安全操作系统中系统分权的实现*)

金 雷 林志强 茅 兵 谢 立
(南京大学软件新技术国家重点实验室 南京210093)

Impose Privilege-Divided Policy on Security-Enhanced Operation System

JIN Lei LIN Zhi-Qiang MAO Bing XIE Li
(National Key Laboratory for Novel Software, Nanjing University, Nanjing 210093)

Abstract In Linux management is still based on some privilege system call, and this is a great underlying threat to system. Privilege-Divided Model (PDM) comes out to solve the problem. In the model, privilege is divided into some parts, each part or several parts can only do some special task which need privilege. So the potential threat is cut down by this mechanism. This paper gives an analysis on the idea of privilege-divided, and illuminates how to design and get privilege-divided system into reality.

Keywords Privilege-divided, Security, Access control, Operation system

1 引言

基于BLP模型构建的强制访问控制系统都存在一个问题:在实际应用中,存在多用户(通常是不同级别的用户)之间存在信息共享,这意味着将违反信息单向流动性原则。为了解决这样一个问题,安全操作系统中通常采用了折衷的方案: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。这实际上为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。其二,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。其三,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带来危害。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。

实际上,计算机系统中几乎所有带有危害性的行为可分为两类,一类是通过篡取某些用户帐号来获取特权,典型的例子有缓冲区溢出和特洛依木马;而另外一类是由于用户使用不当而造成对其拥有的合法权力的滥用,这实际上是一种无意危害行为。而究其根源,我们就会发现这两种危害性行为都具有一个共同特征:由于操作系统对特权分配的粒度过粗而导致危害的产生。

尽管没有完全解决这些问题的办法,但是我们可以考虑减少需要超级用户特权的子系统的个数以接近我们的目标。这里有几种解决方法:一是从系统中剔除需要超级用户特权的子系统的个数,二是对系统进行一定的调整,将超级用户的权限下放,即使得一般用户就能够行使超级用户的权限。一般说来,由于多用户系统本身的特性,我们几乎不能将超级用户从整个系统中剔除,所以第一种方案不可行;而第二种方案实际上还保留了超级用户。为此我们提出一种折衷的方案: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,我们称之为子Root,将分类权限分配给各个子Root,同时完全剔除原有系统中的超级用户。这样,我们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。这就是基于特权分化模型PDM(Privilege-Divided Model)的分权系统的基本思想。

2 系统分析

2.1 系统需求分析

为了更为清楚地介绍该系统的思想,我们先给出如下几个定义:

定义1(计算机信息系统可信计算基TCB, Trusted Computing Base of Computer Information System) 计算机系统内保护装置的总体,包括硬件、固件、软件和负责执行安全策略的结合体。它建立了一个基本的保护环境并提供一个可信计算系统所要求的附加用户服务。

定义2(最小特权) 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。当进程的访问要求变化时,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。

对于超级用户所涉及的权力范围,我们可以把它划分为两个部分:特权部分和一般权限部分。我们关心的是与系统紧密相关的特权部分。根据权限粒度细化的指导思想,我们将特权部分化为几个不相交的部分,即如式(1)、(2)所示。

$$P_{Root}=\bigcup_{i=1}^{n}P_i \tag{1}$$

$$\forall P_i,P_j,0<i,j<n,i\neq j,P_i\cap P_j=\Phi \tag{2}$$

其中,P_{Root}表示超级用户的特殊权限的全集,而P_i、P_j均为P_{Root}的子集。假设对这个全集的一个划分的秩为n。具体含义如图1和图2所示。

为了获得有效而完整的划分,我们需要对特权的范围加以考察。所谓特权,在操作系统中主要体现在可执行程序和系

*)国家高技术研究发展计划(863计划),基于Linux的操作系统安全增强技术的研究与开发课题编号:2001AA144010。金 雷 硕士研究生,研究方向为网络安全及系统安全。林志强 硕士研究生,研究方向为网络安全及系统安全。茅 兵 教授,研究方向为人机交互、分布计算和并行处理系统安全。谢 立 博导,教授,分布式计算和系统安全。

