基于RBAC的信息终端内核模型

李洪心 关可卿

(东北财经大学管理科学与工程学院 大连 116025)

摘 要 随着3G技术、无线网络等新兴技术的发展,桌面系统、嵌入式系统用作个人或企业用户的信息终端,数据的保密性将面临更严峻的安全威胁。为提高信息终端的安全性,分析了操作系统中进程的非法操作对系统信息安全产生的威胁,将RBAC模型应用于信息终端中操作系统内核的安全管理机制中,构建了基于RBAC的系统进程访问控制模型,给出了具体的实现框架,并针对开源操作系统与非开源操作系统,分别给出了实施方法。

关键词 RBAC,访问控制,操作系统内核,数据安全

Model of Information Terminal's Kernel Based on RBAC

LI Hong-xin GUAN Ke-qing

(School of Management Science and Engineering, Dongbei University of Financial and Economics, Dalian 116025, China)

Abstract With the development of such technologies as 3G and wireless network, desktop systems and embedded systems are used as individual's or enterprise user's information terminals, and confidentiality of user's data will face more serious security threats than before. To improve the security of information terminals, this paper analyzed the threats to system's information security which comes from the abnormal operation by processes in operation systems, and applied RBAC model into the safety management mechanism of operation system's kernel. The access control model of system's process based on RBAC was built, and an implement framework was proposed. Finally, this paper illustrated the methods of implementation of the model for open source systems and non-open source systems.

Keywords RBAC, Access control, Operation system kernel, Data security

1 引言

在信息安全的诸多研究领域中,信息终端操作系统的安全性受到广泛关注。特别值得注意的是,随着普适计算相关技术的发展,各种小型、嵌入式智能设备通过无线网络为个人用户提供了"无时无处不在"的服务终端[1]。诸如Windows、Linux等主流操作系统的应用已经从个人电脑逐渐扩展到掌上电脑、PDA、智能手机等移动终端设备。与此同时,随着移动电子支付、在线支付等技术的发展,由于受到经济利益的驱使,个人电脑、智能手机等各种信息终端的操作系统将会成为木马程序、病毒程序的重点攻击目标。在信息终端的操作系统层面对访问控制的保护措施一旦被攻破,木马程序、病毒程序就能给个人用户或企业造成直接的经济损失,并且会对信息的保密性产生重大威胁。为从信息终端的操作系统底层提高其安全性,本文在研究操作系统实现机制以及访问控制理论的基础上,按照基于角色的访问控制策略(Role Based Access Control,RBAC)构建了操作系统内核安全框架模型,通过该模型提高信息终端系统的安全性。

基于角色的访问控制的思想最初于20世纪90年代早期由美国国家标准和技术委员会(NIST)的Ferraiolo等人提出,此后NIST专门成立了RBAC研究机构,对基于角色的访问控制进行系统的研究和应用[2]。我国众多学者也在理论及应用层面对基于角色的访问控制进行了多个角度的研究。在电子商务领域,马勇、卿斯汉(2007)等人提出了一种基于RBAC的电子商务匿名性与可追究性的实现方案,将可信第三方的权力进行合理划分从而达到相互牵制,并且能够相互协调完成匿名性与可追究性的实现[3]。李洪心等(2009)提出用RBAC来改进SET支付系统,通过引入基于RBAC的公共支付平台来提高SET支付系统的灵活性[4]。在信息安全、情报学等领域,单智勇、孙玉芳(2004)将RBAC模型进行面向操作系统的扩展,并结合Capability等机制,将其实现在红旗安全操作系统中[5]。王亚民(2005)基于RBAC思想构建了信息系统的访问控制模型,使组织中的用户可以通过角色间接地访问系统资源[6]。曾忠平等(2007)将RBAC应用于电子政务信息资源访问控制,保障了电子政务信息的安全合理利用[7]。RBAC模型与传统的访问控制策略相比,前者能够兼顾权限配置的灵活性与安全性,因此它在理论和应用层面都受到了广泛关注。

