操作系统级防护方法

一.虚拟化限制方法

1.操作系统级虚拟化

- 运行在单核下, 单操作系统上运行多个虚拟服务

  –如, **chroot, FreeBSD jail**, ...

  –服务提供者可以利用较低的代价提供主机服务

  –**优点**: 性能高, 建立和管理比较容易

  –**缺点**: 所有服务在单操作系统上, 虚拟服务可能打破虚拟限制破坏操作系统

1)chroot系统调用

改变当前进程和子进程到指定路径下的“根”目录，新的“根”目录受真正的root的文件系统约束

a.作用

1.限制被CHROOT的使用者所能执行的程序，如SetUid的程序，或是会造成Load的 Compiler等等

2.防止使用者存取某些特定文件，如/etc/passwd

3.防止入侵者/bin/rm -rf /

4.提供Guest服务

5.增进系统的安全

b.优点

- 增加了系统的安全性，限制了用户的权力；

  • 在经过 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。

- 建立一个与原系统隔离的系统目录结构，方便用户的开发；

• 使用 chroot 后，系统读取的是新根下的目录和文件，这是一个与原系统根下文件不相关的目录结构。在这个新的环境中，可以用来测试软件的静态编译以及一些与系统不相关的独立开发程序

- 切换系统的根目录位置，引导 Linux 系统启动以及急救系统等。

  • chroot 的作用就是切换系统的根位置，而这个作用最为明显的是在系统初始引导磁盘的处理过程中使用，从初始 RAM 磁盘 (initrd) 切换系统的根位置并执行真正的 init。另外，当系统出现一些问题时，我们也可以使用 chroot 来切换到一个临时的系统

c.缺点

- 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
- chroot环境也会影响其他进程和用户空间
- chroot不能限制I/O、带宽, 磁盘空间、CPU时间等资源

d.建立chroot的原则

- 在chroot环境下以 non-root user 运行
- 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()
- 利用chdir显式进入jail（沙盒环境/监狱环境）
- 在jail环境中内容尽量少
- 尽可能让 root 管理jailed 文件
- 限制jail内文件和目录的权限
- 建立权限设置脚本
- 不要在jail环境存放 /etc/passwd 文件
- 在建立jail环境前关闭文件描述符
- chroot配置了网络服务后，外部需要查看chroot环境的配置文件，从外部连接配置文件

  > *–# ln -s /chroot/named/etc/named.conf/etc/named.conf*

如何理解chroot的安全原则

2)FreeBSD jail

a.特点

- Jail 提供分散化的管理(partitioning), 与虚拟机相似
- Jail内的限制因素

  –访问文件名空间时仅受限于jail内部

–可限制绑定在指定IP地址下的网络资源

–使用系统资源、执行特权操作能力大大减少

–仅能与jail内其它进程交互

- 如, ps 仅显示jail内进程名

b.目标

- 虚拟化: 每个 jail 是一个运行于单机的**虚拟环境**，拥有自己的**文件，进程，用户，超级用户**。在jail内部看起来和真实系统一样
- 安全: 每个jail和其他部分无关联，破坏其他部分比较困难
- 容易授权: 因为jail是受限环境, 管理员授权后对整个系统影响不大

c.缺点

- the FreeBSD jail 并不是真正的虚拟化，不同于虚拟机，不能运行不同的操作内核版本，利用原操作系统内核进行上级配置
- 所有虚拟服务共享同一内核，可利用操作系统的bugs 和安全漏洞进行攻击
- 没有聚簇和进程迁移能力, **主机内核和主机系统的单点失效**影响所有虚拟服务

d.安全性

- FreeBSD jails 可有效增加服务安全性，因为jail环境和其它其它部分是分离的
- 对FreeBSD jail的限制

  –Jail间进程不能互相干扰.

  > *如, **ps**只能察看jail内进程*

  –禁止访问和加载模块以防修改正在运行的内核. 禁止修改多数sysctrls配置和安全级别

  –禁止修改网络配置, 包括接口和IP地址、禁止访问路由表. jail仅能绑定指定IP地址，不可以修改防火墙规则

  –禁止Mounting、unmounting文件系统. 不能访问Jail外的目录 (a jail is chrooted).

  –禁止创建device nodes

e.优点

- 不同的jail可以按安装不同的daemon

  > *Daemon（守护进程）是运行在后台的一种特殊进程。它独立于控制终端并且周期性地执行某种任务或等待处理某些发生的事件。*
  >
  > *eg. 系统日志进程syslogd、 web服务器httpd、邮件服务器sendmail和数据库服务器mysqld等。*

- jail内管理员授权方便

  –jail内的超级用户具有有限特权 （参见安全性）

  –很难越过jail环境，很难获得jail外系统环境信息

2.虚拟机

- 在用户进程中模拟硬件

  –模拟软件运行在主机OS上，guest OS运行在模拟软件上

  –优点: 不修改OS，可直接运行多个guest OS

  –缺点: 性能差

3.虚拟指令集

- –无主机OS, 一个小型的虚拟指令集运行在硬件上, 修改guest OSes后在上面运行

  –能使不同和不兼容的OS运行在同一台计算机上

二.root权限划分

1.POSIX/Linux capablities

- Linux 内核将root权限分为多种能力

a.31种能力位

能力位	作用
CAP_CHOWN 0	允许改变文件的所有权
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
CAP_FOWNER 3	忽略文件属主ID必须和进程用户ID相匹配的限制
CAP_FSETID 4	允许设置setuid位
CAP_KILL 5	允许对不属于自己的进程发送信号
CAP_SETGID 6	CAP_SETGID 6
CAP_SETUID 7	允许改变用户ID
CAP_SETPCAP 8	允许向其它进程转移能力以及删除其它进程的任意能力
CAP_LINUX_IMMUTABLE 9	允许修改文件的不可修改(IMMUTABLE)和只添加(APPEND-ONLY)属性
CAP_NET_BIND_SERVICE 10	允许绑定到小于1024的端口
CAP_NET_BROADCAST 11	允许网络广播和多播访问
CAP_NET_ADMIN 12	允许执行网络管理任务
CAP_IPC_LOCK 14	允许锁定共享内存片段
CAP_IPC_OWNER 15	忽略IPC所有权检查
CAP_SYS_MODULE 16	插入和删除内核模块
CAP_SYS_RAWIO 17	允许对ioperm/iopl的访问

能力位	作用
CAP_SYS_CHROOT 18	允许使用chroot()系统调用
CAP_SYS_PTRACE 19	允许跟踪任何进程

b.能力集

- 每个进程有三个能力集
    - effective

      capabilities the process is using

      进程进行某个特权操作时,操作系统检查**cap_effective**的对应位是否有效,而不再是检查进程的有效**UID**是否为**0**
    - Permitted

      capabilities the process can use

      表示进程能够使用的能力,这些能力是被进程自己临时放弃的,**cap_effective**是**cap_permitted**的一个子集
    - inheritable

      capabilities that can be inherited when loading a new program

- 可执行文件能力集

  –Allowed(inheritable) can inherit these capabilities

  –forced(permitted) get these capabilities

  –Transfer(effective) these capabilities are transferred from permitted to effective,

c.能力边界集

- 能力边界集(capability bounding set)是系统中所有进程允许保留的能力
- 能力边界集：

  –通过sysctl命令导出，

  –在/proc/sys/kernel/cap-bound中

d.系统调用接口

- 系统调用capset and capget 设置和获得进程能力
- inheritable集合：当前进程调用exec(2)时可继承的能力集。用fork则子进程仅拷贝当前进程的三个能力集

e.能力的安全作用

- 程序可以使用root的部分特权，而不需要该程序setuid root

  –程序的forced能力位置1即可
- 系统引导时删除部分能力，会保护系统

  –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.*
>
> *去除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*

2.FreeBSD安全级

- -1: Permanently insecure mode.
  0: Insecure mode
  1: Secure mode
  2: Highly secure mode
  3: Network secure mode
- 在内核中执行安全级.

  –当安全级>0, 内核限制执行某些操作; superuser也不能执行限制操作

  不能降低正在运行系统的安全级

  –要降低安全级, 需要修改/etc/rc.conf中的安全级配置并需重启

  kern_securelevel_enable="YES" kern_securelevel="2"

三. 细粒度的强制访问控制

- 每个进程, 都有相应的策略控制该进程可以做什么

  –不同于自主访问控制，自主访问控制权限取决于user id

  –具体指明其能力，访问具体文件的权限
- 控制策略的执行时间

  –加载二进制代码时

  –进程获得数据时

1.细粒度访问控制工具

2.安全增强的Linux（SELinux）

a.模型

- 域-类型：模型意味着在安全域中运行着的每一个进程和每一个资源（一般文件、目录文件和套接字等）都有一个与之相联系的“类型”（type）。
- 在域-类型上建立了一系列规则，这些规则列出了某个域可以在每一个类型上执行的所有动作。
- SELinux的系统中，每一个进程的上下文都包含三个组成部分：一个ID（identity），一个角色（role）和一个域（domain ）

  –ID:用户,进程的所有者，系统进程ID(system_u)，未知用户进程ID(user_u)

–角色用来判断某个处于此角色的ID可以进入哪些域，还用来防止某个处于此角色的ID进入其它不该进入的域。比如，user_r角色就不允许进入 sysadm_t

一个安全上下文可以像 identity:role:domain。可用ps和ls查看

b.类型强制访问控制-TE(Type enforce)

- 所有访问都必须明确授权，SELinux默认不允许任何访问，不管Linux用户/组ID是什么。即在SELinux中，没有默认的超级用户
- 与标准Linux中的root不一样，通过指定主体类型（即域）和客体类型使用**allow规则**授予访问权限
    - allow规则由四部分组成：

      –源类型（Source type(s) ） 尝试访问的进程的域类型

      –目标类型（Target type(s) ） 被进程访问的客体的类型

      –客体类别（Object class(es)） 指定允许访问的客体的类型

      –许可（Permission(s)） 象征目标类型允许源类型访问客体类型的访问种类

> *普通用户修改密码过程:*
> ***1域转变***
> *–Linux给passwd设置set_uid_root*
> *# ls -l /usr/bin/passwd*
> *-r-s--x--x 1 root root 19336 Sep 7 04:11 /usr/bin/passwd*
>
> ***2域转变***
> *Joe的shell域（user_t），*
> *密码程序的域类型（passwd_t）和shadow密码文件的类型（shadow_t）。*
> *passwd可执行文件增加了文件类型（passwd_exec_t）。*
>
> ***3创建TE策略规则允许密码程序passwd_t域类型运行***
> *允许Joe的shell（user_t）启动execve()调用passwd可执行文件（passwd_exec_t）*
> ***allow user_t passwd_exec_t : file {getattr execute};***
> *对passwd_t域的入口访问权*
> ***allow passwd_t passwd_exec_t : file entrypoint;***
> *许可原始的类型（user_t）到新的类型（passwd_t）进行域转变*
> *TRansition 访问*
> ***allow user_t passwd_t : process transition;***