 PDM 划分可信程序/配置集合的原则的第④点)，主要是用作上层管理工具的补充。这样既降低了系统的隐患也提高了系统的效率。

5. 与相关工作的比较

当前国内外的各种安全操作系统非常多，它们在安全管理增强功能上所做的工作也有所不同，例如：1986 和1987 年 IBM 公司的 V. D. Gligor 等发表的安全 Xenix 系统的设计与开发成果中，对系统的管理功能进行分割，设立可信系统程序员、系统安全管理员、系统帐户管理员和系统安全审计员等特权用户，通过不同的操作环境和操作界面限定各特权用户的特权。但其可信系统程序员具有超级用户的全部特权，只能在系统的维护模式下工作，负责整个系统的配置和维护；美国国家安全局(National Security Agency，NSA)资助和主持开放的 SELinux，通过角色设置实现了多管理员；罗马大学主持开放的 REMUS 对管理员权限做了限制，一定程度上实现了分权。德国汉堡大学等在 Linux 上实现的一个支持灵活安全策略的安全系统 RSBAC 通过对 Linux 内核的修改加入基于角色的控制机制实现了由多个管理员共同管理系统。

SoftOS 的 PDM 模块优点在于：1)多个管理员共同管理系统，相互牵制，更加符合最小特权的安全原则；2)双重安全检查，系统管理工具(属于固定的系统资源，其完整性能够得到保证)是由操作系统厂商提供，一般实现了相应的安全检查机制，此外核心在安全管理工具运行时还要进行必要的权限检查。3)表决系统有效解决个别管理因特殊原因无法实施系统管理的困难。4)加入 PDM 后对系统性能影响不大，产品化容易。

结束语 安全操作系统 SoftOS 的基于特权分化模型 PDM 能够较好地解决操作系统自保护和管理控制机制薄弱方面的缺陷，对于防止由于权力过于集中而引起的有意或无意的破坏性操作有着较好的防范作用，提高了系统的容错性。大量的测试及比较表明，加入 PDM 模块后的 SoftOS 在安全管理增强功能方面，已经达到较高水平。

参 考 文 献

1 Olson I M, et. al. Informatin Security Policy, Information Security: An Integrated Collection of Essays Edited by Marshall D. Abrams, et al. IEEE Computer Society Press, 1995

2 IEEE, IEEE POSIX. 1e Draft Standard for Information Technology: Portable Operating System Interface (POSIX): Protection, Audit, and Control Interface, IEEE Computer Society Press

3 Department of Defence, America. TRUSTED COMPUTER SYSTEM EVLUATION CRITERIA CSC-STD-001-83, Aug. 1993

4 Acharya A, Raje M, MAPbox: Using Parameterized Behavior Classes to Confine Untrusted Applications. In: 9th USENIX Security Symposium, Aug. 2000

5 LaPadula L J, Bell D E. Secure Computer Systems: Mathematical Foundations: [MITRE Technical Report 2547]. Volume Ⅰ, 1973

6 O'Shea G. Security in Computer Operating System. NCC Blackwell Limited. 1991

7 Ware W H. Security Controls for Computer Systems: Report of Defense Science Board Task Force on Computer Security, R-609-1. Santa Monica CA. Rand Corp. 1970

8 Wright C, Cowan C, Linux Security Modules: General Security Support for the Linux Kernel. In: 11th USENIX Security Symposium, San Fransisco 2002

9 http://www.nsa.org/selinux/

(上接第164页)

定位。由于共享库的代码是位置无关(PIC)的，内部跳转只使用相对地址，全局函数和数据的绝对地址则是由装载器计算并填入 GOT 表中，然后代码通过该表来间接地引用所需地址。因而我们只需要重定位代码中的相对地址。

3.4 重复前三步操作直到裁剪不能继续进行为止

同2.3节一样，重复前三步操作直到所有共享库的尺寸不再减小，裁剪才算全部完成。

3.5 两种技术裁剪效果的比较

由于 libc 通常是文件系统中最大的共享库，不经任何裁剪一般大小都有1.2M 字节以上，比 Linux 内核还要大得多。所以要是能对 libc 进行一定的裁剪，整个系统的尺寸的减小就比较显著。我们就以对 libc 的裁剪效果比较两种技术的优劣。

从图4中我们可以看到，我们提出的函数级裁剪技术的裁剪效果比原来的目标文件级裁剪技术明显要好，在应用程序数量不多的时候，用前者裁剪的 libc 大小比后者要小200多 kb；在应用较多的时候也要小100多 kb。而在总体上，用我们的技术可以将 libc 裁剪掉500～800kb。由此可见，该技术的裁剪效果还是非常明显的。

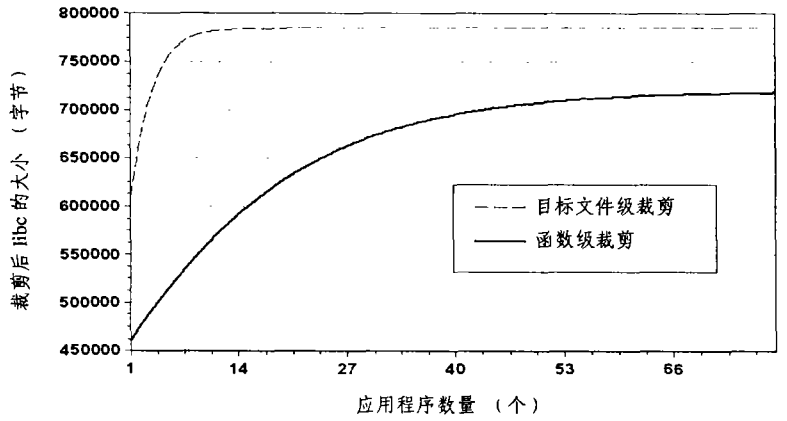

图4 两种技术的裁剪效果比较

小结 本文提出并实现的函数级的共享库裁剪技术能将库中大部分冗余代码裁剪掉，裁剪效率比原来的目标文件级裁剪技术有了很大提高。但是它还是有许多不足，包括：要求库的源码编写比较规范；不同体系结构需要有不同的处理等。但毕竟库裁剪领域才发展不久，还不是很成熟。经过对该技术长时间的测试，相信我们能够弥补它的不足，使它能够在嵌入式 Linux 领域广泛使用。

附录：本文用到的记号

Ψ：将要被裁剪的共享库；Σ：Ψ 中被用到的导出符号的集合；$\mathscr{R}$，$\mathscr{R}'$：Ψ 中各函数块的依赖关系图；F_i：Ψ 的某个函数块；N_i：F_i 在 $\mathscr{R}$ 上的对应结点；Φ，Φ'：$\mathscr{R}$ 和 $\mathscr{R}'$ 上被标记的结点集合。

参 考 文 献

1 Bird T, Right-Sizing Linux: Selective Reduction of Linux for Embedded Devices. Embedded Systems Conference, 2000

2 Levine J R. Linkers and Loaders. Morgan-Kauffman, 1999

3 Executable and Linking Format Spec v1.2. TIS Committee, 1995

4 Aho A V, Sethi R, Jeffrey D. Ullman, Compilers - Principles, Techniques, and Tools. Addison-Wesley, 1986

5 Stevens W R. Advanced Programming in the Unix Environment. Addison-Wesley, 1992

6 路丝林，朱珂等译. David Rusling, Linux 编程白皮书. 机械工业出版社，2000