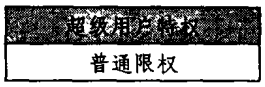
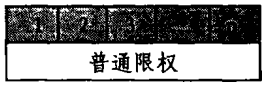


图1 原有系统的权限系统示意图

统调用，其中，前者还包括系统管理命令和 Daemon。根据所涉及的客体不同和特权最小原则，我们将它们划分为五类，分别是安全管理、日志管理、网络管理、系统维护和日常管理。下面的两张表就是我们的分类结果。



注：图中的数字表示部分即为左图中阴影部分的垂直划分

图2 分权系统的权限系统示意图

表1 /sbin 和/use/sbin 下可执行程序的分类结果

特权分类	涉及的可执行程序
安全管理	adduser chpasswd groupadd groupdel groupmod grpck grpconv grpunconv mac mkpasswd newusers pwck pwconv pwunconv useradd userdel userhelper usermod usernetctl vigr vipw
日志管理	initlog kolgd syslogd loglevel minilogd logrotate
网络管理	arp ifconfig ifdown ifport ifup ifuser ipchains ipchains-restore ipchains-save ipfwadm ipfwadm-wrapper ipmaddr iptunnel mailconf netreport ppp-watch pump rmt route rpc. lockd rpc. statd ypbind identd in. fingerd in. ftpd in. identd in. ntalkd in. rexecd in. rlogind in. rshd in. talkd in. telnetd in. tftpd in. wuftpd inetd ipvsadm kppp mailstats makemap netconfig pmap_dump pmap_set praliases routed rpcinfo rpc. rstatd rpc. rusersd safe_finger sendmail snmpd snmptrapd tcpd tcpdchk tcpdmatch tcpdump traceroute yppoll ypset
系统维护	cardctl cardmgr dempod genksyms insmod insmod. static installkernel isapnp kbdrate ksyms losetup lsmod lspci mkdosfs mke2fs mkfs mkfs. ext2 mkfs. minix mkfs. msdos mkinitrd mkpv mkraid modinfo modprobe mount. smb mount. smbfs pcinitrd pnpdump probe raid0run raidhotadd raidhotremove raidstart raidstop rmmod rmmod. static scsi_info setpci gpm kbdconfig kudzu lpc lpd lpf lsof mouseconfig pnpprobe readprofile setup sndconfig
日常管理	askrunlevel badblocks cfdisk chkconfig clock ctrlaltdel debugfs dnsconf dosfsck dump dumpe2fs e2fsck e2label fdisk fixperm fsck fsck. ext2 fsck. minix fsck. msdos ftl_check ftl_format fuser getkey getty halt hdparm hwclock init install-info kerneld killall5 lilo mingetty mkbootdisk mkswap pidof plipconfig portmap poweroff pwdb_chkpwd quotacheck quotaoff quotaon rarp rdump reboot restore rrestore runlevel service setsysfont sfdisk shutdown stinit sulogin swapoff swapon sysctl telinit tune2fs apmd atd atrun authconfig chkfontpath chroot crond edquota kpackage lvs mklost + found nanny ntsysv pulse quotastats ramsize rdev repquota rootflags rwhod setclock setconsole showmount swapdev sys-unconfig timeconfig timed timedc tmpwatch unstrutempter utmpd vidmode zdump zic

表2 涉及系统安全的系统调用分类结果

特权分类	涉及的系统调用
安全管理	setuid getuid setgid getgid geteuid getegid setpgid setfsuid setsid getgroups getpgrp setresuid setreuid getresuid setregid getpgid setfsgid setresgid setresgid
日志管理	
网络管理	sethostname gethosyname socketcall setdomainname
系统维护	mknod mout umount umount2ioctl ioperm iopl init_module delete_module query_module get_kernel_syms create_module
日常管理	stime stty gtty gettimeofday settimeofday adjtimex sysinfo reboot swapoff uname setrlimit getrlimit getrusage sysctl quotactl

