基于微核心的国产操作系统安全性研究*

李 军 孙玉方

(中国科学院软件研究所 北京 100080)

摘要 本文描述基于微核心设计思想的国产操作系统 COSIX V2.0 的安全性设计. 通过对基于微核心的多服务器体系结构的讨论,确立了系统安全性的设计方案. 最后,是对一些值得探索的问题的简述.

关键词 微核心,多服务器,B1 级安全,自主型存取控制,强制型存取控制.

(1)几个计算机安全概念[1,2]

主体和客体(Subject & Object):计算机中存在大量涉及安全的操作. 凡是实施操作的称作主体,而被操作的对象称为客体.

安全策略(Security Policy):由一组规则组成,规定进行存取控制应当遵循的约束条件.

可信计算基 TCB(trusted computing base):是计算机系统内保护机制的统称,包括硬件、固件和软件中所有与实施安全有关的部分.

自主型存取控制 DAC(discretionary access control):对于系统中客体的安全,由客体的用户主或具有指定特权的用户来制定,主要是规定别的用户以怎样的方式访问该客体.

强制型存取控制 MAC(mandatory access control):对于系统中客体的安全,由系统确定一个主体能否访问一个客体.

可信通路(Trusted Path):一个用户与 TCB 可以直接交互的通路,而不是通过不可信程序.

(2)TCSEC(trusted computer system evaluation criteria)

TCSEC 也叫"橙皮书",是美国国防部于 1983 年 8 月颁布的迄今评估计算机安全最有名的一个标准. TCSEC 将安全保护分成 4 等,每等又包含 1 个或多个级别,如图 1 所示.

(3)安全模型[3,4]

构造一个形式化的安全模型并证明其正确,并将之用于一个系统的设计当中,可以使得一个系统的安全性得到最大限度的保证. 随着对计算机安全研究的重视,出现了丰富多采的安全模型. 对一些典型的安全模型进行分析研究,并适当应用于系统的安全性设计,可以使

* 本文基于"八五"国家重大科技项目 85-712-03-05,"国产系统软件开发"中的专题"文件系统开发". 作者李军,1970 年生,实习研究员,主要研究领域为操作系统,计算机安全,计算机网络. 孙玉方,1947 年生,研究员,博士导师,主要研究领域为计算机软件,中文信息处理,信息处理标准.

本文通讯联系人:李军,北京 100080,中国科学院软件研究所

本文 1996-02-01 收到修改稿

D	最低安全	D
C	自主型保护	C1 C2
B	强制型保护	B1 B2 B3
A	验证型保护	A1 A1以上

增强

安全级按D，C1，C2，B1，B2，B3，A1，A1以上而渐次增强

图1 TCSEC安全级

安全结构清晰，最大限度地避免安全“盲点”.

COSIX V2.0的设计，考虑了有穷状态机模型、存取矩阵模型、Bell&Lapadula(BLP)模型等安全模型的特点，并应用到系统的实际设计当中.典型地，强制型存取控制安全策略就是来源于BLP安全模型.本文重点是讨论COSIX V2.0的总体设计和实现，对安全模型不作具体的讨论.

(4)COSIX V2.0的安全性的总体目标

COSIX V2.0的安全性目标定为高于B1级.根据TCSEC标准及现有条件，B1级安全级主要应包括如下安全功能：标识与鉴别(安全登录)；自主型存取控制；强制型存取控制；特权管理；审计；可信通路.

COSIX V2.0实现了以上几项安全功能，也就满足了TCSEC标准的B1级需求，并部分实现B2级特色，从而最终达到高于B1安全级的目标.

