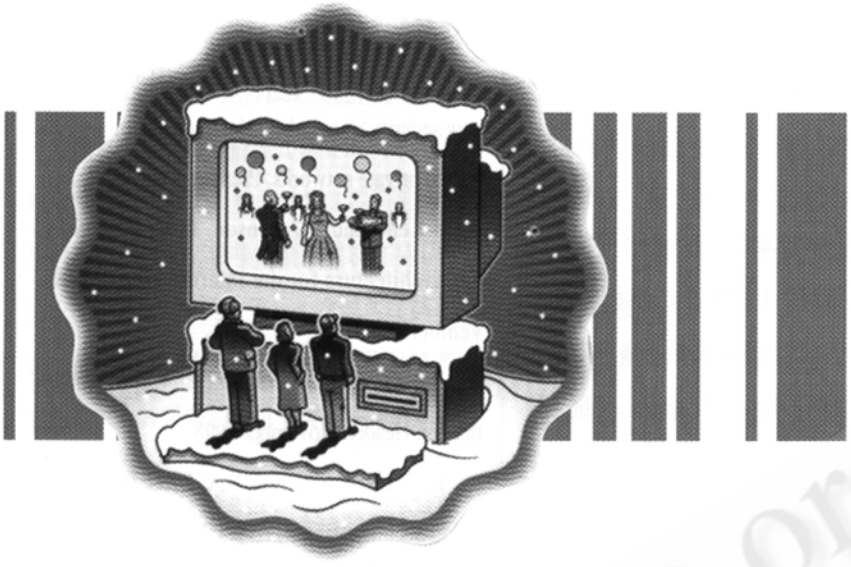

Unix/Linux 操作系统安全(四)

卿斯汉 (中科院信息安全技术工程研究中心 100080)

3.2 进程子系统

进程系统的安全基础是处理器和内存管理的支持。一般处理器至少支持用户态和核心态两种模式。在内存管理方面，现在普遍采用虚拟存储技术，下面以Linux为例。

虚拟内存系统中的所有地址都是虚拟地址而不是物理地址。通过操作系统所维护的一系列表格由处理器实现由虚拟地址到物理地址的转换。

为了使转换更加简单，虚拟内存与物理内存都以页面来组织。不同系统中页面的大小可以相同，也可以不同，这样将带来管理不便。Alpha AXP处理器上运行的Linux页面大小为8KB，而Intel X86系统上使用4KB页面。每个页面通过一个叫页面框号的数字来标示(PFN)。

页面模式下的虚拟地址由两部分构成：页面框号和页面内偏移值。如果页面大小为4KB，则虚拟地址的11：0位表示虚拟地址偏移值，12位以上表示虚拟页面框号。处理器处理虚拟地址时必须完成地址分离工作。在页表的帮助下，它将虚拟页面框号转换成物理页面框号，然后访问物理页面中相应偏移处。

理论上每个页表入口应包含以下内容：

有效标记，表示此页表入口是有效的

页表入口描叙的物理页面框号

访问控制信息，用来描叙此页可以进行哪些操作，是否可写？是否包含执行代码？

为了将虚拟地址转换为物理地址，处理器首先必须得到虚拟地址页面框号及页内偏移，一般将页面大小设为2的次幂。

处理器使用虚拟页面框号为索引来访问处理器页表，检索页表入口。如果在此位置的页表入口有效，则处理器将从此入口中得到物理页面框号。如果此入口无效，则意味着处理器存取的是虚拟内存中一个不存在的区域。在这种情况下，处理器是不能进行地址转换的，它必须将控制传递给操作系统来完成这个工作。

通过将虚拟地址映射到物理地址，虚拟内存可以以任何顺序映射到系统物理页面。

页表入口包含了访问控制信息。由于处理器已经将页表入口作为虚拟地址到物理地址的映射，那么可以很方便地使用访问控制信息来判断处理器是否在以其应有的方式来访问内存。

诸多因素使得有必要严格控制对内存区域的访问。有些内存，如包含执行代码的部分，显然应该是只读的，操作系统决不能允许进程对此区域的写操作。相反包含数据的页面应该是可写的。但是去执行这段数据肯定将导致错误发生。多数处理器至少有两种执行方式：核心态与用户态，任何人都不会允许在用户态下执行核心代码或者在用户态下修改核心数据结构。页表入口中的访问控制信息是和处理器相关的。