如本章第1节所述，根据划分的结果和特权最小原则，我们将生成若干个相应的子 Root 来分别负责各自的权力范围。下面就是我们的划分结果。

1. 安全管理员：从系统中添加、删除用户账号和组账号；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 MAC 开关；添加用户的安全标记，获得、改变原有用户的安全标记；可以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。

2. 日志管理员：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。

3. 网络管理员：网络设备的安装、设置、启动、停止；网络设备模块的加载与卸载；对普通用户使用网络的授权（包括网络设备、网络协议、网络服务等）；网络状态的监视。

4. 系统维护员：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（网络设备模块除外）；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、更改系统核心等；维护系统引导扇区。

5. 日常管理员：除上述四个管理员外，原系统的 root 剩余的管理功能归该日常管理员所有。具体来讲，主要有如下的几项功能：系统时钟设置；设置系统运行级别；系统安装、备份、恢复；软件的安装、设置、升级或删除；磁盘分区与规划；文件系统的维护（如查错等）；交换分区的设置与管理。

3 系统设计

3.1 定义

与其他安全技术一样，分权系统同样应该清晰地定义系统涉及的主客体及其属性：

定义3（被控客体，controlled objects） 分权系统涉及的客体主要包括配置文件（configure file）、管理命令及后台进程（executable & daemon）、系统调用（system call）、设备及/proc 文件系统（device &/proc）、特权指令（privileged instruction）等。

定义4（被控客体的属性） 此处的被控客体的属性是与安全相关的属性，系统凭借这些属性对被控客体的访问进行控制，这些属性也成为可信系统内被控客体的标识。它主要包括文件名称、系统调用号、主从设备号和信号量等。

定义5（控制主体，controlling subjects） 分权系统涉及的主体主要包括安全管理员、日志管理员、网络管理员、系统维护员和日常管理员共五个管理员（UID）和管理命令进程及后台进程（executable & daemon）。

定义6(控制主体的属性) 此处的控制主体的属性是与安全相关的属性,系统凭借这些属性对控制主体的行为进行判断与控制,这些属性也成为可信系统内控制主体的标识。它主要有以下三种属性:用户标识(特别是管理员)UID、进程capability 和可执行文件标识(the Executable)。

3.2 系统的分权描述

在确定系统的主客体之后,就要对系统的分权特性进行描述,这也是整个系统的中心所在。下面是对各种被控客体的控制规则。

1)特定的配置文件(主要是/ect 下的文件)或数据文件(如日志和审计数据文件)只能由特定的管理员来访问,并且该文件的属主即为该管理员。通过特定的管理程序,即这些文件只能由相应的管理程序来访问,从而体现管理员相应的特权。

2)特定的系统调用(或称为特权系统调用,这些系统调用在传统的 Linux 系统中需要 Root 权限)只能由特定的管理员来调用(其体现是通过相应的管理程序来实现的,即这些特权调用只能有相应的管理程序来执行),或者一些系统调用虽然其本身并非特权调用,但它们的一些特殊参数需要特殊管理权限,这时需要相应管理员的权限。特权指令只能由特定的管理员(通过特定的管理程序)来使用。

3)特定的管理命令(通常是/usr/sbin 及/sbin 目录下的可执行文件)只能由特定的管理员来执行,这些命令的文件属主也为该管理员。

4)特定的设备特殊文件(虚/实两种设备)只能由特定的管理员通过特定的管理程序来访问。

5)特定的后台进程(daemon)以特定的系统管理员的权限或 daemon 用户的权限(该用户为非终端用户,即不可以从终端登陆进来的用户)进运行。

3.3 分权系统访问控制描述

系统分权,主要体现在上一节中所描述的控制主体和被控客体,以及它们之间的访问控制关系上。本小节从被控客体的角度(即被控客体的访问控制流程)来阐述系统分权的特性。

3.3.1 *配置文件访问控制* 进程对配置文件的访问主要体现在对 open()以及 chown()、chmod()调用簇的调用上。因此,对配置文件的访问控制主要体现在对上述系统调用的控制,具体的控制流程见图3:

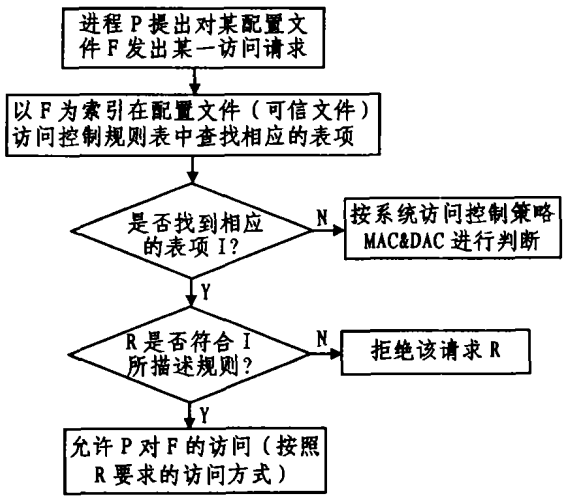


图3 配置文件访问控制流程图

3.3.2 *管理命令及后台程序启动控制* 管理命令的调用以及后台进程的启动的主要体现在 exec()系统调用簇的调用上,因此对他们的控制也就主要体现在对该簇系统调用的控制上,具体情况见图4:

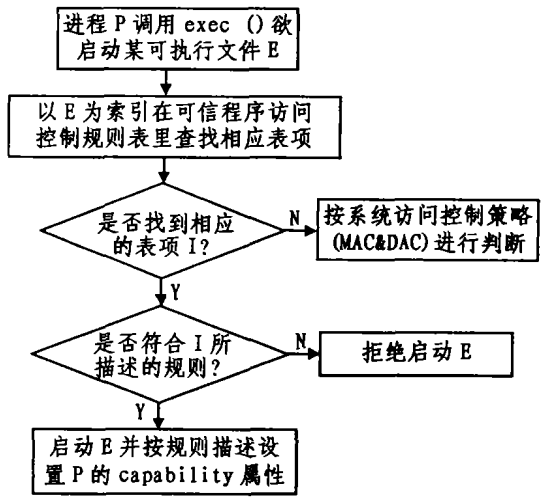


图4 管理命令及后台程序启动控制流

3.3.3 *系统调用访问控制* 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;对于特权调用(包括普通调用的特权参数),则按照图5的描述进行访问控制;

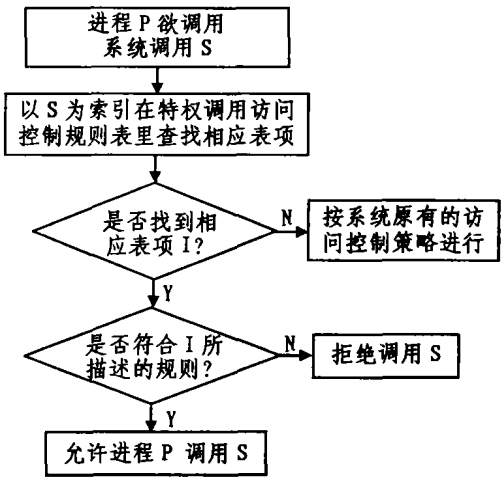


图5 系统调用控制流程图

3.3.4 *设备及/proc 文件访问控制* 几乎系统中所有的用户都需要使用设备,但普通的用户一般仅仅是利用设备进行数据的输入与输出;对于管理员而言,其管理职责主要体现在对设备的属性设置。在具体的管理过程中,系统核心级通过 ioctl()系统调用来实现。因此对设备的访问控制主要体现在对该系统调用各种参数的控制,这种控制主要由相应的规则来描述,流程图类似图3(以设备的主、次设备号为索引来查找相应的规则),在此处略过。

