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()。

（4） 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。

（5） 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

（3） 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;
```