Alpha AXP页表入口如图6所示。

这些位域的含义如下：

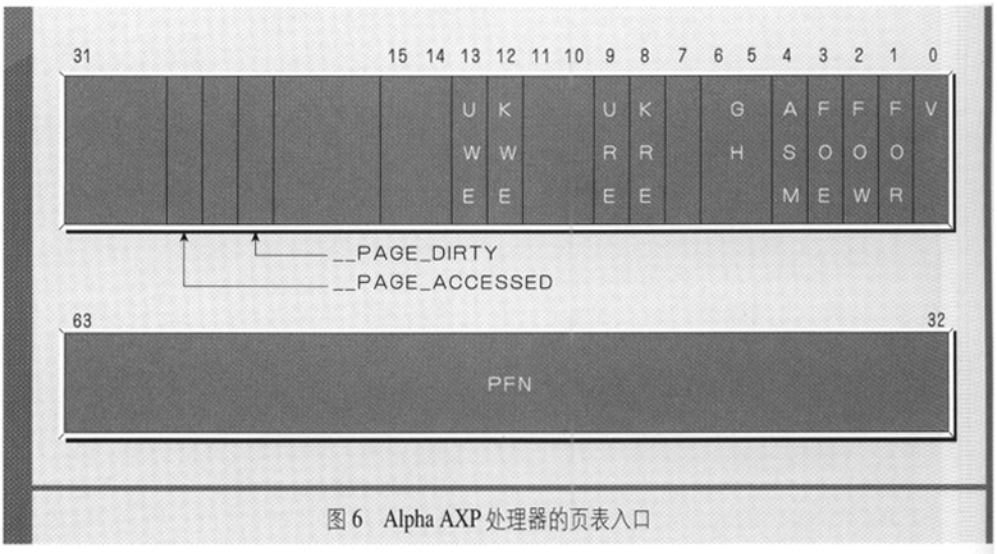

图6 Alpha AXP处理器的页表入口

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

V——有效，如果此位置位，表明此PTE有效

FOE——"执行时失效"，无论合时只要执行包含在此页面中的指令，处理器都将报告页面错误并将控制传递。

FOW——"写时失效"，除了页面错误发生在对此页面的写时，其他与上相同。

FOR——"读时失效"，除了页面错误发生在对此页面的读时，其他与上相同。

ASM——地址空间匹配。被操作系统用于清洗转换缓冲中的某些入口。

KRE——运行在核心模式下的代码可以读此页面。

URE——运行在用户模式下的代码可以读此页面。

GH——将整个块映射到单个而不是多个转换缓冲时的隐含粒度。

KWE——运行在核心模式下的代码可以写此页面。

UWE——运行在用户模式下的代码可以写此页面。

page frame number——对于V位置位的PTE，此域包含了对应此PTE的物理页面框号；对于无效PTE，此域不为0，它包含了页面在交换文件中位置的信息。

以下两位由Linux定义并使用。

-PAGE-DIRTY——如果置位，此页面要被写入交换文件。

-PAGE-ACCESSED——Linux用它表示页面已经被访问过。

进程管理子系统控制进程的创建、终止、记账以及调度。它监视进程的状态变化以及核心态和用户态之间的切换。进程子系统的安全设计使得Unix系统实现了两种执行模式-具有较高特权的内核态和特权较低的用户态。实现用户程序在用户态执行，内核功能在内核态执行。由于用户进程在较低的特权级上运行，他们将不能意外地或故意地破坏其他进程或内核，程序错误造成的破坏被局域化。

Unix系统中，每个进程都具有一个固定的结构，内核将此结构称为进程的映像。进程的二进制映像既包括用户地址空间也包括内核地址空间，如图7图8所示。

内核地址空间只能在内核态访问，系统中只有一个内核实例运行，因此所有进程都映射到单一内核地址空间。尽管所有的进程都共享内核，但内核空间是受保护的，进程在用户态是不能访问的。进程不能直接访问内核，而必须通过系统调用接口。当进程调用一个系统调用(system call)时，其执行了一个特殊的指令序列，使系统进入内核态（实现模式转换），并将控制权交给内核，由内核代替进程完成操作。当系统调用完成后，内核执行另一组特征指令将系统返回到用户态（实现另一个模式转换），控制权返回给进程。