本文在系统内核层面对威胁信息安全的关键因素进行分析,构建基于系统内核的RBAC模型,然后阐述模型的实现方法,最后讨论如何将模型实施在不同类型的信息终端的系统中。

到稿日期:2010-12-19 返修日期:2011-02-26 本文受国家自然科学基金项目(71002094)资助。

李洪心(1956-),女,博士,教授,博士生导师,主要研究方向为电子商务、供应链建模;关可卿(1979-),男,博士生,主要研究方向为电子商务、信息安全,E-mail:keqinguan@163.com(通信作者)。

2 基于 RBAC 的系统内核安全强化模型

2.1 病毒与木马程序对系统信息安全造成的威胁

计算机病毒和木马程序是黑客窃取他人数据的重要手段。黑客常常利用这些技术，通过一些隐蔽的操作来实现对他人关键数据的窃取。为分析这种实现机制，定义如下概念。

定义 1(进程主体)　在系统内核中创建的进程，通过访问特定资源来实现预期的功能。

定义 2(可信客体)　进程主体具有访问权限、可以正常访问或创建的系统资源的集合。

定义 3(可疑客体)　对于进程主体而言，除可信客体之外的所有系统资源。

基于上述定义，可以把计算机病毒和木马程序在操作系统中的行为描述为如下特点：

(1)在可视化操作空间，进程主体对可信客体进行正常操作，产生正常的可视化操作结果。例如，用户在访问互联网时，网页的页面在浏览器中正常地展现出来。

(2)在非可视化操作空间，进程主体对可疑客体进行非正常操作，在这个过程中，不产生任何可视化的操作结果，因此，用户或系统管理员无法觉察。例如，用户在浏览网页时，关于系统启动的自动加载选项被修改或木马程序在系统后台读取用户的数据文件，并把读到的数据上传至网络。

以上特点具体如图 1 所示。

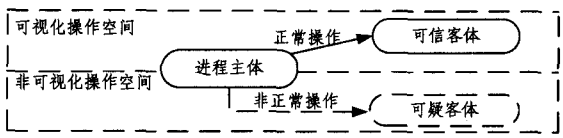

图 1　进程主体可能产生的非正常操作

因此，为防范计算机病毒、木马程序对计算机系统安全的威胁，可以通过限制进程主体对可疑客体的非正常操作来实现，即对进程主体进行有效的访问控制，阻止进程访问可疑客体。这样，即使计算机病毒、木马程序入侵到信息终端的系统中，也无法对数据进行窃取。

2.2 模型的构建

由于病毒、木马程序有可能绕过系统上层的保护机非法读取用户的关键数据信息，因此我们从信息终端的系统内核层面对系统安全进行强化，以有效地提高系统的安全性。按照 RBAC 思想构建了对系统进程的行为进行访问控制的安全强化模型，即面向系统内核的 RBAC 访问控制模型，在本文中称其为 Kernel-RBAC 模型，如图 2 所示。

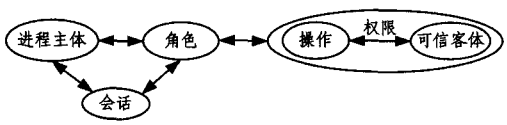

图 2　Kernel-RBAC 模型

2.2.1 模型涉及的元素

进程主体 S(Subject)：在系统内核中管理的进程；其中包括系统自发运行的进程，以及用户运行应用程序而启动的进程。

角色 R(role)：进程主体在系统中为完成一项工作而被赋予某些权限的概念化抽象，既包括进程在系统中的角色，如：系统进程、应用程序等；又包括操作该进程的用户当前的角色，如系统管理员、普通用户等。通过角色继承机制，子角色可以继承父角色的权限。

权限 P(permission)：某角色的进程主体对特定资源进行特定操作的许可。权限被授予角色，而不是直接授予进程主体。进程主体通过角色获得权限。

操作 Op(Operation)：对数据信息进行读取、写入、创建、删除等操作。

可信客体 Ob(Object)：某角色的进程具有正常访问权限的系统资源，可以包括系统关键资源、普通用户资源；具体而言，可以包括特定路径下的数据文件、特定区域的内存中的数据、特定设备或特定端口等。

会话 SE(Session)：为使用户与系统之间能够正常交互而建立的连接，对于当前活动的进程主体进行动态管理，在进程创建时，系统自动建立会话，并且在会话中激活相应的角色。

2.2.2 元素之间的映射关系

(1)进程主体 S 与角色 R 之间的映射

进程主体 S 与角色 R 是多对多的映射关系，即一个进程主体可以被赋予多个角色，同一个角色可以被赋予多个进程主体。

(2)角色 R 与权限 P 之间的映射

角色 R 与权限 P 之间是多对多的映射关系，即一个角色可以拥有多个权限，同一个权限可以被赋予多个角色。

(3)进程主体 S 与会话 SE 之间的映射

进程主体 S 与会话 SE 之间是一对多的映射，一个进程主体可以创建多个会话，一个会话只能属于一个进程主体。

(4)角色 R 与会话 SE 之间的映射

角色 R 与会话 SE 之间是多对多的映射关系，在一个会话中可以激活多个角色，同一个角色可以在多个会话中被激活。

2.2.3 访问控制规则

访问控制规则的制定是确保数据安全的关键，系统通过特定的访问控制规则来判断进程对资源的访问操作是否合法，若合法，则允许访问，否则拒绝访问。设进程主体 S 的可信操作集合为：

S_{op} ={该进程在当前会话中激活的角色所具有的权限的集合}。

设 S 的当前操作为 $op_{current}$，若 $op_{current} \in S_{op}$，则视为正常访问，否则视为非正常访问，并进行阻止。以数据文件为例进行说明，设 DIR_{admin} 为系统管理员所保存的数据文件的路径集合，DIR_{user} 为某普通用户所保存的数据文件的路径集合。当该用户启动的进程试图访问 DIR_{admin} 路径下的文件时，则视为非法访问；当该进程试图访问 DIR_{user} 路径下的文件时，则视为合法访问。

3 Kernel-RBAC 模型的实现

3.1 Kernel-RBAC 模型在操作系统内核的实现框架

在操作系统内核实现 Kernel-RBAC 模型，即在原有的操作系统内核中构建 Kernel-RBAC 模块，通过 Kernel-RBAC 模块把进程的操作请求，传递给系统内核应用程序接口(即系统内核 API)，在传递的过程中，进行相应的访问控制，如图 3 所示，具体说明如下：

(1)把进程和线程统一抽象为进程来进行管理,不考虑进程间消息的传递。

(2)系统内核在创建进程时就会把进程名、进程 ID 等关键信息登录到进程管理模块。

(3)在进程需要访问系统资源(数据信息)时,向系统内核发起系统调用的操作请求,RBAC 模块内部的访问控制模块会截取进程的操作请求,并由 AEF(访问控制执行功能)模块向 ADF(访问控制判决功能)模块发起判决请求,ADF 模块从进程管理模块、角色管理模块、资源管理模块获取相关信息,即前文中提到的进程主体、角色、操作、可信客体等信息。ADF 模块获取这些信息后,进行访问控制的判定,并将判定结果反馈给 AEF 模块,在必要的情况下更新访问控制缓存以提高处理效率。

