系统安全

系统权限

一、安装系统

1. 选择稳定版操作系统
2. 最小化安装
3. 不要安装gcc,make
4. 安装完系统后更新系统

```
[root@localhost ~]# yum -y update
```

二、文件(目录)权限

1. 基本权限 rwx
   - 对于目录，默认权限=777-umask
   - 对于文件，默认权限=666-umask(文件默认无执行权限)
   - 修改umask

```
[root@localhost ~]# vim /etc/bashrc
 71         umask 002 #普通用户
 72     else
 73         umask 022 #超级用户
 74     fi
 [root@localhost ~]# vim /etc/profile
 60     umask 002 #普通用户
 61 else
 62     umask 022 #超级用户

```

2. 特殊权限 suid sgid sticky
   - suid 冒险位，执行二进制文件与文件所有人有关，与谁来执行无关

```
[root@localhost ~]# chmod 4777 filename
```

   - sgid 强制位，对目录生效，在此目录中创建文件自动归入目录所在组

```
[root@localhost ~]# chmod 2777 dirname
```

- sticky 粘制位，目录中的文件只能被文件拥有者删除

```
[root@localhost ~]# chmod 1777 dirname
```

3. 文件ACL getfacl setfacl
   - 对文件的权限进行附加说明的权限设定方式
   - ACL提供传统的owner/group/other的read/write/execute之外的细部权限设定。（可以针对单一的使用者，目录等等）
   - 查看

```
[root@localhost ~]# ls -l
总用量 1
-rw-r--r--  1 root root    4 2月  17 20:56 test.txt
#-rw-r--r--+ 如果权限后面带有‘+’号表示有ACL权限

[root@localhost ~]# getfacl test.txt
# file: test.txt 文件名
# owner: root 文件所属者
# group: root 文件所属组
user::rw- 属主权限
group::r-- 属组权限
other::r-- 其他人权限
```

   - 设定ACL权限

```
[root@localhost ~]# useradd jack
[root@localhost ~]# setfacl -m u:jack:rw test.txt

#setfacl -m <u|g|m>:<username|groupname>:权限  filename
#setfacl -x <u|g>:<username|groupname>  filename    ##去除某个用户或者组的acl
#setfacl -b                  filename    ##删除文件上的权限列表

[root@localhost ~]# getfacl test.txt
# file: test.txt
# owner: root
# group: root
user::rw-
user:jack:rw- 为jack设置rw权限
group::r--
mask::rw-
other::r--
[root@localhost ~]# ls -l test.txt
-rw-rw-r--+ 1 root root 4 2月  17 20:56 test.txt
```

4. 文件属性 chattr lsattr +a -a +i -i -d

```
[root@localhost ~]# chattr +a test.txt
#只能给文件添加内容，但是删除不了，属于追加
[root@localhost ~]# chattr -d test.txt

[root@localhost ~]# chattr +i test.txt
#文件不能删除，不能更改，不能移动

[root@localhost ~]# lsattr test.txt
----i----------- test.txt

```

案例1：防删除，防修改

```
[root@localhost ~]# find /bin /sbin /usr/sbin /usr/bin /etc/shadow /etc/passwd
/etc/pam.d -type f -exec chattr +i {} \;
```

案例2：日志文件防删除

```
[root@localhost ~]# chattr +a /var/log/messages /var/log/secure
#日志切割要先去掉a属性，之后增加a属性

[root@localhost ~]# vim /etc/logrotate.d/syslog
prerotate
    chattr -a /var/log/messages
endscript

...

postrotate
    chattr +a /var/log/messages
endscript
}
```

5. mask umask权限

```
[root@localhost ~]# umask
0022
[root@localhost ~]# umask -S
u=rwx,g=rx,o=rx
```

