文章编号:1001 - 9081(2005)06 - 1286 - 04

一个分层隔离的操作系统内核

谢 钧[1,2],张 韬[1],张士庚[1],黄 皓[1]
(1. 南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093;
2. 解放军理工大学 指挥自动化学院,江苏 南京 210007)
(yulu_mail@263.net)

摘 要:传统单块结构操作系统的所有内核代码在一个公共的、共享的地址空间运行,因此内核中任何一个漏洞或在内核中加载任何不可靠模块都会威胁到整个系统的安全。研究并实现了一个分层隔离的操作系统安全内核,将内核特权分割隔离,阻止内核安全漏洞的扩散,防止恶意内核模块代码对内核代码数据的随意篡改。原型操作系统完全自主开发,支持 i386 体系结构。

关键词:操作系统安全;内核结构;隔离保护机制;计算机安全;
中图分类号: TP393.08 **文献标识码**:A

Layered and separated operating system kernel

XIE Jun[1,2], ZHANG Tao[1], ZHANG Shi-geng[1], HUANG Hao[1]
(1. *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing Jiangsu* 210093, *China*;
2. *Institute of Command Automation, PLA University of Science and Technology, Nanjing Jiangsu* 210007, *China*)

Abstract: In traditional monolithic kernel operating systems, all kernel codes run within a common and shared address space, and any vulnerabilities in kernel or any untrusted modules loaded in kernel would compromise the whole system security. The development of a layered and separated secure kernel was described in this paper. Since the powers of kernel are partitioned, the vulnerabilities of kernel are confined, and arbitrarily tampering of kernel by malice codes was prevented. The prototype system is entirely developed from beginning for the i386 architecture.

Key words: operating system security; kernel structure; separation mechanism; computer security;

0 引言

信息安全现在已被越来越多的人们所重视,各种安全产品应运而生,如防火墙、入侵检测系统、安全服务器等等。但是许多攻击者利用操作系统某些个别模块(网络协议、内存管理、文件管理等)的漏洞来获取高级权限,从而绕过应用安全软件(如网络防火墙)的安全策略和访问控制机制。

Unix、Linux 等单块结构操作系统的内核都是一个庞大、复杂的程序,虽然在设计上这些内核都是由几个逻辑上不同的部分组成,但这些逻辑部件都共享同一个内核地址空间并具有同样的特权。这样的系统一旦内核中某个模块出现漏洞,则可能被攻击者利用进而破坏系统数据,甚至劫持内核,并以此进行其他非法活动。

一个利用内核攻击的典型例子是在内核中插入木马并将其隐蔽起来。攻击过程首先获得主机的管理员权限,然后设法在内核中插入木马程序。为了隐藏插入的木马程序,攻击者修改系统调用表,将系统调用的请求定向到木马程序。当进程调用系统调用时,木马截获系统调用并在完成其动作之后激活真正的系统调用,以完成发出系统调用的进程所需要的功能。由于内核木马程序可以篡改系统调用,因此它的一切活动都可以不被记录下来,从而具有更强的隐蔽性。

木马程序之所以能截获所有系统调用并顺利完成木马的动作而不被发现,关键是内核中的木马程序可以访问所有内核空间甚至篡改系统调用。如果插入的木马程序只有有限的权限,可能就无法完成预定的动作,例如不能修改系统调用表从而无法不被发现地截获所有的系统调用而。

另一方面,广泛应用于各种安全操作系统的 BLP[1]、DTE[2] 等访问控制安全模型将系统抽象为若干主体、客体和一个访问监控器以及一套访问控制规则,但不讨论访问监控器内部的组成和结构以及如何保证访问监控器的正确执行。因此现有的各种安全操作系统都不能很好地解决操作系统内核的自身安全。本文提出一种新的操作系统内核结构来增强内核的自身安全,从而为各种安全应用提供一个坚实安全的系统平台。

1 操作系统的分层隔离结构

操作系统的功能实际上是由很多逻辑模块来共同完成的,一个简单的典型划分包括:进程管理、内存管理、文件系统、设备管理、网络系统等。这些逻辑模块虽然相互关系紧密但功能相对独立,它们有自己的数据结构且相互间有较明确的接口界面,但在传统单块内核系统中,这些逻辑模块间可以直接互相访问,一个模块的错误和漏洞会直接影响其他模块

收稿日期:2004 - 12 - 20;**修订日期**:2005 - 03 - 15
基金项目:国家自然科学基金资助项目(60473093);江苏省自然科学基金资助项目(BK2002073)
作者简介:谢钧(1973 -),男,四川成都人,博士研究生,主要研究方向:计算机安全; 张韬(1982 -),男,江苏苏州人,硕士研究生,主要研究方向:计算机安全; 张士庚(1981 -),男,山东滕州人,硕士研究生,主要研究方向:计算机安全; 黄皓(1957 -),男,江苏海门人,教授,博士生导师,主要研究方向:计算机网络和信息安全.

甚至整个系统的安全。

操作系统各功能模块间有着服务与被服务的层次关系,从安全角度看各层都有各自的安全条件(通过底层的安全服务和自己的实现来保证),又为其他上层提供安全服务,以保证上层安全功能的实现。强制的安全层次结构为操作系统建立了有纵深的防御体系,在不同强度的攻击下能最大限度地保证系统的各种安全特性不被破坏。为此设计了如图 1 所示的操作系统内核安全结构。

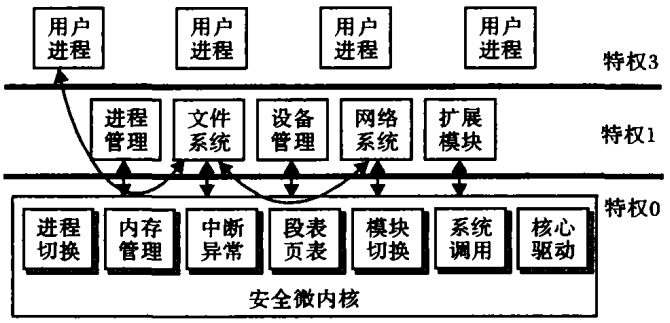

图 1　分层隔离结构

安全微内核负责最核心的特权功能和最关键的访问控制功能,实现安全区域的隔离、服务模块切换、任务切换、安全消息机制、中断处理、段表页表等安全关键数据的保护等功能。安全微内核管理所有区域隔离、访问控制所需要的信息,其他功能模块不能直接访问。用户进程对各功能服务模块的访问以及各功能服务模块之间的访问必须通过安全微内核的系统调用和内核模块切换机制来实现。一些性能敏感设备驱动也放入安全微内核中,如网卡驱动和磁盘驱动等。

服务功能层由进程管理、文件系统、网络系统、设备管理等模块组成,并且这些模块被安全微内核提供的内核隔离机制完全隔离开,它们之间的访问必须通过安全微内核的访问控制机制。同时用户进程对各服务功能模块的访问也必须通过安全微内核。因此单个模块的漏洞和错误不会直接影响其他模块和安全微内核,从而增强了系统的安全性。

将段表页表保存在安全微内核中,其他操作系统服务模块不能通过篡改段表页表来破坏安全微内核的内存空间隔离功能。由于低层安全微内核保证了安全区域隔离服务不被破坏,文件系统则不能篡改进程管理的代码和数据,如进程标识符、进程的消息队列等。同样进程管理模块也不能通过破坏文件系统来访问各种操作系统安全敏感文件,从而形成了有纵深的层间分级、层内隔离的保护体系,不会因为个别漏洞导致整个安全机制被破坏。

Linux 为提高系统的可扩展性提供了可装载内核模块功能,模块可以被单独编译然后在需要时装载到内核中,成为内核代码的一部分。该方法即保证了内核的紧凑性又提高了系统的可扩展性。但由于这些模块代码与其他内核代码一样具有特权,因此为黑客提供了一个攻击内核的有效途径:通过装载恶意模块到内核中可以获得所有要的权力而不被用户发现。在本文的安全结构中将各种可装载内核模块(扩展模块)放入一个彼此隔离的保护域中,它们不能随意访问其他内核代码和数据,只能直接访问安全策略所允许的内存空间和 I/O 空间,从而有效的限制了内核模块攻击的能力。

2　原型系统的设计与实现

现有的各种流行操作系统如:Windows NT、Unix 以及 Linux 等提供了丰富的功能并有大量的应用,但它们都较复杂、庞大而难以验证。我们的目的是建立一个实现以上分层结构但小到便于进行形式化验证的系统,同时该系统面向安全专用系统如网络防火墙等而不是通用操作系统,因此我们的原型系统内核完全自主重新开发。目前系统只支持 i386 体系结构。

目前原型系统已实现:安全微内核(主要部分)、内存管理、进程管理、文件系统、简单设备管理、内核模块隔离机制、网卡驱动等基本功能,并作为应用实例在此结构上实现了基于服务模块的单向网关模块以及一些简单的应用程序(如 shell 和文本编辑器)。单向网关模块实现内外网络间的应用层(目前只支持 FTP 协议)数据传输的单向性。原型系统总体结构如图 2 所示。

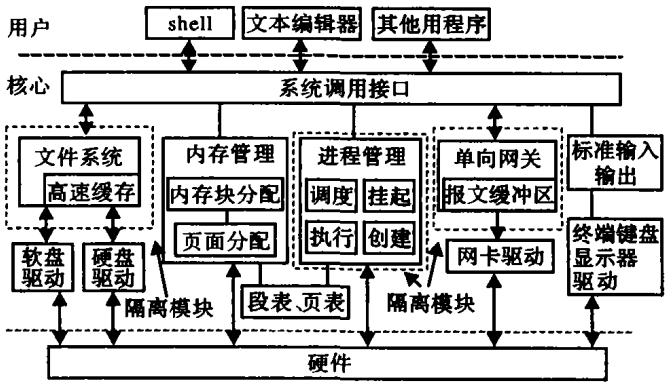

图 2　原型系统总体结构

为了使系统达到小而高效的目的,简化了一般通用操作系统的一些复杂功能,如:内存交换、虚拟文件系统等。

3　基于分段保护的内核模块隔离机制

内核模块隔离机制是实现以上分层隔离结构的关键技术之一。内核模块隔离机制通过 i386 提供的硬件保护机制保证安全微内核的不被篡改性,以及操作系统各服务功能模块之间存储空间和 I/O 空间的隔离。

3.1　利用分段机制隔离地址空间

按照内核模块隔离模型的安全要求,用户空间的代码不能访问安全微内核和服务功能模块的地址空间,各服务功能模块的代码不能访问安全微内核和用户进程以及其他保护模块的地址空间。为实现以上地址空间隔离要求,利用 i386 体系结构中的分段保护机制将线性地址空间划分为多个不同属性的空间,如图 3 所示。

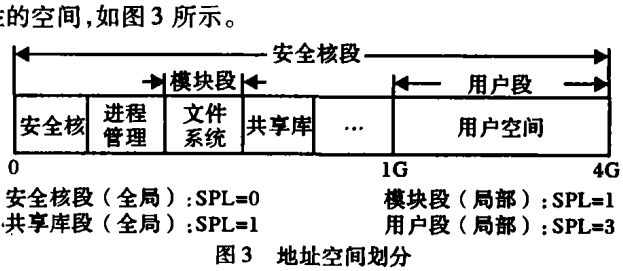

图 3　地址空间划分

虽然低特权级代码不能访问高特权级段,但还不能实现服务功能模块对用户进程以及其他服务功能模块的地址空间的访问隔离要求。为此系统中每个模块有一个局部描述符表(LDT),所有用户进程共用一个 LDT。只有安全微内核段和共享函数库段的段描述符在全局描述符表(GDT)中,而各模块和用户进程的代码段以及数据段的段描述符都在各自的 LDT 中。在 i386 中通过装载 LDT 寄存器来指定当前 LDT,当

执行代码从安全微内核切换到各服务功能模块或用户进程时,安全微内核先装载指定 LDT 然后再进行切换。由于各模块和用户进程的段不在同一个 LDT 中,因此各模块代码不可能访问其他模块段和用户段。

3.2 I/O 空间的隔离

一些被隔离的服务功能模块如:网络系统、设备驱动等需要访问 I/O 端口。如果允许其访问所有 I/O 端口则不能实现隔离的安全目标。例如,它们可以绕过文件系统的访问控制而直接通过 I/O 来访问磁盘数据。在 i386 中可以通过设置任务状态段 TSS 的 I/O 允许位图来允许低于 I/O 特权级的代码访问指定 I/O 端口。因此原型系统为各服务功能模块设置独立的 TSS 数据结构,在切换服务功能模块时切换当前 TSS,从而实现 I/O 空间的隔离。

3.3 调用关系与特权切换

用户进程通过系统调用访问安全微内核来获得系统服务。安全微内核服务程序根据需要可通过"模块调用"访问服务功能模块提供的服务,而服务功能模块也可以通过"内核调用"来访问安全微内核提供给服务功能模块的公共服务。

系统调用、模块调用及内核调用都需要跨越特权边界。在 i386 体系结构中通过中断门可以实现特权级切换。原型系统利用软件中断 INT80 实现系统调用,而通过 INT81 和 INT82 分别实现模块调用和内核调用,如图 4 所示。

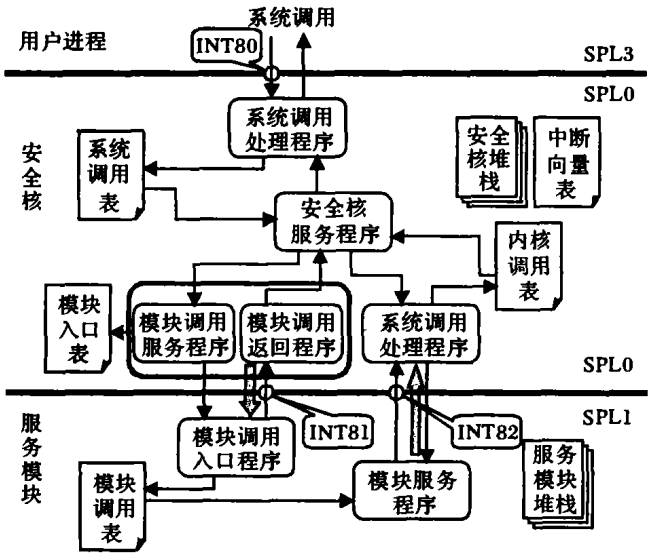

图 4 调用关系与特权切换

由于中断门只允许低特权级代码调用高特权级代码,而模块调用正好相反,是高特权级代码调用低特权级代码,因此必须通过中断返回指令 iret 来调用模块调用入口程序,而用 int 指令来返回安全微内核。为实现这种非常规调用,模块调用服务程序需要在内核堆栈中为模块调用构造中断返回上下文,使得执行 iret 指令后通过恢复硬件上下文的方式进入指定的服务功能模块,并从模块调用入口程序开始执行同时切换相应的特权级。

3.4 模块调用的过程与算法

当安全核调用服务模块代码时,先调用模块调用服务程序 module_call_inv(moduleId, arg1, …),该程序执行:

1) 在内核堆栈保存上下文环境。根据 moduleId 获得目的模块的 LDT 描述符,在安全核堆栈中保存当前 LDTR,并装载新 LDT 描述符。

2) 在堆栈中保存当前 TSS 的 esp0,并将当前 esp 保存在 TSS 的 esp0 中。

3) 在 eax(模块调用号), ebx, ecx, edx, esi, edi 及 ebp 中保存模块调用参数。

4) 在安全核堆栈中构建服务模块硬件环境,包括服务模块的堆栈段选择符和堆栈指针,标志寄存器 EFLAGS 以及服务模块代码的段选择符和入口地址。

5) 最后执行 IRET 从安全核堆栈中恢复构建的服务模块硬件环境并切换到服务模块段执行模块调用入口程序。

模块调用入口程序调用模块服务程序完成模块服务后执行 INT 81 返回安全核并执行模块调用返回处理程序 pmodule_call_ret 的地址,该程序负责恢复安全核的执行:

1) 将 esp 修改为 TSS 中 esp0 来忽略由硬件自动保存的服务模块执行环境。

2) 恢复堆栈中保存的 TSS 中 esp0 原值,恢复 LDTR,恢复各段寄存器和 EFLAGS。

3) 最后执行 RET 指令恢复因调用 pmodule_call_inv 保存在安全核堆栈中的执行环境,恢复安全核的执行。

4 共享内存空间与内核共享函数库

在安全微内核与服务功能模块之间,以及服务功能模块与服务功能模块之间有时需要传递大块的数据,如网络数据包等。在隔离保护空间之间一般是利用 CPU 的通用寄存器来传递参数,但要传递大块数据,在隔离保护域之间必须有可共享的地址空间。

由于安全微内核段覆盖进程的所有地址空间,并且拥有最高特权级,可以访问所有服务功能模块的地址空间。系统在服务功能模块段中分配了一个数据区专门用于安全微内核传递数据给服务功能模块,如图 5(a)所示。由于所有的服务功能模块间的调用必须通过安全微内核,因此各服务功能模块间的大块数据的传递可以通过安全微内核从一个服务功能模块地址空间复制到另一服务功能模块空间,但大量数据的复制势必会影响系统的性能。在我们的原型系统中利用共享数据段来实现服务功能模块间大块数据的零拷贝。如图 5(b)所示,每个服务功能模块可以拥有多个数据段,这些段的描述符都在各服务功能模块的局部段表中,并可以通过安全微内核提供的内核调用服务将自己的数据段的访问权赋予其他服务功能模块,安全微内核在各服务功能模块局部段表中为各共享数据段设置不同访问属性的描述符,以实现在各服务功能模块间地址空间的共享及访问控制。

在单向网关模块中为了避免网络报文缓冲区的复制,为单向网关模块分配一个 DMA 内存段,该段与安全核共享,作为网络报文缓冲区段(包括分配池和报文队列)。当网卡硬件收到网络报文后自动将数据复制到某报文缓冲区(DMA 内存)中,驱动程序将其链接到接收报文队列中,由于共享报文缓冲区段,单向网关模块处理完报文后将其链接到另一网卡的发送队列中,启动网卡驱动程序发送该报文。在整个处理过程中无需将报文数据复制多次。

在内核中有一些库函数是各服务功能模块经常使用的,但由于各服务功能模块在地址空间上被隔离,不能直接调用。这些库函数并不涉及操作系统的服务功能,仅仅是完成一些

常用操作,所以无需私有数据,如 memcopy、strcmp 等。原型系统利用共享代码段实现了内核模块共享函数库。共享函数库段的是一个代码段,且段特权级为 1,其段描述符位于 GDT 中,因此所有服务功能模块都可以直接调用该段的函数而不需要切换特权级。

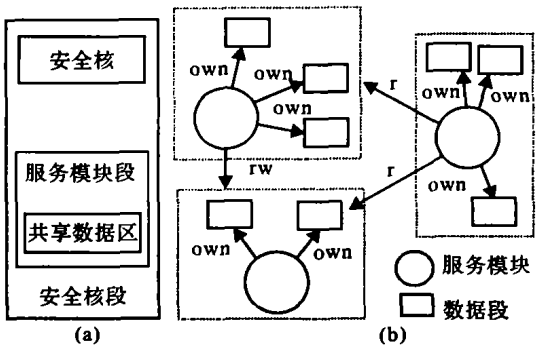

图 5　共享内存空间

5　文件系统透明操作磁盘数据块

文件系统负责文件超级块、索引节点、目录的管理与操作,但不能直接访问磁盘数据。系统将 I/O 特权级设置为 0,因此文件系统不具有访问 I/O 的权力,不能直接操作磁盘,必须通过内核调用机制调用的磁盘驱动程序来完成对磁盘数据块的读写操作。

根据系统安全要求文件系统不能随意篡改和读取用户程序从磁盘中读取的文件内容,在其完成正常的文件操作的功能时,即在用户程序通过文件系统从磁盘中读取数据块并复制到用户空间的过程中,文件系统不能直接访问从磁盘上读取的数据块。即文件系统要透明地操作磁盘数据块。为此,系统在安全核中为用户空间存放文件内容的数据缓冲区建立用户缓冲区描述符,该结构中记录用户缓冲区的地址、大小以及进程 ID 等。磁盘数据缓冲区在安全核中分配。当文件系统调用磁盘驱动程序读取磁盘数据块时,磁盘驱动将数据放入磁盘数据缓冲区中并不复制到文件系统中,而文件系统只获得一个缓冲区标号,当文件系统要将用户需要的文件数据复制到用户空间时,将磁盘缓冲区标号和用户缓冲区标号以及复制偏移和大小传递给安全核的缓冲区复制服务,由安全核来完成复制工作,并检查相应的安全属性。

6　内核隔离机制的性能测试与分析

由于分层隔离的内核结构使得各模块间的访问要进行特权级的切换,这是影响系统性能的主要因素。为了说明引入内核模块隔离机制后对系统性能的影响,我们对模块调用、内核调用的性能开销进行了测试,同时与函数调用、系统调用以及进程间通信(一般微内核系统多采用进程间通信机制)进行了比较。表 1 为我们在 PC 机上(英特尔奔腾 IV1GHz 处理器,256M 内存)对几种调用方式的空调用进行测试的结果。

表 1　几种空调用的性能比较

调用方式	模块调用	内核调用	函数调用	系统调用	进程间通信
时间	1.03μs	0.91μs	0.015μs	0.92μs	4.3μs
特权切换次数	2	2	0	2	2

由表 1 可以看出模块调用与内核调用都大体与系统调用相当,但由于模块调用需要人为构建硬件上下文因此性能稍差一些。由于服务功能模块对服务功能模块的调用需要一次模块调用和一次内核调用,因此大约是系统调用的 2 倍。但因为无须进程调用和进程间切换,不切换页全局目录,不刷新转换后援缓冲器 TLB,是一种轻量级的上下文切换,与进程间通信相比有一定的性能优势。

7　结语

安全操作系统是各种安全产品的基石,但目前的各种安全操作系统考虑的主要内容是如何通过访问控制机制来约束用户进程以及如何监视用户进程的异常操作,而忽略了对自身安全的保障。Unix、Linux 等单块结构的内核中一旦某模块存在漏洞(如缓冲区溢出)或被装载恶意内核代码,则攻击者可能利用其破坏系统数据甚至劫持内核。在 MACH[3]、QNX[4] 等微内核结构的系统中,操作系统内核仅提供很少的服务,而其他操作系统服务模块同用户进程一样在自己独立的地址空间中运行,虽然可以增加系统的可靠性,但其结构却并不是为安全目的而设计的。本文的操作系统分层隔离安全结构,将内核划分为安全微内核层和服务功能模块层,由于 i386 的硬件保护机制保证了安全微内核的不被篡改性,有效地隔离了各种内核安全威胁,抵御内核模块攻击等针对系统内核的攻击,保证了系统的整体安全性,从而为各种安全软件提供一个更加坚实安全的系统平台。

下一步的研究工作包括对设计的分层安全内核的安全相关操作和安全状态进行形式化建模,定义安全微内核和各服务模块的安全条件,证明设计的内核分层结构的合理性和正确性。研究各服务模块在安全微内核保护的安全相关的信息确定的情况下的最大权能,确定这些最大权能不影响系统安全的条件,并对这些权能和条件进行形式化的描述。通过形式化方法证明各服务模块不能破坏安全内核的安全性,也不能破坏其他模块的安全性。进一步完善原型系统并应用到实际的防火墙等安全产品上。

参考文献:

[1] BELL DE, LAPADULA LJ. Secure computer systems: unified exposition and multics interpretation[R]. MTR-2997, MITRE Corp, 1976.

[2] WALKER KM, STEME DF, BADGER ML, *et al*. Confining root programs with domain and type enforcement(DTE)[A]. Proceedings of the 6th USENIX Security Symposium[C]. Usenix Association, 1996. 21-36.

[3] BLACK DL, GOLUB DB, JULIN DP, *et al*. Microkernel Operating System Architecture and Mach[A]. Proceedings of the USENIX Workshop on Micro-Kernels and Other Kernel Architectures[C]. Usenix Association, 1992. 11-30.

[4] HILDEBRAND D. An architectural overview of QNX[A]. Proceedings of the USENIX Workshop on Micro-Kernels and Other Kernel Architectures[C]. Usenix Association, 1992. 113-126.