(4)AEF 模块根据 ADF 模块反馈的访问控制判定结果来决定操作请求是否被允许。如果允许,就会调用系统内核 API 进行实际的操作,否则直接返回出错码。

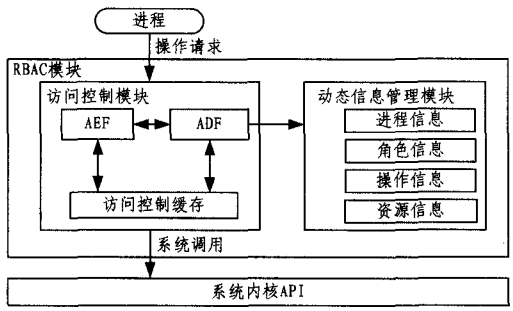

图 3 Kernel-RBAC 模型的实现框架

3.2 关键数据结构的创建

在图 3 中提到的动态信息管理模块主要涉及到对进程信息、角色信息、操作信息、资源信息的管理,管理这 4 类信息的数据结构及其相互关系如图 4 所示,下面分别加以介绍。

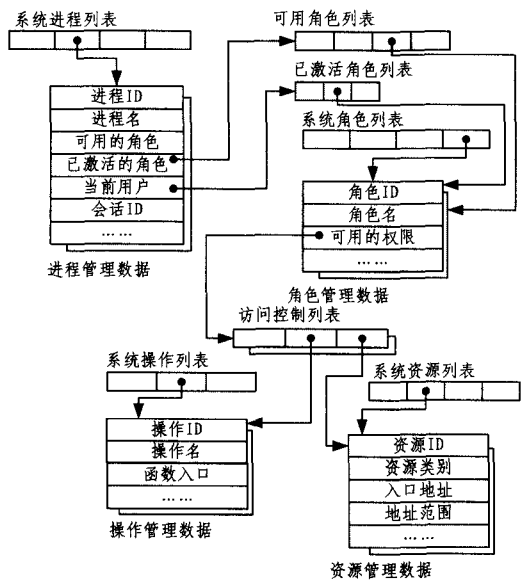

图 4 实现 Kernel-RBAC 的数据结构

(1)进程信息:在系统初始化时构建系统进程列表,用于管理进程信息,以支持 Kernel-RBAC 模型。列表中的每一个数据项里面有一个指针,指向用于存储进程管理数据的结构体。该结构体中保存了进程 ID、进程名、可用的角色、已激活的角色、当前用户、会话 ID 等信息。其中可用的角色与已激活的角色分别指向可用角色列表与已激活角色列表。

(2)角色信息:在系统初始化时,建立系统角色列表,用于管理角色信息。角色列表中每个数据项的指针指向角色管理数据结构体,该结构体中存储了角色 ID、角色名、可用的权限等信息,在这里由访问控制列表管理可用的权限。

(3)操作信息:由系统操作列表的每个数据项中的指针指向操作管理数据结构体,该结构体中包括操作 ID、操作名、函数入口地址等信息。

(4)资源信息:由系统资源列表的每个数据项中的指针指向资源管理数据结构体,该结构体中包括资源 ID、资源类别、入口地址、地址范围等信息。在其中的资源类别中,可以对特定资源进行标识,以便于对系统关键资源进行保护。

3.3 访问控制功能在系统中的实现时序

在系统启动、对各个执行模块进行初始化时即加载 Kernel-RBAC 访问控制模块,并开始对进程行为进行监控,当进程试图超越自身权限访问时进行阻止。在系统运行时,访问控制模块在系统中的执行时序如图 5 所示,具体说明如下:

(1)库函数调用:用户进程在运行过程中通过标准库函数的调用来获取相应的系统资源,即数据信息。

(2)内核函数调用:标准库函数通过调用内核 API 函数来访问特定数据信息,如文件、内存等,此外,用户进程也有可能直接调用内核函数。

(3)访问控制判定:包含 Kernel-RBAC 的访问控制服务模块截获标准库函数对内核函数的调用操作,并根据当前进程的角色相关信息对该进程的当前函数调用的操作进行访问控制判定,判断是否允许当前访问。若允许,则执行第 4 步操作,否则,直接返回出错码。

(4)实现函数调用:经过前面的判断,认为当前函数调用为可信任的操作,则实现函数调用,允许用户进程对相关数据信息进行访问。

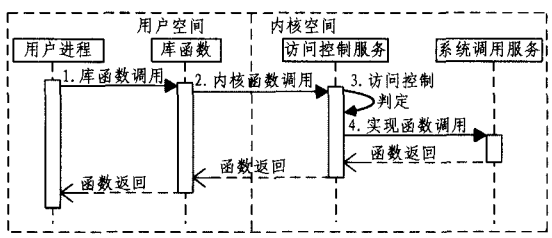

图 5 在系统内核中截取系统调用的实现时序

3.4 模型的性能测试

根据以上设计,以 Ubuntu 10.04 版操作系统作为实验平台,在 Linux 2.6.32 版本的内核上构建 Kernel-RBAC 模型的测试环境,在 CPU 主频 1.73G Hz,2G 内存的 PC 机上进行性能测试,针对 1k 字节的数据分 R0、R1、W0、W1 4 种情况重复 10 次操作,取平均时间进行对比,具体测试结果如表 1 所列(精确度为 μs)。其中 R0 代表关闭 Kernel-RBAC 的情况下通过系统内核直接读取数据并输出在终端所用的时间,R1 代表开启 Kernel-RBAC 的情况下,经过访问控制判定后通过系统内核读取数据并输出在终端所用的时间;W0、W1 分别代表关闭和开启 Kernel-RBAC 的情况下,通过系统内核写入数据所需要的时间。根据实验结果可知,该访问控制模型对系统性能的影响处于用户可接受的范围内。

表1 Kernel-RBAC模型对系统性能的影响测试

	R0	R1	W0	W1
1	136	136	50	50
2	135	140	49	49
3	141	136	50	50
4	136	139	50	51
5	134	137	49	50
6	142	140	50	50
7	136	135	50	50
8	140	141	49	51
9	135	137	50	50
10	137	136	50	50
平均	137.2	137.7	49.7	50.1

3.5 Kernel-RBAC模型在不同系统中的实施

不同类型的信息终端操作系统具有不同的具体实现机制，所以要把Kernel-RBAC模型加载到原有系统的内核中，就需要针对各种信息终端中原有操作系统自身的特点考虑具体的实施方法。目前无论是以桌面系统还是以嵌入式移动设备作为信息终端，使用的操作系统主要分为两大类，即以Linux为代表的开源操作系统和以Windows为代表的非开源操作系统，分别介绍在这两大类系统下的Kernel-RBAC模型的实施方式。

3.5.1 基于内核源代码的实施

针对以Linux为代表的开源操作系统，可以直接对内核源代码进行修改，增加相应的访问控制机制以实施Kernel-RBAC模型，具体实施方法如下：

(1)构建内核信息监视器(Kernel Information Monitor，KIM)，实现动态信息管理功能。保证KIM在系统启动时即被启动，通过KIM创建图4所示的数据结构，根据系统中原有的task_struct数据结构中的pid和uid获取进程信息和用户信息，为管理进程信息、角色信息、权限信息分别创建task_manage_struct、role_manage_struct、permission_control_struct等数据结构，建立进程与角色的关联，并实时监控系统进程信息、资源信息。

(2)构建系统调用监控器(System Call Controller，SCC)，实现访问控制的功能，其核心功能在于对系统调用的控制。由于系统内核通过特定的中断0x80来管理系统调用[8]，并通过system_call_table[]数组来索引具体的系统函数，因此，需要改变system_call_table[]数组所指向的入口位置，使其指向SCC中的系统调用管理函数，该函数首先保存系统调用的实际入口，而后根据系统调用号、进程号和调用参数以及KIM所提供的信息来进行访问控制判定，最终根据访问控制的判定结果来决定是否进行实际的函数调用。

3.5.2 基于系统服务挂载的实施

对于非开源操作系统，由于其内核源代码不对外公开，因此不能以修改内核源代码的方式将Kernel-RBAC模型实施在这一类系统内核上，以Microsoft Windows为例来说明具体的实施方法。

Windows操作系统通过动态链接库的形式导出应用程序调用接口(API)供外部应用程序调用。这些API可以分为两类，一类运行在用户态，另一类运行在内核态。运行在内核态的API被称为Native API，这一类API是我们所关注的。

由于不能对原有的系统内核进行修改，因此只能通过构建内核驱动的方式实现内核级别的访问控制。这里同样需要实现KIM与SCC两个模块。其中KIM的实现机制及数据结构与3.5.1节中提到的方法类似，这里不再赘述，重点讨论SCC的实现方法。

Windows下的系统调用由ntoskrnl.exe实现，通过ntdll.dll动态库进行调用。ntdll.dll作为系统调用的前端，使用INT 2EH中断使处理器由用户态进入内核态。该中断对应的处理函数按照Native API的ID值和调用参数地址根据系统服务分配表(System Service Dispatch Table，SSDT)进行系统调用[9]。

因此，在Windows内核驱动中构建SCC，对系统调用进行控制，就需要对SSDT中保存Native API的入口地址进行备份和更改，使其指向SCC中的系统调用管理函数system_call_controller()，由该函数结合KIM提供的信息进行访问控制判定。当外部程序调用Native API时，系统将首先执行system_call_controller()函数，该函数将决定是否进行实际的Native API调用，从而实现对外部调用的控制。此外，SCC还应建立对SSDT的保护机制，防止恶意程序再度对SSDT进行修改。

综上所述，无论在哪一类信息终端操作系统实施Kernel-RBAC，其关键点就在于对系统调用的截取和控制。

结束语 本文基于RBAC思想，在信息终端的系统内核层面构建了对系统进程的行为进行访问控制的安全强化模型Kernel-RBAC，给出了该模型的具体实现框架，并探讨了如何将该框架实施在不同类型的信息终端操作系统中。该安全框架的实施不仅局限于桌面系统，同时也支持嵌入式移动设备，因此，在普适计算相关技术日趋成熟、智能移动信息终端逐渐普及的情况下，该安全框架能够为提高终端操作系统的安全性提供一个有效的解决方案，对个人、企业的重要信息提供更可靠的安全保障。

参考文献

[1] 李刚，孙红梅，李智，等. 资源受限Web服务[J]. 计算机学报，2010，33(2)：193-207

[2] Ferraiolo D，Kuhn R，Sandhu R，et al. Role Based Access Control (RBAC) and Role Based Security[EB/OL]. http://csrc.nist.gov/groups/SNS/rbac/，2009

[3] 马勇，卿斯汉，贺也平. 一种基于RBAC的电子商务匿名性与可追究性实现方案[J]. 计算机科学，2007(7)：86-89

[4] Li Hong-xin，Guan Ke-qing. RBAC modeling based on B/S architecture and the application in SET payment system[A]//Fei Yu. Proceedings of the 2009 International Symposium on Information Processing[M]. OULU，FINLAND：ACADEMY PUBLISHER，2009：387-390

[5] 单智勇，孙玉芳. 一个应用于操作系统的RBAC模型及其实施[J]. 计算机研究与发展，2004(2)：287-298

[6] 王亚民. 基于RBAC的信息系统访问控制模型[J]. 情报杂志，2005(10)：43-45

[7] 曾忠平，李宗华，卢新海. 基于RBAC的电子政务信息资源访问控制策略研究[J]. 情报杂志，2007(10)：39-41

[8] 施军，朱鲁华，尤晋元，等. 可定制的安全操作系统内核[J]. 计算机工程，2001，27(8)：66-68

[9] 胡大磊，周学海. 平台无关的的访问控制框架研究与实现[J]. 计算机系统应用，2010，19(3)：17-20

[10] 罗海，安世全. 网格访问控制及对RBAC模型扩展的研究[J]. 重庆邮电大学学报：自然科学版，2008，20(6)：714-718