6. mount 权限 -o

rw ro

sync async

> 此选项的默认模式为异步模式。在同步模式下，内存的任何修改都会实时的同步到硬盘当中，这种模式的安全性基本属于最高，但是因为内存的数据基本一直都在变化，所以这种模式会使得程序运行变得缓慢，影响效率。而在异步模式下，虽然同步没有实时，但是现在考虑到日志文件系统的存在，所以安全性基本不用考虑，而异步模式的效率会更高，随意目前普遍使用异步模式为默认

auto noauto

> 合理规划权限，尽量避免777权限出现

用户授权

su

- 由超级用户切换为普通用户，仅切换用户，环境变量不切换，如若为普通用户，会导致命令不可用

```
[root@localhost ~]# su jack
[jack@localhost root]$
```

- 由超级用户切换为普通用户，切换用户至家目录，环境变量会发生改变

```
[root@localhost ~]# su - jack
上一次登录：日 2月 17 21:48:05 CST 2019pts/1 上
[jack@localhost ~]$
```

- 由普通用户切换为root用户

```
[jack@localhost ~]$ su - root
密码：  #需要输入密码，可获取root全部权限，此处密码认证由PAM提供
上一次登录：日 2月 17 20:45:56 CST 2019从 192.168.2.1pts/1 上
[root@localhost ~]#
```

sudo

给普通用户提升(赋予)权限的方法

- suid,sgid
- usermod
- switching users with su
- running commands as root with sudo

使用sudo提升(赋予)权限普通用户的权限

可根据/etc/sudoers文件设置普通用户使用sudo命令时可以以root身份或其他用户身份运行命令

- 使用vim直接编辑/etc/sudoers文件

```
[root@localhost ~]# vim /etc/sudoers
#不推荐
```

- 使用visudo编辑/etc/sudoers

```
[root@localhost ~]# visudo
#推荐，会检查语法
```

- sudo语法

```
#user        MACHINE=(RUN_AS_USER)            COMMANDS
jack         ALL=ALL                          ALL
#允许jack用户  在任何主机上=（以任何人的身份）      执行任何命令
```

- 案例

案例1：对用户

```
[root@localhost ~]# grep jack /etc/passwd
jack:x:1001:1001::/home/jack:/bin/bash
[root@localhost ~]# grep owen /etc/passwd
owen:x:1002:1002::/home/owen:/bin/bash

[root@localhost ~]# visudo
jack    ALL=/sbin/ip, /sbin/fdisk, /bin/less
#赋予jack用户使用以上3个命令的权限
[root@localhost ~]# visudo
owen    ALL=NOPASSWD: /bin/less
##赋予owen用户使用以上1个命令的权限,切换时不需要输入密码
```

案例2：对组

```
[root@localhost ~]# groupadd smartgo
[root@localhost ~]# useradd it01 -G smartgo
[root@localhost ~]# useradd it02 -G smartgo
[root@localhost ~]# id it01
uid=1003(it01) gid=1004(it01) 组=1004(it01),1003(smartgo)
[root@localhost ~]# id it02
uid=1004(it02) gid=1005(it02) 组=1005(it02),1003(smartgo)

[root@localhost ~]# visudo
%smartgo        ALL=NOPASSWD: /sbin/ip
%smartgo        ALL=NOPASSWD: /sbin/useradd, /sbin/userdel, /bin/passwd
%smartgo        ALL=NOPASSWD: !/bin/passwd root, !/bin/passwd root --stdin,
!/bin/passwd --stdin root

[root@localhost ~]# su - it01
[it01@localhost ~]$ sudo passwd root
对不起，用户 it01 无权以 root 的身份在 localhost.localdomain 上执行 /bin/passwd root。
```

案例3：别名使用

```
[root@localhost ~]# visudo
## Host Aliases
# Host_Alias     FILESERVERS = fs1, fs2
Host_Alias         MAILSERVERS = smtp, smtp2

## User Aliases
User_Alias ADMINS = jsmith, mikem

## Command Aliases
## These are groups of related commands...
## Networking
Cmnd_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhclient,
/usr/bin/net,
/sbin/iptables, /usr/bin/rfcomm, /usr/bin/wvdial, /sbin/iwconfig, /sbin/mii-tool

## Installation and management of software
Cmnd_Alias SOFTWARE = /bin/rpm, /usr/bin/up2date, /usr/bin/yum

## Services
Cmnd_Alias SERVICES = /sbin/service, /sbin/chkconfig

## Updating the locate database
Cmnd_Alias LOCATE = /usr/bin/updatedb