1 COSIX V2.0的体系结构

COSIX V2.0(如图2所示)的设计采用分层实现法.它采用微核心技术[5]，将控制基本硬件等资源管理放入一个小型内核，即微内核中实现，而将操作系统对逻辑对象(如进程、文件等)的管理以微核心为基础实现，不直接与硬件打交道.在COSIX V2.0中实现为多个服务器，即COSIX V2.0采用基于微核心的多服务器体系结构.[6]

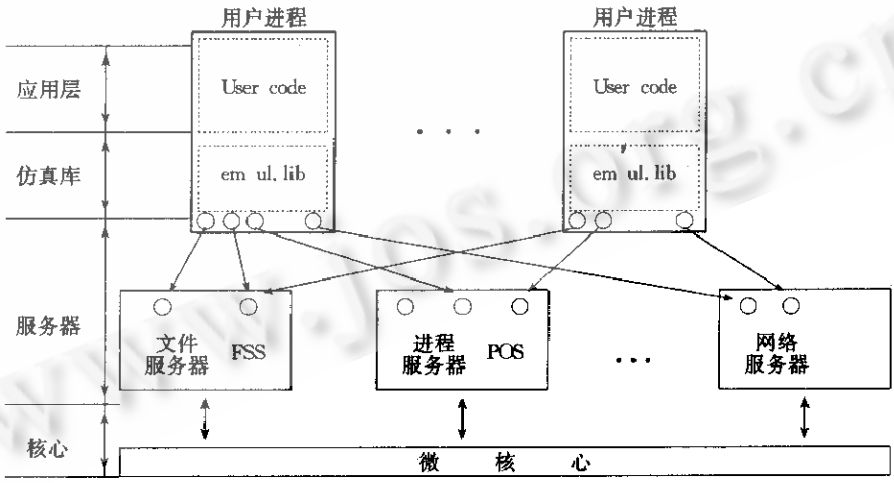

图2 COSIX V2.0的体系结构

COSIX V2.0将操作系统最基本的功能放入微核心中，通过核心管理将机器各类资源抽象成一组对象，包括任务、线程、端口、存储对象、设备等，并提供对对象的操作和管理机制，特别是使对象间通过端口发送消息，实现相互间的服务，供核心之外的应用程序使用.[7]

这样的多服务器系统结构由于其层次分明而便于设计，由于其服务器不直接与硬件打交道而便于移植，由于其每个服务器可以单独重构而便于改造、扩充与维护.

在COSIX V2.0中,实现了文件系统服务器FSS(file system server)[8],进程服务器POS(process server)以及其它服务器.而每一个用户任务在执行时还要将仿真库[9]DML(dynamic mapped library)链入其运行空间,当一个用户发出一个系统调用(如read)时,陷入微核心,微核心判断自身不能完成对它的处理,就利用系统调用改向机制,调用仿真库中的相应入口.仿真库根据所请求的服务类型,再请求FSS或POS为其服务,协助其完成相应功能.FSS或POS执行了相应动作后,将结果以IPC形式返回给仿真库,仿真库将结果直接返回给同一用户空间的用户程序.某些情况下,DML不必请求FSS或POS的服务,而直接利用其内部缓存的系统信息直接实现.COSIX利用仿真库实现了与传统UNIX的二进制兼容.

2　COSIX V2.0安全性的总体设计

2.1　COSIX V2.0安全性总体结构

操作系统应当控制任何主体对任何客体的访问.在COSIX V2.0中,用户任务是安全主体,用户任务可以通过DML机制请求POS或FSS对它的服务,包括打开文件、创建目录、访问设备、安装一个文件卷、创建或设置信号量、共享内存或消息结构、向另一个用户任务发信号等等.这里的文件(含目录、设备、管道)、文件卷、IPC客体(即消息、信号量和共享内存)以及用户任务自身都是安全控制的客体,在COSIX V2.0中,对于任何主体对客体的访问都要实现自主型访问控制、强制型访问控制、特权管理和有选择的审计.

由于COSIX V2.0采用多服务器体系结构,对以上各类客体的管理并不集中在一个服务器内,比如对IPC客体和用户任务(即进程)的管理主要由POS负责,而对文件类型的客体则由FSS负责.在POS和FSS内分别有相应的安全管理,它们对客体的管理有相当多的类似之处,但还是分别放在相应的服务器内来实现.标识与鉴别、审计、可信通路都分别由专门的一个任务来实现.图3给出了COSIX V2.0的安全性设计总体结构,它们构成了系统的TCB.