当进程上下文中，进程在用户态执行，可以访问用户空间；在内核态时，内核代表当前进程执行（例如，执行一个系统调用），此时，内核通过地址转换表可以直接访问当前进程的地址空间。计算机存储管理单元（MMU）一般有一组存储器来标识当前进程的转换表，在当前进程将CPU放弃给另一个进程时（一次上下文切换），内核通过指向新进程地址转换表的指针加载这些寄存器。MMU寄存器时有特权的，只能在内核态访问。这保证了一个进程只能访问自己用户空间内的地址，而不会访问和修改其他进程的空间，请参看图9。

内核也必须完成系统级任务，如中断。这些任务并不是为了特定的进程完成的，因此在系统上下文（中断上下文中）处理。这时，内核不会访问当前进程的地址空间，如图10所示：

3.3 系统调用

当要执行一个文件时，文件子系统要和进程控制子系统交互。进程通过系统调用来对文件进行操作，也可以通过C语言的函数调用来操作文件，而C的函数库使用系统调用来实现这些函数调用。所以，系统调用是两个执行模式转换的关键，也是系统安全的关键。

本节介绍一些常用的和安全有关的系统调用，用户使用时要注意其安全属性，实现安全

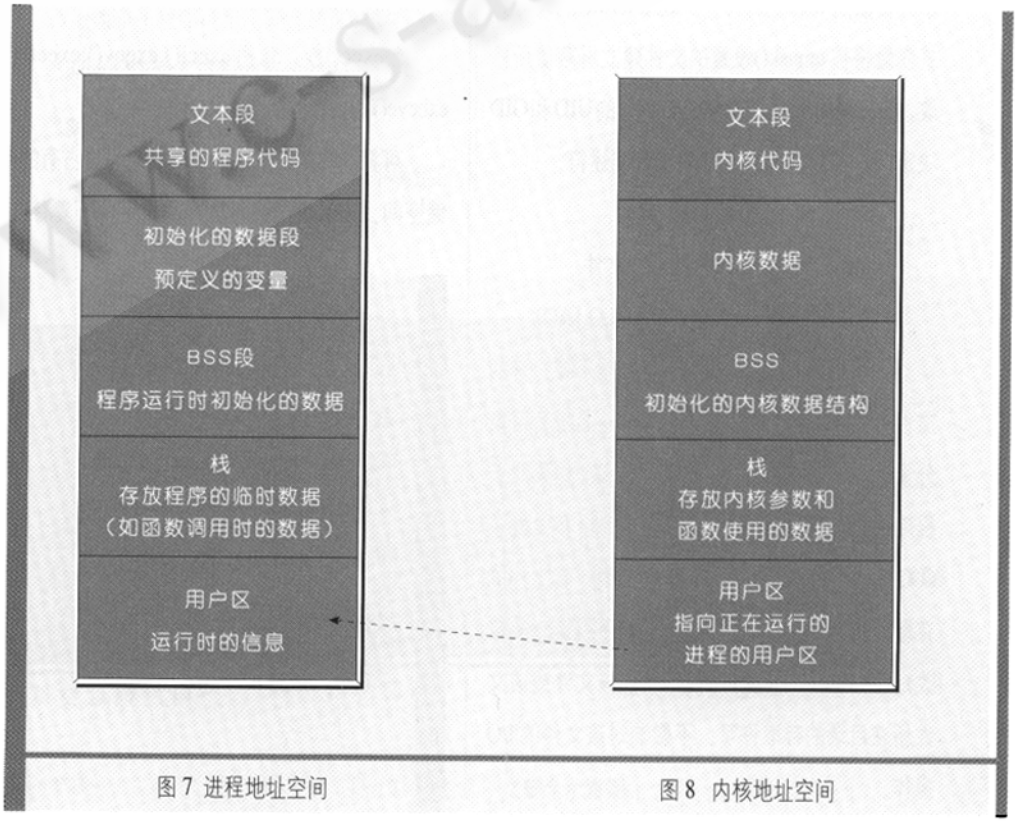

图7 进程地址空间　　图8 内核地址空间

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

操作。

I/O 子程序

* creat():建立一个新文件或重写一个暂存文件

参数:文件名和存取许可值(8 进制方式)。如:creat("/usr/pat/read-write", 0666) /* 建立存取许可方式为 0666 的文件 */