```

```
## Storage
Cmnd_Alias STORAGE = /sbin/fdisk, /sbin/sfdisk, /sbin/parted, /sbin/partprobe,
/bin/mount, /bin/umount


jack            ALL=NOPASSWD: NETWORKING
%smartgo        ALL=NOPASSWD: STORAGE
ADMINS          ALL=NOPASSWD: NETWORKING, STORAGE
```

- sudo日志

```
[root@localhost ~]# grep '^authpriv' /etc/rsyslog.conf
authpriv.*                                              /var/log/secure
[root@localhost ~]# tail -f /var/log/secure
Feb 17 22:31:52 localhost passwd: pam_unix(passwd:chauthtok): password changed for
root
Feb 17 22:31:52 localhost passwd: gkr-pam: couldn't update the login keyring password:
no old password was entered
Feb 17 22:32:15 localhost sudo:    it01 : command not allowed ; TTY=pts/1 ;
PWD=/home/it01 ; USER=root ; COMMAND=/bin/passwd root --stdin
Feb 17 22:32:28 localhost su: pam_unix(su-l:session): session closed for user it01
Feb 17 22:33:10 localhost su: pam_unix(su-l:session): session opened for user it01 by
root(uid=0)
Feb 17 22:33:17 localhost sudo:    it01 : command not allowed ; TTY=pts/1 ;
PWD=/home/it01 ; USER=root ; COMMAND=/bin/passwd root
Feb 17 22:33:30 localhost sudo:    it01 : TTY=pts/1 ; PWD=/home/it01 ; USER=root ;
COMMAND=/bin/passwd owen
Feb 17 22:33:36 localhost passwd: pam_unix(passwd:chauthtok): password changed for
owen
Feb 17 22:33:36 localhost passwd: gkr-pam: couldn't update the login keyring password:
no old password was entered
Feb 17 22:33:39 localhost su: pam_unix(su-l:session): session closed for user it01
```

用户认证

用户认证方式

- PAM(gdm,kdm,su,ssh,ftp,samba)
- 自带数据库验证方式(MySQL,Zabbix)
- web验证方式(htpasswd)
- 集中式身份认证

PAM介绍

PAM(Pluggable Authentication Modules) 即可插拔式认证模块，它是一种高效而且灵活的用户级别的认证方式，它也是当前Linux服务器普遍使用的认证方式。

PAM可以根据用户的网段、时间、用户名、密码等实现认证。

PAM身份认证

使用PAM做身份认证的服务有：本地（login、gdm、kdm），sshd， vsftpd，samba等

不使用PAM做身份认证的服务有：MySQL-Server，Zabbix等

- PAM使用帮助

```
[root@localhost ~]# firefox /usr/share/doc/pam-1.1.8/html/Linux-PAM_SAG.html
```

- PAM认证原理

```
Service(进程文件) →  PAM(配置文件) →  pam_*.so  →  模块的配置文件
/usr/sbin/sshd    /etc/pam.d/sshd  /lib64/security/pam_access.so
/etc/security/access.conf
                                   /lib64/security/pam_limits.so
/etc/security/limits.conf
                                   /lib64/security/pam_time.so
/etc/security/time.conf
/bin/su           /etc/pam.d/su    /lib64/security/pam_rootok.so
```

- PAM认证原理案例

```
[root@localhost ~]# ldd /usr/sbin/sshd | grep -i pam
        libpam.so.0 => /lib64/libpam.so.0 (0x00007f65d9d8e000)

[root@localhost ~]# grep -i pam /etc/ssh/sshd_config
# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via ChallengeResponseAuthentication may bypass
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# WARNING: 'UsePAM no' is not supported in Red Hat Enterprise Linux and may cause
several
UsePAM yes


[root@localhost ~]# vim /etc/pam.d/sshd
#%PAM-1.0
auth       required     pam_sepermit.so
auth       substack     password-auth
```

```
auth       include      postlogin
# Used with polkit to reauthorize users in remote sessions
#与polkit一起使用以重新授权远程会话中的用户
-auth      optional     pam_reauthorize.so prepare
account    required     pam_nologin.so
account    include      password-auth
password   include      password-auth
# pam_selinux.so close should be the first session rule
#selinux关闭执行如下
session    required     pam_selinux.so close
session    required     pam_loginuid.so
# pam_selinux.so open should only be followed by sessions to be executed in the user context
#selinux开启执行如下
session    required     pam_selinux.so open env_params
session    required     pam_namespace.so
session    optional     pam_keyinit.so force revoke
session    include      password-auth
session    include      postlogin
# Used with polkit to reauthorize users in remote sessions
-session   optional     pam_reauthorize.so prepare

```

- PAM常见的四种认证类型

```
auth        认证管理          验证使用者身份，账号和密码
account     用户管理          基于用户时间或密码有效期来决定是否允许访问
password    密码（口令）       认证管理 禁止用户反复尝试登录，在变更密码时进行密码复杂性控制
session     会话管理          进行日志记录，或者限制用户登录的次数，资源限制
```

- PAM认证流程控制(流程标记)

```
Required    (必要条件)  验证失败时仍然继续，但返回fail              用户不会知道哪里失败
Requisite   (必要条件)  验证失败时则立即结束整个验证过程，返回fail  面试若不成功，马上失败，效率高
Sufficient  (充分条件)  验证成功则立即返回，不再继续，否则忽略结果并继续      相当于面试中的拔高题
Optional    (可选条件)  无论验证结果如何，均不会影响                    常用于session类型
Include                 包含另外一个配置文件中类型相同的行
substack                垂直叠加
```

- PAM常用模块

```
模块：pam_rootok.so
功能：用户UID是0，返回成功

示例：限制root切换用户也需要密码
[root@localhost ~]# head -1 /etc/pam.d/su
#auth sufficient pam_rootok.so

示例：sshd不需要密码登录
[root@localhost ~]# head -1 /etc/pam.d/sshd
auth sufficient pam_rootok.so
#放在文件的第一行
```

```
模块：pam_access.so
功能：访问控制，默认配置文件/etc/security/access.conf
通常作用于登录程序，如su,login,gdm,sshd，
例如：限制用户从哪些网段登录sshd

示例：不允许root从192.168.1.0/24登录sshd
[root@localhost ~]# grep access.so /etc/pam.d/sshd
auth    required    pam_access.so
[root@localhost ~]# vim /etc/security/access.conf
-:root:192.168.122.0/24
-:root:ALL EXCEPT 192.168.1.0/24
-:root:192.168.122.0/24 EXCEPT 192.168.122.1


示例：使用不同的模块配置文件
[root@localhost ~]# grep access /etc/pam.d/login
auth    required    pam_access.so accessfile=/accessfile2
[root@localhost ~]# grep jack /accessfile2
-:jack:tty5 tty6
================================================================================
[root@localhost ~]# grep access.so /etc/pam.d/sshd
auth    requisite    pam_access.so accessfile=/accessfile1
[root@localhost ~]# grep 110 /accessfile1
-:root:ALL EXCEPT 192.168.2.110
```

```
模块：pam_listfile.so
功能：基于自定义文件允许或拒绝（黑名单或白名单）

示例：vsftpd黑名单或白名单
[root@localhost ~]# grep listfile /etc/pam.d/vsftpd
auth       required     pam_listfile.so item=user sense=deny file=/etc/vsftpd/ftpusers onerr=succeed

示例：sshd黑名单或白名单
[root@localhost ~]# grep listfile /etc/pam.d/sshd
auth       required     pam_listfile.so item=user sense=allow file=/etc/ssh_users onerr=fail
[root@localhost ~]# echo root > /etc/ssh_users

当/etc/ssh_users不存在时，fail
```

```
模块：pam_time.so
功能：基于时间的访问控制，默认文件/etc/security/time.conf

示例：基于时间限制sshd的访问
[root@localhost ~]# grep time /etc/pam.d/sshd
account    required     pam_time.so
[root@localhost ~]# grep 0800 /etc/security/time.conf
sshd;*;*;MoTuWeThFr0800-1100
```

```
模块：pam_tally2.so
功能：登录统计

示例：实现防止对sshd暴力破解
[root@localhost ~]# grep tally2 /etc/pam.d/sshd
auth       required     pam_tally2.so deny=2 even_deny_root root_unlock_time=60 unlock_time=60

#deny=2 连续错误登录最大次数，超过最大次数，将被锁定
#even_deny_root root用户也被要求锁定
#root_unlock_time root用户被锁定后等待的时间，单为秒
#unlock_time 普通用户被锁定后等待的时间，单为秒


[root@localhost ~]# pam_tally2  -u root
#查看用户错误登录次数

[root@localhost ~]# pam_tally2 --reset -u root
#清除用户错误登录次数


```

PAM资源限制

PAM资源限制主要是对用户进行系统资源使用的限制

PAM资源限制默认已使用，我们只需要调整相应限制值即可。

```
模块：pam_limits.so
功能：限制用户会话过程中对各种资源的使用情况。缺省情况下该模块的配置文件是
/etc/security/limits.conf
/etc/security/limits.d/*.conf
```

PAM资源限制案例

案例1：设置用户最大打开文件数

```
[root@localhost ~]# ulimit -a

[root@localhost ~]# ulimit -n
1024

[jack@localhost ~]$ ulimit -n
1024

[root@localhost ~]# vim /etc/security/limits.conf
*                soft    nofile          10240
*                hard    nofile          20480
```

案例2：设置用户最大创建的进程数

```
[jack@localhost  ~]$ ulimit -u
1024

[root@localhost  ~]# vim /etc/security/limits.d/90-nproc.conf
*          soft    nproc     10240
*          hard    nproc     10240
```

案例3：设置用户jack最大使用CPU的时间

```
[root@localhost ~]# vim /etc/security/limits.conf
jack           hard    cpu             1
```

> PAM资源限制针对用户，不针对进程，如果需要实现进程资源限制，可以考虑使用Cgroup。

扩展：Control Group(CGroup)资源限制组

控制组（CGroups）是Linux内核的一个特性，主要用来对共享资源进行隔离、限制、审计等。只有能控制分配到容器的资源，才能避免多个容器同时运行时对宿主机系统的资源竞争。控制组可以提供对容器的内存、CPU、磁盘IO等资源进行限制和计费管理。控制组的设计目标是为不同的应用情况提供统一的接口，从控制单一进程（比如nice工具）到系统级虚拟化（包括OpenVZ、Linux-VServer、LXC等）。

具体来看，控制组提供：

资源限制（Resource limiting）：可以将组设置为不超过设定的内存限制。比如：内存子系统可以为进程组设定一个内存使用上限，一旦进程组使用的内存达到限额再申请内存，就会出发Out of Memory警告。 优先级（Prioritization）：通过优先级让一些组优先得到更多的CPU等资源。 资源审计（Accounting）：用来统计系统实际上把多少资源用到适合的目的上，可以使用cpuacct子系统记录某个进程组使用的CPU时间。 隔离（isolation）：为组隔离命名空间，这样一个组不会看到另一个组的进程、网络连接和文件系统。 控制（Control）：挂起、恢复和重启动等操作。

cgroups: Control Groups 基于进程的限制，而非用户，因此对于超户运行的进程也是一样

cgroup将各种子系统定义为资源，命名为controller: 可配额/可度量 - Control Groups (cgroups)

cgroups实现了对资源的配额和度量九大子系统的资源

1. blkio 限制每个块设备的输入输出控制。例如:磁盘,光盘以及usb
2. cpu 限制使用cpu比例
3. cpuacct 产生cgroup任务的cpu资源报告。
4. cpuset 多核心的cpu时为cgroup任务分配单独的cpu和内存
5. devices 允许或拒绝对设备的访问。
6. freezer 暂停和恢复cgroup任务。
7. memory 设置内存限制以及产生内存资源报告。
8. net_cls 标记每个网络包。
9. ns 名称空间子系统