/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来访问核心数据。对它的访问控制类似于对配置文件的控制,流程图就此略过。

3.3.5 *系统启动* 系统核心加载并初始化后,系统启动 init 进程作为用户空间所有进程的祖先进程。init 进程将根据相应的配置文件对系统进行配置,并启动各个后台进程。在此期间,牵涉到各个管理员的管理权限问题。因此在启动过程中,需要相应的管理员权限时,要求该管理员进行身份/口令

认证。认证通过后，由该管理员确认是否需要该项配置。流程图较为简单，此处略过。

3.3.6 文件、目录属性设置 对于文件、目录属性的设置主要是通过 chown()和 chmod()调用簇来实现的。在原系统中，只有文件/目录主才可设置相应的属性，而超级用户可以任意更改文件、目录的属性。在安全操作系统中，对配置文件等重要文件的属性设置被看作是对该文件(或目录)的一种访问，应符合相应规则。五个管理员不允许对任意文件、目录的属性进行任意改动，须符合一定的控制规则。文件、目录的属性设置许可有两种情况：一种是文件、目录的属性不可更改，即任何用户包括管理员都不可对其进行更改；另一种是保留方式，即按照系统原有的属性设置策略进行控制。

4 分权系统中访问控制规则

针对系统分权中不同的被控客体，系统中使用相应的控制规则来控制某被控客体的访问。有些规则中的字段仅仅是定义，为系统保留，目前系统暂不实现相应的功能。以下各小节主要描述各个规则的组成成分及相关语义。

4.1 配置文件访问控制规则

配置文件的访问控制规则由下述结构描述：

UID	EXEC	MODE	ATTR

其中各部分内容的含义为：

UID：该规则所要求的用户标识，一般为相应的管理员的 UID(可扩展为一个用户标识集合，即包含多个用户)

EXEC：该规则所要求的可执行文件，一般用于实现指定可执行文件访问指定(配置)文件的这一类特殊的访问控制技术，使用该技术可以取代系统原有的 SUID/SGID 功能，而不影响系统原有的功能。

MODE：该规则所要求的访问模式，每个模式占用模式字段的一位，主要有以下几类访问模式(可参照 open()系统调用的 flag 参数)：

MOD_RDONLY_ 只读模式
MOD_WRONLY 只写模式
MOD—RDWR 读写模式
MOD_APPEND 附加模式
MOD_DELETE 可删除

上述的访问模式可以通过逻辑或操作形成复合模式。

ATTR：该规则的属性，每个属性占用属性字段的一位(2值逻辑)，合法的配置文件访问控制规则属性如下：

ATTR_CONST：表示该规则为常规则，即在系统的运行期间该规则不允许被修改；

ATTR_SUBTREE：该属性用于目录文件，即表示该规则适用于在该目录中的所有文件，包括子目录中文件(该规则可以被子树中所有的对象继承)；

ATTR_CONSTMOD：表示该规则适用的配置文件的属性不可被更改；

ATTR_EXCLUSIVE：受该规则控制的配置文件不可接受其他规则的控制；

ATTR_OVERDAC：表示该规则可以跨越系统 DAC，即对相应的配置文件的访问受该规则的约束，而不受系统 DAC 的约束；

ATTR_OVERMAC：表示该规则可以跨越系统 MAC，即对相应的配置文件的访问受该规则的约束，而不受系统 MAC 的约束；

ATTR_OWNER：表示对相应的配置文件的访问只能由其属主来进行；

上述几个属性可以通过逻辑或操作形成复合属性。

4.2 管理命令与后台进程启动控制规则

管理命令与后台进程访问控制规则由下述结构描述：

UID	CAP	ATTR	SYSCALL_MAP	CONF_LIST

其中各部分内容的含义为：

UID：该规则所需求的用户标识，一般为相应的管理员的 UID(可扩展为一个用户标识集合，即包含多个用户)。

CAP：当该规则适用的可执行文件被启动时，则按照该字段的内容设置进程的 capability 位。

ATTR：该规则的属性字段，合法的属性与配置文件访问控制规则中的 ATTR 定义相同，ATTR_OWNER 属性的含义与它有所不同，在此处是指受该规则控制的管理命令与后台进程只能由其属主来启动。增加的属性有：

ATTR_GRANU：粒度位，表示涉及到的系统调用的单位为类。否则为个。

ATTR_SYSCALL：系统调用涉及方式。若该属性被设置，则系统调用映射图中的系统调用为合法的系统调用，即当该程序试图调用在该图以外的系统调用时，将导致失败；若该属性位没有设置，则映射图中的系统调用为非法系统调用，即当该程序试图调用该图中的系统调用时，将导致失致。

SYSCALL_MAP：系统调用映射图，每一位用来标识该规则涉及的一个或一类系统调用(取决于属性字段的粒度位)，1表示涉及到该类系统调用，0表示不涉及，涉及方式取决于属性字段。

CONF_LIST：受该规则控制的管理命令与后台进程所允许访问的配置文件的列表。当相应的管理命令与后台进程在试图访问某个后台进程时，系统会检查该列表，只允许出现在该列表中的文件被访问。若该字段的内容为空，则对任意的配置文件都可进行访问；此外，能否真正访问成功，还要取决于系统其他的访问控制规则。

4.3 系统调用访问控制规则

系统调用访问控制规则由下述结构描述：

SYSCALL#	UID	CAP	EXEC	PARAM_LIST	ATTR

各字段的含义是：

SYSCALL#：系统调用号，指明该规则适用的系统调用号。

UID：该规则所要求的用户标识，一般为相应的管理员的 UID(可扩展为一个用户标识集合，即包含多个用户)。

CAP：该规则所要求的 capability，只有请求进程具备规则要求的 capability，相应的系统调用才有可能被允许调用。

EXEC：该规则所要求的可执行程序(管理命令或后台进程)，即该规则所控制的系统调用只能由这个可执行程序来调用。

PARAM_LIST：该规则所允许的调用参数列表；

ATTR：属性字段，在此处合法的属性有：ATTR_CONST，ATTR_EXCLUSIVE，其含义与上文中相同，复合

(下转第178页)

$$T_{31}(M_3)=(M_3+\sum_{i=1}^{\bar{k}}C_{3i}\times L_{3i})/\sum_{i=1}^{\bar{k}}C_{3i} \quad (11)$$

定理的证明，参见文[2,3]，此略。

根据该定理，当储丝柜物料量 M_3 给定时，依次运用式(9)、(10)、(11)，即可求得这些物料从“储丝柜→卷接包线→产品”所经历的最短时间。

4 卷烟生产线柔性优化调度模型的实际算例

以下以重庆烟草工业有限责任公司下属的涪陵卷烟厂为背景，给出实际算例。

1. *获得卷烟生产线工艺参数值* 通过对该厂卷烟生产线现场考察和分析，得到有关工艺参数值如下：

$L_{01}=35(m)$；$L_{11}=20(m)$；

$L_{2i}=25(m)$；$L_{3i}=15(m)$ $(i=1\sim7)$；

$L_{3i}=20(m)$ $(i=8\sim16)$；

$C_{01}=1500\times i$ $(i=1,2,3,\cdots)$；

$C_{11}=2250(kg/h)=37.5(kg/m)$

$C_{21}=3000(kg/h)=50(kg/m)$；

$C_{3i}=7000(支/m)=7000\times0.85/1000(kg/m)$

$=5.95(kg/m)$ $(i=1\sim7)$

$C_{3i}=2500(支/m)=2000\times0.85/1000(kg/m)$

$=1.7(kg/m)$ $(i=8\sim16)$

2. *计算满足(9)式的最大 k 值 $\bar{k}$* 分别将 k=1,2,3,…,16代入(9)式左边，求得相应的值，如表1所示。

根据表1可知，当储丝柜的物料量 M3<208(kg)时，$\bar{K}=7$，即只用7条 PASSIM 卷接线进行生产时，所需生产时间最短；否则，$\bar{K}=16$，即用16条卷接线同时生产时，所需生产时间最短。

3. *计算生产线上各过程物料流动的最优时间* 根据算式(1)、(2)、(3)、(15)，并针对不同的批量数，得到生产线上各过程物料流动的最优时间，如表2所示。

表1 不同的 K 值所对应的式(9)左边算式值明细表

K 值	算式值	K 值	算式值	K 值	算式值	K 值	算式值
1	0	5	0	9	208.25	13	208.25
2	0	6	0	10	208.25	14	208.25
3	0	7	0	11	208.25	15	208.25
4	0	8	208.25	12	208.25	16	208.25

表2 不同的批量物料流完不同生产过程所需的最优时间表

物料量	物料流完各生产过程所需的最优时间(m)			
(kg)	生产过程1	生产过程2	生产过程3	生产过程4
1500	40	60	55	40
3000	80	100	85	69
4500	120	140	115	95
6000	160	180	145	122

结论 根据上述研究，得出以下结论：

(1)根据实际卷烟生产线的生产现场情况，分析了生产线的工艺流程；

(2)利用现代图论的基本思想，将卷烟生产线的工艺流程抽象成了基于连续物料流的连通网络系统，并由此将卷烟生产线分解成4个生产过程；

(3)根据现场工艺和生产实际，提出了“柔性优化调度”的概念；

(4)针对上述的4个生产过程和柔性优化调度的思想，分别建立了优化调度数学模型，并给出了实际算例。

参 考 文 献

1 陈庄，邓万先，等. 卷烟生产线优化模型研究. 重庆工学院学报，2000,14(3):1～7

2 Chen Y L, Chin Y H. The quickest path problem. Computers Ops Res.,1990,17(2):153～161

3 Resen J B, Sun S Z. Algoriths for the quickest path problem and enumeration of quickest paths. Computers Ops Res.,1991,18(6):579～584

4 戴建设，王书宁，杨小茵. 连续型多通道最速路问题研究. 自动化学报，1995,21(3):274～279

(上接第157页)

属性的形成方式也同上文。

4.4 设备文件与/proc 文件系统的访问控制规则

设备文件与/proc 文件系统的访问控制规则由下述结构描述：

DEV	OPER	IO_PARAM	UID	CAP	EXEC	ATTR

各字段的含义是：

DEV：该规则所控制的设备。

OPER：该规则所允许的在上述设备上执行的操作映射图，每一位表示一种操作。

IO_PARAM：该规则所允许的在此设备上调用 ioctl()系统调用所使用的参数(在 OPER 字段中包含 ioctl()操作)。

UID：该规则所允许的访问此设备的用户标识。

CAP：该规则所要求的访问此设备必需的 capability。

EXEC：允许访问该设备的可执行文件(管理命令或后台进程)。

ATTR：规则属性字段，其合法取值参照配置文件访问控制规则中相应的定义。

/proc 文件系统的访问控制规则类似于配置文件，主要的访问控制规则是特定的可执行程序访问相应的文件，此处控制规则描述略过。

参 考 文 献

1 陈家民，于康友，管海民，等. 计算机的安全与保密. 电子工业出版社，1992

2 A Security Policy Configuration for the Security-Enhanced Linux Stephen Smalley, NAI Labs, Timothy Fraser, NAI Labs, Feb. 2001

3 Forrest S, Hofmeyr S A, Somajayi A. A Sense of Self for Unix Processes. In: Proc. of 1996 IEEE Symposium on Computer Security and Privacy, IEEE Press, 1996

4 公安部计算机管理监察司. 计算机信息系统安全技术. 北京：群众出版社，1998

5 Zhong Q. Providing secure environments for untrusted network applications-with case studies using virtual vault and trusted sendmail proxy. In: Proc. of Second IEEE InternationalWorkshop onEnterprise Security, Los Alamitos, CA, IEEE CS Press, 1997. 277～283