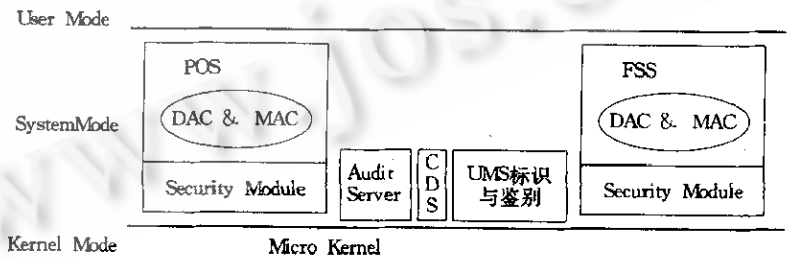

图3 COSIX V2.0的安全性设计总体结构

2.2　COSIX V2.0安全性功能

2.2.1　标识与鉴别

它用于确保只有合法用户才能进入系统使用资源.由用户管理服务器(UMS)来实现.UMS检查用户注册名和口令,赋予用户进程uid、gid,并检查用户安全级、计算特权集、赋予用户进程安全级和特权集标识(用户进程所执行的程序由/etc/passwd获得).

它的具体实现是:在/etc/security目录下建立用户的安全文件,它与/etc/passwd相对

应,表明系统中每个用户的安全级范围及特权集.当用户注册时,可选择安全级;当UMS判定所输入的安全级不在安全文件所规定的安全级范围内则拒绝该次注册.若用户没有选择安全级,则从/etc/default/login中取出默认安全级.当UMS确认了用户的注册名、口令和安全级后,便向注册进程返回消息,认可用户注册.以后该用户的进程则具有该次注册时获取的安全信息.其中uid,gid等用于自主型存取控制,安全级用于强制型存取控制,特权集则用于特权管理.

标识与鉴别对于系统的安全性具有重要意义,因为对系统中任何客体的访问控制是否允许,都是通过查看主体的安全属性信息,而这些信息又主要是在用户注册时的标识与鉴别过程中获得的.假设这一过程出现破绽,那么不论如何进行其它几种安全功能的设计,系统的安全都是脆弱的.

2.2.2 自主型&强制型存取控制

自主型存取控制(DAC)与强制型存取控制(MAC)是要实现的2个最重要的安全策略.

自主型存取控制(DAC)既兼容传统UNIX的owner-group-other三级存取控制机制,同时又引入存取控制表ACL(access control list),使自主型存取控制的粒度可以细到每个具体用户.一个用户可以规定它所拥有的客体(典型的如文件)的ACL为如图4所示.

第1部分	uid1	*	用户uid1对该客体的读、写、执行权限
	.....	.....	.....
第2部分	uidn	*	用户uidn对该客体的读、写、执行权限
	*	gid1	用户组gid1对该客体的读、写、执行权限
	.....	.....	.....
第3部分	*	gidm	用户组gidm对该客体的读、写、执行权限
	*	*	若一个用户或其所在的用户组未在前面列出,那么,这里为其对此客体的读、写、执行权限

图4 ACI表

强制型存取控制(MAC)通过赋予系统中的安全主体和客体以安全级,并在主体对客体进行存取访问时对安全级进行比较,来判断是否允许存取.判断的依据主要来源于BLP模型中规定的安全状态所应满足的2个特性——简单安全特性和*-特性,其中规定:主体对客体有写权限,则客体安全级至少和主体的当前安全级一样高;主体对客体有读权限,则客体安全级不会比主体当前安全级高.

2.2.3 特权管理

TCSEC标准对于B2级安全性要求实现最小特权管理,即系统中每个任务或进程只具有完成其功能所需具有的最小特权.

