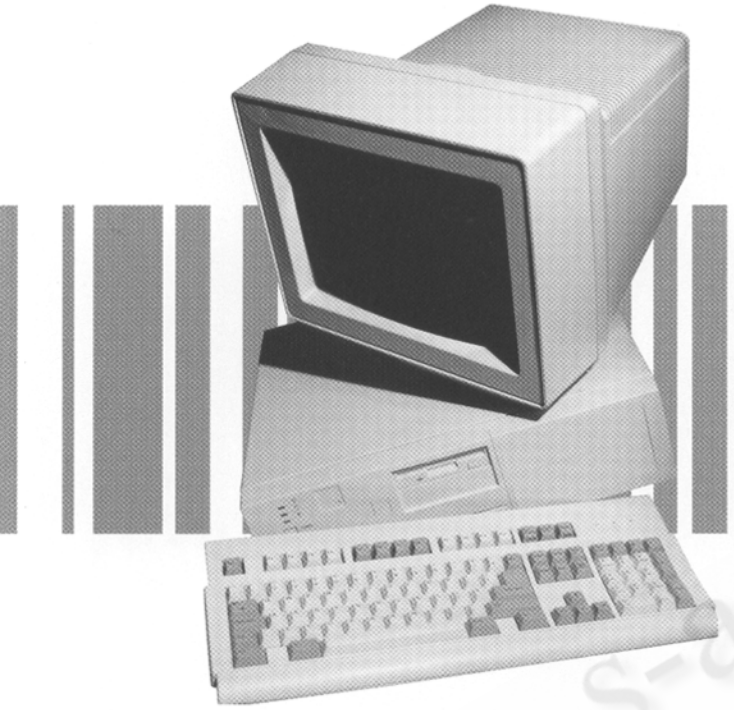

Unix/Linux 操作系统安全(一)

卿斯汉 (中科院信息安全技术工程研究中心 100080)

1 UNIX操作系统安全性概述

Unix是一种多用户、多任务的操作系统。这类操作系统的一种基本功能就是防止使用同一台计算机的不同用户之间互相干扰。所以，Unix的设计宗旨是要考虑安全的。当然，Unix中仍然存在很多安全问题，其新功能的不断纳入及安全机制的错误配置或不经心使用，都可能带来很多问题。

Unix的系统结构由用户、内核和硬件三个层次组成，如图1所示。

UNIX操作系统借助以下四种方式提供功能：

系统调用：用户进程通过Unix API的内核部分－系统调用接口，显式地从内核获得服务。内核以调用进程的身份执行这些请求。

异常：进程的某些不正常操作，诸如除数为0，或用户堆栈溢出将引起硬件异常。异常需要内核干预，内核为进程处理这些异常。

中断：内核处理外围设备的中断。设备通过中断机制通知内核I/O完成状态变化。内核将中断视为全局事件，与任何特定进程都不相关。

像swapper和pagedaemon之类的一组特殊的系统进程执行系统级的任务。比如，控制活动进程的数目或维护空闲内存池。

系统具有两个执行态：核心态和用户态。运行内核中程序的进程处于核心态，运行核外程序的进程处于用户态。系统保证用户态下的进程只能存取它自己的指令和数据，而不能存取内核和其他进程的指令和数据，并且保证特权指令只能在核心态执行，象中断、异常等在用户态下不能使用。用户程序可以通过系统调用进入核心，运行完系统调用，又返回用户态。系统调用是用户在编写程序时可以使用的界面，是用户程序进入Unix内核的唯一入口。因此，用户对系统资源中信息的存取都要通过系统调用才能完成。一旦用户程序通过系统调用进入内核，便完全与用户隔离，从而使内核中的程序可对用户的存取请求进行不受用户干扰的访问控制。本章中，我们从用户管理、内核及系统调用三方面来描述unix操作系统的安全性。

