**第4章习题**

**1. chroot时有一些原则，**比如：

1）chroot之前关闭文件描述符；2）在chroot环境下以 non-root user 运行；3）正确的"放弃" 权限；4）利用chdir显式进入jail；5）尽可能让 root 管理jailed 文件

问题：理解chroot的应用原则，并说明不这么做会产生什么安全问题，如果你是攻击者会针对不同的情况如何进行攻击。对各个说明如何进行攻击或采取什么操作能避免上述问题。

1. chroot之前关闭文件描述符。

每个进程在进程表中都有一个记录项，每个记录项中包含一张打开文件描述符，包含：文件描述符标志、指向一个文件表项的指针。

dup函数，用来复制一个现存的文件描述符。

通过猜测文件描述符，即可使用已打开的文件，对已打开的文件进行读、写执行操作。

所以关闭文件描述符，能防止文件被非法操作。

（2）在chroot环境下以 non-root user 运行

因为仅root用户可运行 chroot。应防止用户将一个setuid程序(如, 一个假的/etc/passwd 文件)放入一个特殊编制的chroot环境以避免受骗而获得高权限

chroot并非在所有系统上都完全安全。因为是root权限下的chroot环境，一旦攻破chroot会影响root安全

（3）正确的"放弃" 权限。

程序利用"saved" uid在 non-root user 和 root 用户切换

利用无歧义函数setresuid() seteuid(), setreuid()。

1. 利用chdir显式进入jail

不进入jail文件夹，就没有进入虚拟环境。能通过虚拟环境进入外部的根目录。

1. 尽可能让 root 管理jailed 文件

root管理文件，在系统更新、升级时，虚拟环境能自动升级，否则要手动单独升级。

root用户应用了很多管理方法，应利用系统的管理方法。

**2. 对比chroot和root能力划分两种管理方式，分析其应用场景和安全性。哪种方式安全性更高？**

Chroot主要应用在启动时需要root权限，启动后不使用root或很少使用root的场景下。服务启动后，需要root权限的代码和主服务代码能够剥离开，适合使用chroot。

root能力位，应用于某能力位作用清晰，和其他服务的能力位非常明显的区分场景。比如kill能力，就是杀掉进程，root可以利用kill能力杀掉其他用户的进程。 root的进程管理、文件管理的能力位使用时还是需要注意安全性。

Chroot将进程限制在有限空间内，不影响其它进程。Chroot建立操作系统下的虚拟环境。在单核、单操作系统上运行多个虚拟服务。经过 chroot 之后，在新根下将访问不到旧系统的根目录结构和文件，这样就增强了系统的安全性。

Linux 内核将root权限分为多种能力。当把root的高权限授予某进程，普通用户运行时即可使用该高特权。对系统的影响也很大。

比如，给/bin/chown程序授予cap\_chown能力，那么普通用户可以用/bin/chown程序更改任意文件的owner。

因此，在使用root的能力位的时候，要仔细考量什么情况下可以授予什么能力位，尽量减少普通用户使用root能力位的时间。如果root的能力位应用不当，会对系统造成更大的伤害。

**3.** **root的可执行程序权限被setuid,和支持capability的系统中给可执行程序赋予某能力，这两种方式有何差别？**

**4. 在支持capability的linux系统中，CAP\_SYS\_MODULE、CAP\_LINUX\_IMMUTABLE能力的功能和使用方法。**

系统引导时删除部分能力，会保护系统。如，保护系统工具和日志的完整性。

CAP\_LINUX\_IMMUTABLE能力,允许修改文件的IMMUTABLE和APPEND属性标志。系统启动时没有CAP\_LINUX\_IMMUTABLE能力,攻击者不能删除其攻击轨迹、不能安装后门工具、系统日志文件为“append-only” 、系统工具不被删除和修改。

CAP\_SYS\_MODULE能力,允许修改系统内核。如果系统启动时没有CAP\_SYS\_MODULE能力，攻击者不能修改系统的内核。系统内核被改动，需要重新启动系统才能使用新内核。所以，对24小时在线运行的服务器来说，是不需要系统升级等修改内核操作的，要去除系统的CAP\_SYS\_MODULE能力。

**5. 在支持root能力集系统中，组合能力，在系统级设置几个从低到高的运行级别**

例如：

1. 设置网络服务能力，将CAP\_NET\_BIND\_SERVICE、CAP\_NET\_BROADCAST能力打开，即可获得网络服务能力。
2. 拥有root的文件管理能力。授予CAP\_DAC\_OVERRIDE、CAP\_DAC\_READ\_SEARCH能力即可

CAP\_DAC\_OVERRIDE 1 忽略对文件的所有DAC访问限制

CAP\_DAC\_READ\_SEARCH 2 忽略所有对读、搜索操作的限制

1. 拥有root的用户管理能力。授予CAP\_SETGID、CAP\_FSETID能力。

CAP\_FSETID 4 允许设置setuid位

CAP\_SETGID 6 允许改变组ID

**6. SELinux系统的授权方式**

1）理解SELinux系统的对进程细粒度的授权方式；

2）要求：系统中运行了httpd程序，允许www用户杀死httpd进程

* + 问题：如何进行细粒度的授权？并进行解释。

答案：

1. 主体对进程细粒度的授权方式，是给用户的进程进行授权，进程的权限是有限的，每个进程仅能执行有限的操作，不会越权执行权限。

粗粒度的授权方式，是将客体的权限授予用户，用户带着很多权限去执行进程。如果用户运行了root的setuid程序，可以利用root权限破坏系统文件或进程。而细粒度的访问控制因用户没有多余的权限，破坏系统的可能性大大降低。

2）允许www用户杀死httpd进程

允许root用户（root\_t）启动调用kill可执行文件（kill\_t）

allow root\_t kill\_t : httpd {getattr execute};

允许www用户（www\_t）启动execve()调用kill可执行文件（kill\_exec\_t）

allow www\_t kill\_exec\_t : file {getattr execute};

对kill\_t域的入口访问权 ：

allow kill\_t kill\_exec\_t : file entrypoint;

许可原始的类型（www\_t）到新的类型（kill\_t）进行域转变 TRansition 访问

allow www\_t kill\_t : process transition;