COSIX V2.0亦采用了特权管理,用户成功注册后,其进程便具有相应特权,系统将对特权的赋值、继承和传递进行控制,同时亦保证带有特权的主体在访问客体时考虑其特权因素.

2.2.4 审 计

正如从军事安全管理制度抽象出BLP模型并应用于操作系统的安全性设计当中一样,审计也是将现实社会中的审计机制引入到计算机系统中来,监视和记录系统的活动,包括对指定用户、指定事件、指定时间进行审计,获取审计报告等.

2.2.5　可信通路功能

本功能为 TCSEC B2 级所要求,用于对多用户状态下模仿 login 或 su 的特洛伊木马的防范.这主要是通过改造终端驱动程序,当用户敲入一特定组合键时,产生中断,由中断处理程序进行注册检测,由图 3 所示的 CDS 实现.

2.3　由安全核思想得到的启发

安全核是和 BLP 模型、引用监控器、强制存取安全策略等一块儿诞生的一个重要的安全概念.可以这样看待它们的关系:安全核是实现引用监控器的硬件和软件,引用监控器采用 BLP 模型(和/或其它安全模型)作为存取控制的框架,而强制存取控制等安全策略则在 BLP 模型中得到体现,如图 5 所示.

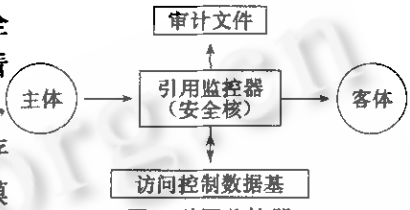

图5　引用监控器

安全核的最重要的 3 个设计原则是:(1)自身是安全的;(2)任一主体对客体的访问都必须经过安全核;(3)需小而精,便于进行安全性验证.

COSIX V2.0 以高于 B1 级安全性为目标,本身并不需要进行类似 A1 级高安全性的形式化验证.但既然 COSIX 也要实现较高的安全性,不妨把微内核连同多个服务器(COSIX 的 TCB)整个地类同于安全核,那么,如果认为 COSIX V2.0 是高于 B1 级安全的,那么也必须至少要满足 2 点:

(1)COSIX V2.0 自身是安全的,是不可被窜改的;

(2)任一主体对客体的访问都必须通过 COSIX V2.0 的控制.

具体而言,对文件型客体的访问必须要经过 FSS 的安全检查,对 IPC 客体和用户进程的访问必须要经过 POS 的安全模块检查.

对于第 1 点:COSIX V2.0 自身的安全性.

我们可以利用有穷状态机模型的概念在这里给一个简单的非形式化的说明.假设系统从一开始就运行得很好,它通过启动微核心、基本服务器,到加载 FSS,POS 及 UMS,CDS,直至到提示用户注册,都是一个稳定无害的过程,那么我们可以初步认为除非上述几个部件(如 FSS)已被非法替换成另一个部件(如 FSS′)或作了改动(也叫 FSS′),那么系统这时处于安全状态.这种替换或改动又只有系统管理员才可以进行,而所有安全操作系统的一个前提就是对系统管理员给予足够的信任(比如在 SunOS 下就可以在启动前由系统管理员启动另一个 UNIX 核心文件而不是默认的/vmunix 文件).

在系统处于前述的这种安全状态下时,任何用户(包括系统管理员)对所有的系统资源的访问,又都受到系统中按照安全模型制定的安全策略的制约.按照对安全模型(如 BLP 模型)的证明,只要在安全策略控制之下,系统初态又是安全状态,那么系统将始终处于安全状态之中.当然,若安全模型本身的证明是忽视了现实系统中的某些特点(如 BLP 模型起初并未考虑隐通道)或者在模型到实际系统中的对应中,存在偏差或盲点,那么系统存在安全的漏洞是可能的.

对于第 2 点:系统要保证任一主体对客体的访问都必须经过 COSIX V2.0 的控制.

最容易产生疑虑的是:一个用户任务有没有可能通过直接使用微核心的调用来实现对一些客体(设备)的读取,从而绕过文件服务器和进程服务器的安全控制呢?