调用此子程序的进程必须要有建立的文件的所在目录的写和执行许可，置给creat()的许可方式变量将被umask()设置的文件建立屏蔽值所修改，新文件的所有者和小组由有效的UID和GID决定。返回值为新建文件的文件描述符。

* fstat(): 见后面的stat()

* open(): 在C程序内部打开文件

参数:文件路径名和打开方式(I,O,I&O)

如果调用此子程序的进程没有对于要打开的文件的正确存取许可(包括文件路径上所有目录分量的搜索许可)，将会引起执行失败。如果此子程序被调用去打开不存在的文件，除非设置了O-CREAT标志,调用将不成功。此时，新文件的存取许可作为第三个参数(可被用户的umask修改)。当文件被进程打开后再改变该文件或该文件所在目录的存取许可，不影响对该文件的I/O操作。

* read(): 从已由open()打开并用作输入的文件中读信息

它并不关心该文件的存取许可。一旦文件作为输入打开，即可从该文件中读取信息。

* write(): 输出信息到已由open()打开并用作输出的文件中

同read()一样它也不关心该文件的存取许可。

进程控制

* exec()族：包括execl(),execv(),execle(),execve(),execlp()和execvp()。

可将一可执行模快拷贝到调用进程占有的存储空间。正被调用进程执行的程序将不复存在，新程序取代其位置。这是UNIX系统中一个程序被执行的唯一方式：用将执行的程序复盖原有的程序。

安全注意事项：

实际的和有效的UID和GID传递给由exec()调入的不具有SUID和SGID许可的程序

如果由exec()调入的程序有SUID和SGID许可，则有效的UID和GID将设置给该程序的所有者或小组

文件建立屏蔽值将传递给新程序

除设了对exec()关闭标志的文件外,所有打开的文件都传递给新程序。用fcntl()子程序可设置对exec()的关闭标志

* fork():用来建立新进程

其建立的子进程是与调用fork()的进程(父进程)完全相同的拷贝(除了进程号外)

安全注意事项:

子进程将继承父进程的实际和有效的UID和GID

子进程继承文件方式建立屏蔽值

所有打开的文件传给子进程

* signal()：允许进程处理可能发生的意外事件和中断

参数：信号编号和信号发生时要调用的子程序

信号编号定义在signal.h中。信号发生时要

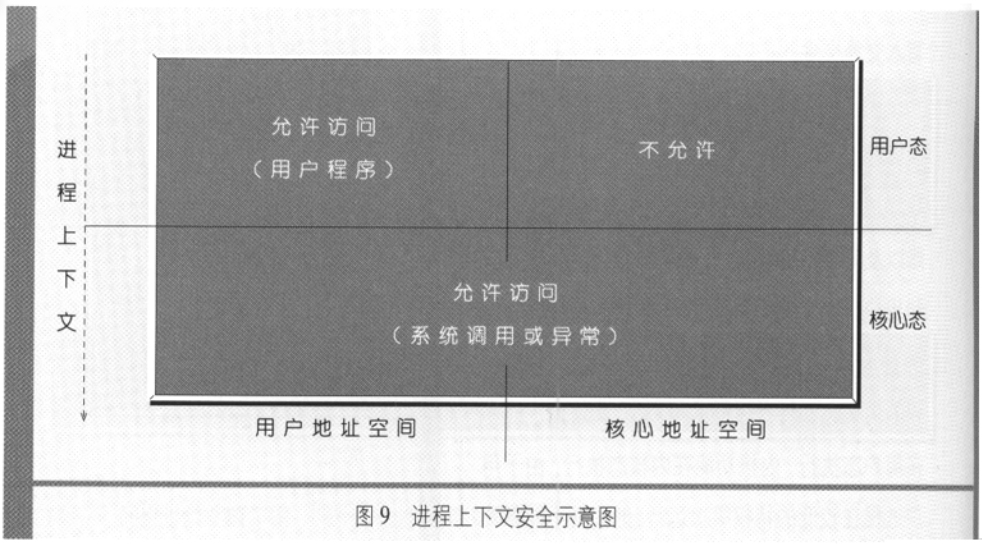

图9 进程上下文安全示意图

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

调用的子程序可由用户编写，也可用系统给的值，如:SIG-IGN则信号将被忽略，SIG-DFL则信号将按系统的缺省方式处理。如许多与安全有关的程序禁止终端发中断信息(BREAK和DELETE)，以免自己被用户终端终止运行，有些信号使UNIX系统的产生进程的核心转储(进程接收到信号时所占内存的内容,有时含有重要信息)，此系统子程序可用于禁止核心转储。