在安全结构上，Linux与Unix基本上是相似的。不同之处，我们会作比较和说明。

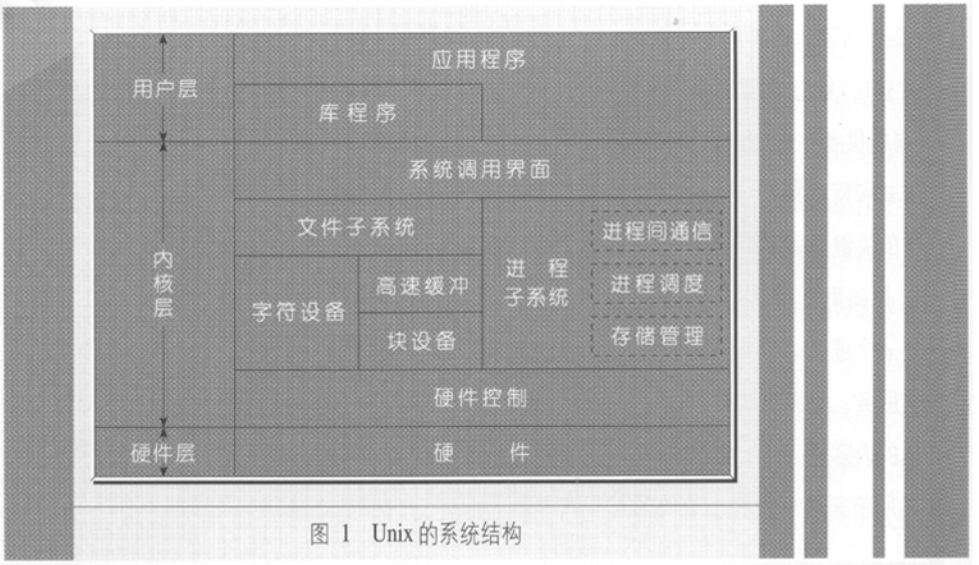

图1 Unix的系统结构

2 Unix/Linux操作系统的安全策略

一个安全的计算机系统必须有一个明确的，且定义良好的安全策略。系统中，只有授权用户或代表用户工作的进程才可以读、写、删除或建立相应的信息资源，也就是说，系统实现了完备的信息存取控制机制。对与系统安全有关的事件要进行审计、记录，并能找到当事人。安全机制必须是不可被篡改和非授权改变的等等。

2.1 标识

Unix/Linux的各种管理功能都被限制在一个帐号 (Root) 中，其功能和 Windows NT 的管理员 (Administrator) 或 Netware 的超级用户 (Supervisor) 功能类似。作为根用户，可以控制一切，包括：用户帐号，文件和目录，网络资源。根帐号允许你管理所有资源的各类变化情况，或者只管理很小范围的重大变化。例如：每个帐号都是具有不同用户名、不同的口令和不同的访问权限的一个单独实体。这样就允许你有权授予或拒绝任何用户、用户组合和所有用户的访问。用户可以生成自己的文件，安装自己的程序等。为了确保次序，系统会分配好用户目录，每个用户都得到一个主目录和一块硬盘空间。这块空间与系统区域和其他用户占用的区域分割开来。这种作用可以防止一般用户的活动影响其他文件系统。进而，系统还为每个用户提供一定程度的保密。作为根，可以控制哪些用户能够进行访问以及他们可以把文件存放在那里。控制用户能够访问哪些资源，用户如何进行访问等。

用户登录到系统中时，需输入用户名标识其身份。内部实现时，当该用户的账户创建时，系统管理员便为其分配一个唯一的标识号：UID。

系统中的/etc/passwd文件含有全部系统需要知道的关于每个用户的信息 (加密后的口令也可能存于 /etc/shadow 文件中)。/etc/passwd 中包含有用户的登录名，经过加密的口令、用户号、用户组号、用户注释、用户主目录和用户所用的 shell程序。其中用户号 (UID) 和用户 组号 (GID) 用于UNIX系统唯一地标识用户和同组用户及用户的访问权限。系统中，超级用户 (root) 的 UID 为 0。每个用户可以属于一个或多个用户组，每个组由 GID 唯一标识。

在大型的分布式系统中，为了统一对用户管理，通常将于每一台工作站上的口令文件信息存在网络服务器上。目前流行的系统有：Sun 公司的网络信息系统 (NIS)；Sun 公司的 NIS+；开放软件基金会的分布式计算机环境 (DCE) NetT计算机上的 NetInfo。

2.2 鉴别

用户名是个标识，它告诉计算机该用户是谁，而口令是个确认证据，用户登录系统时，需要输入口令来鉴别用户身份。当用户输入口令时，Unix 使用改进的 DES 算法 (通过调用 crypt ()函数实现) 对其加密，并将结果与存储在 /etc/passwd 或 NIS 数据库中的加密用户口令比较，若二者匹配，则说明该用户的登录合法，否则拒绝用户登录。

为防止口令被非授权用户盗用，对其设置应以复杂、不可猜测为标准。一个好的口令应当至少有6个字符长，不要取用个人信息，普通的英语单词也不好 (因为可用字典攻击法)，口令中最好有一些非字母 (如数字、标点符号、控制字符等)。用户应定期改变口令。通常，口令以加密的形式表示。由于 /etc/passwd 文件对任何用户可读，故常成为口令攻击的目标。所以系统中常用 shadow 文件 (/etc/shadow) 来存储加密口令，并使其对普通用户不可读。

2.3 存取控制

Unix 文件系统控制文件和目录中的信息以何种方式存在磁盘及其他辅助存储介质上。它控制每个用户可以访问何种信息及如何访问。它表现为一组存取控制规则，用来确定一个主体是否可以存取一个指定客体。Unix的存取控制机制通过文件系统实现。

2.3.1 存取权限

命令 ls 可列出文件 (或目录) 对系统内的不同用户所给予的存取权限。如：

-rw -r- -r--- 1 root root 1397 Mar 7 10:20 passwd

图 2 给出了文件存取权限的图形解释：

存取权限位共有 9bit 位，分为三组，用以指出不同类型的用户对该文件的访问权限。

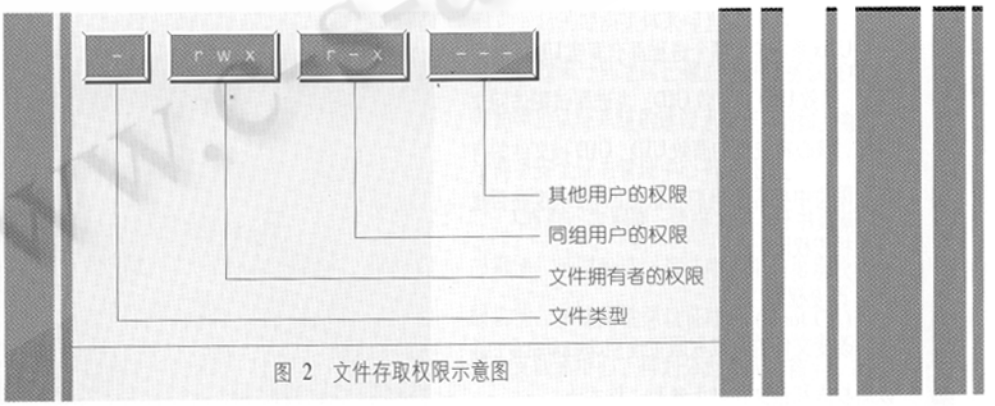

图 2 文件存取权限示意图

权限有三种：

r 允许读

w 允许写

x 允许执行

用户有三种类型：

owner 该文件的属主

group 在该文件所属用户组中的用户，即同组用户

other 除以上二者外的其他用户

如上图表示文件的属主具有读写及执行权限 (rwx)，同组用户允许读和执行操作，其他用户没有任何权限。权限位中，"–" 表示相应的存取权限不允许。

上述的授权模式同样适应于目录，用ls－l列出时，目录文件的类型为d。用ls列目录要有读许可，在目录中增删文件要有写许可，进入目录或将该目录作路径分量时要有执行许可，故要使用任一个文件，必须有该文件及找到该文件的路径上所有目录分量的相应许可。仅当要打开一个文件时，文件的许可才开始起作用，而rm、mv只要有目录的搜索和写许可，不需文件的许可，这一点应尤其注意。

一些版本的Unix系统支持访问控制表(ACL)，如AIX和HP-UX系统。它被用作标准的Unix文件存取权限的扩展。ACL提供更完善的文件授权设置，它可将对客体(文件、目录等)的存取控制细化到单个用户，而非笼统的"同组用户"或"其他用户"。使你可以为任意组合的用户以及用户组设置文件存取权限。

以HP-UX系统为例，用lsacl命令可以观察一个文件的ACL。如对于文件test:

(a.%, rw-) (%.b, r-x) (%.%,---) test

表示用户a(可以是任何组的成员)、用户组b及所有其他用户和用户组的权限。其中%为通配符。

Unix系统中，每个进程都有真实UID、真实GID、有效UID及有效GID。当进程试图访问文件时，核心将进程的有效UID、GID和文件的存取权限位中相应的用户和组相比较，决定是否赋予其相应权限。

2.3.2 改变权限

改变文件的存取权限可使用chmod命令，并以新权限和该文件名为参数。格式为：

chmod [-Rfh] 存取权限文件名

chmod也有其他方式的参数可直接对某组参数修改，在此不再多说，详见UNIX系统的联机手册。合理的文件授权可用于防止偶然性地重写或删除一个重要文件(即使是属主自己)。改变文件的属主和组名可用chown和chgrp，但修改后原属主和组员就无法修改回来了。

文件的授权可用一个4位的8进制数表示，后三位同图2所示的三组权限，许可位置1，不允许该权限则相应位置0。最高的一个8进制数分别对应SUID位、SGID位、sticky位。其中前两个与安全有关，我们将其做为特殊权限位在下节描述。

Umask(Unix对用户文件模式屏蔽字的缩写)也是一个4位的8进制数，Unix用它确定一个新建文件的授权。每一个进程都有一个从它的父进程中继承的umask。Umask说明你想对新建文件或新建目录的缺省授权加以屏蔽的部分。

新建文件的真正存取权限 = (~umask) & (文件授权)

Unix中相应有umask命令，若将此命令放入用户的profile文件，就可控制该用户后续所建文件的存取许可。umask命令与chmod命令的作用正好相反，它告诉系统在创建文件时不给予什么存取许可。

2.3.3 特殊权限位

有时，没有被授权的用户需要完成某些要求授权的任务。如passwd程序，对于普通用户，他允许改变自身的口令，但不能拥有直接访问/etc/passwd文件的权力，以防止改变其他用户的口令。为了解决这个问题，Unix允许对可执行的目标文件(只有可执行文件才有意义)设置SUID或SGID。

如前所述，当一个进程执行时就被赋于4个编号，以标识该进程隶属于谁，分别为实际和有效的UID、实际和有效的GID。有效的UID和GID一般和实际的UID和GID相同，有效的UID和GID用于系统确定该进程对于文件的存取许可。而设置可执行文件的SUID许可将改变上述情况，当设置了SUID时，进程的有效UID为该可执行文件的所有者的有效UID，而不是执行该程序的用户的有效UID，因此，由该程序创建的都有与该程序所有者相同的存取许可。这样，程序的所有者将可通过程序的控制在有限的范围内向用户发表不允许被公众访问的信息。同样，SGID是设置有效GID。用"chmod u+s 文件名"和"chmod u-s 文件名"来设置和取消SUID设置。用"chmod g+s 文件名"和"chmod g-s 文件名"来设置和取消SGID设置当文件设置了SUID和SGID后chown和chgrp命令将全部取消这些许可。