在COSIX V2.0目前的实现当中,这是不可能的.COSIX V2.0采用了面向对象的管理方法.每个任务(比如服务器)作为一个目标,都有相应的端口,另一个任务作为客户要访问服务器的服务,就必须对该端口具有发送权限,以请求服务.服务端口的权限可以通过消息发送传给客户,如图6所示.

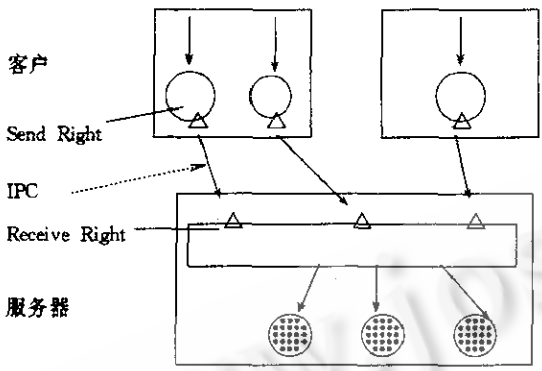

图6 端口和面向对象的管理方法

另外,COSIX V2.0服务器还实现了一个网络名字服务器(Netname Server),一个任务Ts可以把自己的服务端口注册到名字服务器中,另一个任务Tc要获取它的服务,可以从网络名字服务器中查询到Ts的端口权限,下一步就可以请求Ts的服务.

在FSS,POS等服务器初启时,将它们的服务器端口权限注册到了名字服务器中,当一个用户任务要请求FSS或POS的服务时,DML执行对网络名字服务器的查询工作,获取对FSS或POS的端口发送权限.这样,以后的对FSS或POS的请求就能够得到服务.

用户任务尽管可以直接调用微核心的一些功能,但是涉及到安全的核心功能函数(如device_open()),要想执行成功,也必须具有对相应端口的发送权.比如,要对设备进行操作,就必须先要有对master_device_port的发送权,才能执行device_open()得到具体设备的端口,进而再操纵设备.而master_device_port并未注册入网络名字服务器,只是将其发送权传送到FSS,从而用户任务不能通过netname_lookup()获取device_master_port,微核心等也不会向任一个用户任务传送对该port的发送权.因此,用户任务并不能成功地调用这些涉及安全的核心功能.

类似地,这也限制了用户另编一个FSS,通过该FSS实现文件操作,因为该FSS没有对设备端口的权限.

3 几个值得探索的问题

虽然,目前这样系统的实现可以满足高于B1级安全的需求,但应当看到,"计算机安全"不是一个绝对的概念,只要有足够的工具和足够的时间,一个被认为"安全"的系统仍然可以受到致命的攻击.这也是最有名的安全标准TCSEC被称作可信计算机系统评估准则,而不是安全计算机系统评估准则的原因,因为达到某一个安全级别的系统,只能说更安全了,变得可信了,而不意味着绝对安全性.不过,这也从另一个角度表明,计算机安全研究还有很长的路要走.对基于微核心的多服务器体系下的安全性设计,还存在如下的问题值得探索.

3.1 基于微核心的多服务器体系下的TCB组织和安排

本文在安全模型的基础上从总体上讨论了基于微核心的多服务器体系结构下操作系统的安全性设计.COSIX V2.0采用基于微核心的多服务器体系结构,总的工作可以分为微核

心设计、进程服务器(POS)的设计、文件系统服务器(FSS)等服务器的设计和系统安全性设计. 这种结构下的安全性设计的最大特点就是:由于系统中的安全客体,既有位于 FSS 内的文件型客体,如文件、目录、特别文件、管道等,也有位于 POS 内的诸如进程、IPC 等客体,安全性的设计是分别在基于微核心的各个服务器中实现安全管理,同时也实现用户管理服务器(UMS,用于注册检测)、审计服务器(Audit Server)等用于安全管理的服务器. 鉴于 FSS 与 POS 是 2 个独立的服务器,FSS 与 POS 内部各自存在着安全管理机制,可以说是很自然的一种系统 TCB 组织形式. 但是,这样的体系结构下 TCB 是否有更合理的组织形式,是否目前 COSIX V2.0 的 TCB 结构就是最优的?

3.2　微核心自身的安全性考虑

目前 COSIX V2.0 采用的基于微内核的多服务器体系下的安全管理仅仅局限于微核心之外,对于微核心本身的安全管理应有一个更完整的考虑. 实际上,象微核心中实现的诸如端口、存储对象,甚至包括任务,都应可以进一步施加安全机制,比如,可以考虑引入 BLP 模型到微核心中,对以上概念当作安全客体予以管理.

3.3　DML 对安全性的影响

COSIX V2.0 采用 DML 仿真库来实现与传统 UNIX 的二进制的兼容,但同时 DML 亦带来一些安全性上的问题. 由于 DML 是链接在每个用户任务各自的地址空间内,用户可以修改 DML 代码使得 DML 按其自己的意愿工作. 又由于 DML 的具体实现是被多个用户任务所共享的,下一个用户就可能使用被修改过的 DML,这点极可能导致安全漏洞. 目前 COSIX V2.0 是基于 80386 体系结构来实现的,通过将 DML 放在系统态来禁止用户任务在用户态修改 DML 代码. 但应当看到,很多机器只有核心态和用户态,如何解决 DML 所带来的安全问题,仍然需要作出有效的解答.

3.4　施加安全性对系统的不良影响

安全设计往往会导致系统性能降低. 一个好的安全性设计,应当在考虑安全目标的同时,亦注重不要使系统性能降低得过多.

另一个后果是,安全设计可能带来不良的用户界面,甚至在某些时候,一个通常被认为安全的操作,系统却当作不安全的操作拒绝执行.

3.5　其　它

①网络文件系统和分布式环境下,系统安全性的设计. 比如网络安全研究中流行的网络防火墙(Firewall)技术;②隐通道的防范;③UNIX 病毒防御;④密码管理.

参考文献

1　Rein Turn. Advances in computer system security, Volume II. Artech House, 1984.

2　Rein Turn. Advances in computer system security, Volume III. Artech House, 1988.

3　Stanley R Ames, Jr Morrie Gasser, Roger R Schell. Security kernel design and implementation: an introduction. Computer, July 1983, **16**(7):14～22.

4　Landwehr Carl E. Formal models for computer security. ACM Computing Surveys, Sept. 1981, **13**(3):247～278.

5　Richard Rashid *et al*. Mach: a system software kernel. Carnegie Mellon University.

6　David Golub, Randall Dean, Alessandro Forin *et al*. Unix as an application program. Carnegie Mellon University, Proceedings of the USENIX Summer Conference, June 1990.

7 Keith Loepere. Mach 3 kernel principles. Open Software Foundation and Carnegie Mellon University, NORMA-MK12; July 15, 1992.

8 Paul J Roy. Unix file access and caching in a multicomputer environment. Open Software Foundation, Proceedings of the USENIX Mach III Symposium, USENIX Association, April 1993.

9 Simon Patience. Redirecting system calls in mach 3.0, an alternative to the emulator. Open Software Foundation, Proceedings of the USENIX Mach III Symposium, USENIX Association, April 1993.

SECURITY RESEARCH FOR MICRO-KERNEL BASED NATIONALIZED OPERATING SYSTEM

LI Jun SUN Yufang

(*Institute of Software The Chinese Academy of Sciences Beijing* 100080)

Abstract This paper discusses the security design for nationalized COSIX V2.0 operating system. Through the discussion of micro-kernel based multi-server architecture of COSIX V2.0, the security design scheme of COSIX is formed. Finally, some issues are proposed with regard to security design in such an architecture.

Key words Micro kernel, multiserver, B1 security, discretionary access control, mandatory access control.