例如:对某个进程使用内存进行限制步骤：

1. 需要在controller memory下建立cgroup，如nginx_mem控制组，并针对该控制组nginx_mem设置相应的内存限制参数
2. 将进程Nginx分配到 memory controller的控制组(nginx_mem)，没有使用controller则不会限制。

Cgroup实现资源限制的方法：

a. cgexec 手动分配 b. cgred 自动分配

Cgroup部署方法：

```
[root@localhost ~]#yum -y install libcgroup*
[root@localhost ~]#systemctl enable cgconfig
[root@localhost ~]#systemctl start cgconfig
[root@localhost ~]#man cgconfig.conf
```

Cgroup限制步骤：

1. 创建cgroup，定义相应的限制
2. 分配程序到cgroup

案例1：限制进程使用CPU

1. 使用cpu子系统创建两个cgroup

```
[root@localhost ~]#vim /etc/cgconfig.conf
```

```
group lesscpu {
    cpu {
        cpu.shares=200;
    }
}
group morecpu {
    cpu {
        cpu.shares=800;
    }
}
```

```
[root@localhost ~]#systemctl restart cgconfig
```

2.将程序分配到相应的group 实验中，为了让两个进程抢CPU时间片，故意只留一个CPU在线

```
[root@localhost ~]#lscpu
[root@localhost ~]#echo 0 > /sys/devices/system/cpu/cpu0/online
[root@localhost ~]#echo 1 > /sys/devices/system/cpu/cpu1/online
```

手动分配：

```
[root@localhost ~]#cgexec -g cpu:lesscpu sha1sum /dev/zero
[root@localhost ~]#cgexec -g cpu:morecpu md5sum /dev/zero
[root@localhost ~]#top
```

> 以上三条命令请在三个shell终端中打开，观察各进程所占用CPU情况。

案例2：限制进程使用Memory

- 添加cgroup

```
[root@localhost ~]#vim /etc/cgconfig.conf
```

```
group lessmem {
        memory {
            memory.limit_in_bytes=268435465; //物理内存限制256M
        }
}
```

```
[root@localhost ~]#systemctl restart cgconfig
```

- 创建内存盘

```
[root@localhost ~]# mkdir /mnt/mem_test
[root@localhost ~]#mount -t tmpfs /dev/shm /mnt/mem_test
[root@localhost ~]# cgexec -g memory:lessmem dd if=/dev/zero of=/mnt/mem_test/file bs=1M count=200   //OK
[root@localhost ~]# cgexec -g memory:lessmem dd if=/dev/zero of=/mnt/mem_test/file bs=1M count=500   //OK
[root@localhost ~]#free -m
```

> 结果为失败

- 创建cgroup

```
[root@localhost ~]#vim /etc/cgconfig.conf
```

```
group poormem{
        memory{
            memory.limit_in_bytes=268435465;      //物理内存限制256M
            memory.memsw.limit_in_bytes=268435465;  //总内存限制，物理+SWAP
        }
    }
```

```
[root@localhost ~]#systemctl restart cgconfig
```

- 创建内存盘并测试

```
[root@localhost ~]# mkdir /mnt/mem_test
[root@localhost ~]# mount -t tmpfs /dev/shm /mnt/mem_test
[root@localhost ~]# cgexec -g memory:lessmem dd if=/dev/zero of=/mnt/mem_test/file bs=1M count=200   //OK
[root@localhost ~]# cgexec -g memory:lessmem dd if=/dev/zero of=/mnt/mem_test/file bs=1M count=500   //Fail
[root@localhost ~]#free -m
```

扩展：LDAP集中式身份认证