文件属性

* access():检测指定文件的存取能力是否符合指定的存取类型

参数:文件名和要检测的存取类型(整数)

存取类型的数字意义和chmod命令中规定许可方式的数字意义相同,此子程序使用实际的UID和GID检测文件的存取能力(一般有效的UID和GID用于检查文件存取能力)。

返回值: 0:许可 -1:不许可

* chmod():将指定文件或目录的存取许可方式改成新的许可方式

参数:文件名和新的存取许可方式

* chown():同时改变指定文件的所有者和小组的UID和GID.(与chown命令不同)

由于此子程序同时改变文件的所有者和小组,故必须取消所操作文件的SUID和SGID许可,以防止用户建立SUID和SGID程序,然后运行chown()去获得别人的权限。

* stat():返回文件的状态(属性)

参数:文件路径名和一个结构指针,指向状态信息的存放的位置。

返回值: 0:成功 1:失败

*umask(): 将调用进程及其子进程的文件建立屏蔽值设置为指定的存取许可

参数: 新的文件建立屏值

UID和GID的处理

* getuid():返回进程的实际UID

* getgid():返回进程的实际GID

以上两个子程序可用于确定是谁在运行进程

* geteuid():返回进程的有效UID

* getegid():返回进程的有效GID

以上两个子程序可在一个程序不得不确定它是否在运行某用户而不是运行它的用户的SUID程序时很有用,可调用它们来检查确认本程序的确是以该用户的SUID许可在运行。

* setuid():用于改变有效的UID

对于一般用户,此子程序仅对要在有效和实际的UID之间变换的SUID程序才有用(从原有效UID变换为实际UID),以保护进程不受到安全危害。实际上该进程不再是SUID方式运行。

*setgid():用于改变有效的GID

(未完待续)

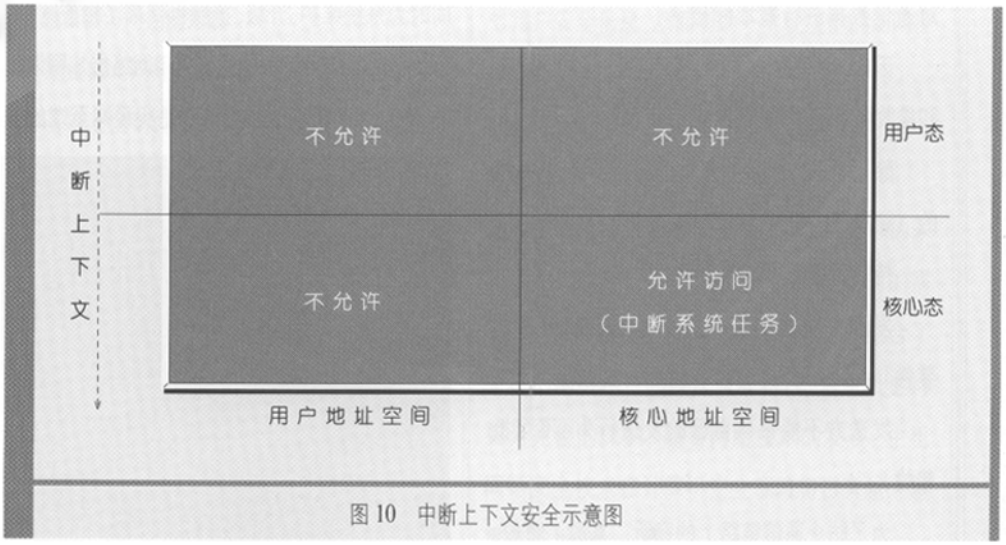

图10 中断上下文安全示意图

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn