实验一

要求：

（1） 设想一种场景需要进行普通用户和 root 用户切换，设计程序实现 euid 的安全管理

配合第 3 章 完成进程中 euid 的切换，实现 root 权限临时性和永久性管理，加强程序的安全性

说明：1 学时，不分组实现

（2） 搭建安全的沙盒环境，在沙盒环境中提供必须的常见工具，并提供程序验证沙盒环境的安全性

配合第 3 章 实现系统中的虚拟化限制方法，实现安全的系统加固，测试虚拟化空间的加固程度

说明：3 学时，2 人一组，分组实现

1.1 Linux 系统文件和目录权限设置与辨识 setuid 程序 uid 差异

1、 设计并实现不同用户对不同类文件的 r、w、x 权限:

（1）查看系统文件的权限设置

a)查看/etc/passwd 文件和/etc/bin/passwd 文件的权限设置，并分析其权限为什么这么设置；

```
hiter-lxj1170301015@hiter-lxj1170301015 ~ ls -l /etc/passwd
-rw-r--r-- 1 root root 2577 12月  3 01:35 /etc/passwd
```

/etc/passwd 权限

```
✘ hiter-lxj1170301015@hiter-lxj1170301015 ~ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 59640 3月  23  2019 /usr/bin/passwd
```

/usr/bin/passwd 权限

分析：/etc/passwd 仅对 root（该文件拥有者）开放写权限，普通用户仅设置读权限，这样做可以保证系统的安全。而/usr/bin/passwd 文件的所有者也是 root，但该文件设置了 setuid 位，这样，当该文件由另一个进程（普通用户）执行时，该进程可以拥有 root 的权限，普通用户使用 passwd 命令更改登录口令时，shell 会调用/usr/bin/passwd，此时 shell 具有 root 权限，所以可以修改 /etc/passwd 文件来更改用户登录口令。注意，此时用户修

改的口令保存在/etc/shadow 里面。

b)找到 2 个设置了 setuid 位的可执行程序，该程序的功能，该程序如果不设置 setuid 位是否能够达到相应的功能。

如下图所示，sudo & ping 设置了 setuid 位。

Sudo 功能：控制用户对系统命令的使用权限,root 允许的操作。通过 sudo 可以提高普通用户的操作权限，不过这个权限是需要进行配置才可使用。

Ping 功能：用来查看网络上另一个主机系统的网络连接是否正常的一个工具。ping 命令的工作原理是：向网络上的另一个主机系统发送 ICMP 报文，如果指定系统得到了报文，它将把回复报文传回给发送者。

```
 hiter-lxj1170301015@hiter-lxj1170301015 ~ ls -l /usr/bin/sudo
-rwsr-xr-x 1 root root 149080 10月  11 02:32 /usr/bin/sudo
 hiter-lxj1170301015@hiter-lxj1170301015 ~ ls -l /bin/ping
-rwsr-xr-x 1 root root 64424 6月   28 19:05 /bin/ping
```

设置了 setuid 位的可执行程序

Ping 不设置 S 位时：

```
 hiter-lxj1170301015@hiter-lxj1170301015 ~ ping www.baidu.com
PING www.a.shifen.com (182.61.200.6) 56(84) bytes of data.
64 bytes from 182.61.200.6 (182.61.200.6): icmp_seq=1 ttl=128 time=23.6 ms
64 bytes from 182.61.200.6 (182.61.200.6): icmp_seq=2 ttl=128 time=27.1 ms
^C
--- www.a.shifen.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 23.688/25.395/27.102/1.707 ms
 hiter-lxj1170301015@hiter-lxj1170301015 ~ sudo chmod u-s /bin/ping
 hiter-lxj1170301015@hiter-lxj1170301015 ~ ping www.baidu.com
ping: socket: 不允许的操作
 ✘ hiter-lxj1170301015@hiter-lxj1170301015 ~
```

恢复设置：

```
 ✘ hiter-lxj1170301015@hiter-lxj1170301015 ~ sudo setcap cap_net_raw+ep /bin
/ping
 hiter-lxj1170301015@hiter-lxj1170301015 ~ ping www.bai.com
PING bai-s2.nodes.gz.com (14.17.96.13) 56(84) bytes of data.
64 bytes from 14.17.96.13 (14.17.96.13): icmp_seq=1 ttl=128 time=60.6 ms
64 bytes from 14.17.96.13 (14.17.96.13): icmp_seq=2 ttl=128 time=76.8 ms
^C
--- bai-s2.nodes.gz.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 60.672/68.753/76.835/8.085 ms
```

Sudo 不设 S 位时：

```
 hiter-lxj1170301015@hiter-lxj1170301015 ~ sudo chmod u-s /usr/bin/sudo
 hiter-lxj1170301015@hiter-lxj1170301015 ~ sudp
zsh: command not found: sudp
 ✘ hiter-lxj1170301015@hiter-lxj1170301015 ~ sudo
sudo: /usr/bin/sudo 必须属于用户 ID 0(的用户)并且设置 setuid 位
```

恢复设置时：

```
 hiter-lxj1170301015@hiter-lxj1170301015 ~ sudo
usage: sudo -h | -K | -k | -V
usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user]
            [command]
usage: sudo [-AbEHknPS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
            prompt] [-T timeout] [-u user] [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-AknS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
            prompt] [-T timeout] [-u user] file ...
 ✘ hiter-lxj1170301015@hiter-lxj1170301015 ~
```

附：恢复 sudo 的方法

进入 recovery 模式--->重启-->重启开始时按 esc 进入选择模式--->选择 recovery 模式---->选择 root，进入后执行命令行：

chown root:root /usr/bin/sudo

chmod 4755 /usr/bin/sudo

reboot

分析：不设置 setuid 普通用户是无法使用上述命令的，因为命令代码中有创建原始套接字的部分，需要调用 setuid 把进程的有效用户 ID 设置成实际用户 ID，运行程序的进程必须拥有超级用户特权才能创建原始套接字。故如果没有 setuid，以普通用户是无法使用命令的。

（2）设置文件或目录权限

a)用户 A 具有文本文件"流星雨.txt"，该用户允许别人下载；

创建新用户 user1

```
 hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 sudo useradd user
1
[sudo] hiter-lxj1170301015 的密码：
 hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 sudo passwd user1

输入新的 UNIX 密码：
重新输入新的 UNIX 密码：
passwd: 已成功更新密码
```

在桌面/ex/lab1 文件夹中创建 流星雨.txt 修改其权限为 644

```
 hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 touch 流星雨.txt
 hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 ls -l
总用量 4
-rw-r--r-- 1 hiter-lxj1170301015 hiter-lxj1170301015 26 12月  3 02:24 流星雨.txt
 hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 chmod 644 流星雨.
txt
 hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 ls -l
总用量 4
-rw-r--r-- 1 hiter-lxj1170301015 hiter-lxj1170301015 26 12月  3 02:24 流星雨.txt
```

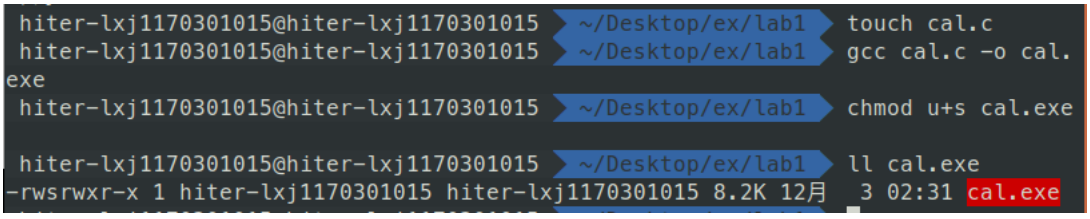
切换至用户 user1，此时该用户可以查看其内容

```
 hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 su user1
密码：
$ ls
流星雨.txt
$ cat 流星雨.txt
THis is a liuxingyu.txt .
```

切换回后，修改其权限为 740，再次用 user1 查看，此时没有权限

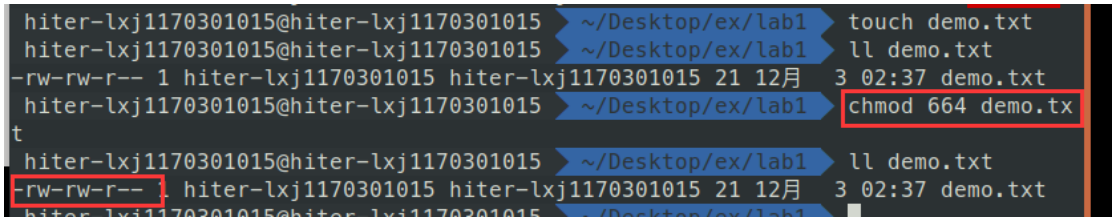
b)用户 A 编译了一个可执行文件”cal.exe”，该用户想在系统启动时运行；

在桌面/ex/lab1 文件夹创建 cal.c 并进行编译，如需在系统启动时运行，设置 cal.exe 的 setuid 位，开机启动需要将可执行文件 “cal.exe” 放到/etc/init.d 中，在/etc/rc.d/rc3.d 中建立软链接。



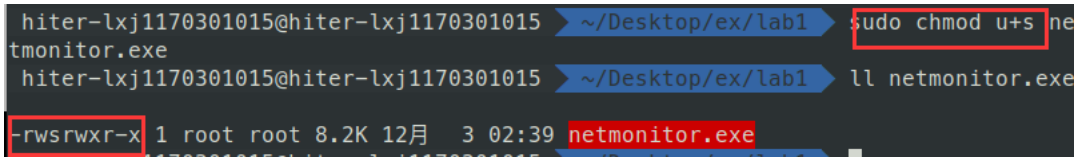
c)用户 A 有起草了文件”demo.txt”，想让同组的用户帮其修改文件；

在桌面/ex 文件夹创建 demo.txt，设置同组用户的写权限，即修改权限位为 664



d)一个 root 用户拥有的网络服务程序”netmonitor.exe”，需要设置 setuid 位才能完成其功能。

设置 netmonitor.exe 的 setuid 位



2、 一些可执行程序运行时需要系统管理员权限，在 UNIX 中可以利用 setuid 位实现其功能，但 setuid 了的程序运行过程中拥有了 root 权限，因此在完成管理操作后需要切换到普通用户的身份执行后续操作。

(1)设想一种场景，比如提供 http 网络服务，需要设置 setuid 位，并为该场景

编制相应的代码；

创建自己的 http_server.exe，并用代码实现设置 setuid 位

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main()
{
    printf("This is a setuid test.\n");
    if(setuid(getuid())< 0){
        perror("setuid error");
    }
    show_ids();
    system("echo chmod u+s /home/hiter-lxj1170301015/Desktop/ex/lab1/http_server.exe");
    system("chmod u+s /home/hiter-lxj1170301015/Desktop/ex/lab1/http_server.exe");
    system("ls -l /home/hiter-lxj1170301015/Desktop/ex/lab1/http_server.exe");
    system("/home/hiter-lxj1170301015/Desktop/ex/lab1/./http_server.exe");
    return 0;
}

void show_ids()
{
    uid_t ruid,euid,suid;
    getresuid(&ruid,&euid,&suid);
    printf("pid : %d\n",getpid());
    printf("real uid: %d\n",ruid);
    printf("effective uid: %d\n",euid);
    printf("saved uid: %d\n",suid);
}
```

设置 setuid 位

```
hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 ./setuid
This is a setuid test.
pid : 4830
real uid: 1000
effective uid: 1000
saved uid: 1000
chmod u+s /home/hiter-lxj1170301015/Desktop/ex/lab1/http_server.exe
-rwsrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 8304 12月   1 11:20 /home/hi
ter-lxj1170301015/Desktop/ex/lab1/http_server.exe
This is a http server test.
```

实现效果

(2)如果用户 fork 进程后，父进程和子进程中 euid、ruid、suid 的差别；

fork 之后父子进程的 euid，ruid，suid 没有变化，子进程继承了父进程的三个 userID。

```c
int main()
{
    uid_t ruid,euid,suid;
    printf("This is a fork test.\n");
    if(fork()==0){
        getresuid(&ruid,&euid,&suid);
        printf("Son:");
        printf("pid : %d\n",getpid());
        printf("real uid: %d\n",ruid);
        printf("effective uid: %d\n",euid);
        printf("saved uid: %d\n",suid);
    }
    else{
        getresuid(&ruid,&euid,&suid);
        printf("Father:");
        printf("pid : %d\n",getpid());
        printf("real uid: %d\n",ruid);
        printf("effective uid: %d\n",euid);
        printf("saved uid: %d\n",suid);
    }
    return 0;
}
```

创建子进程

实现 fork

```
hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 ./fork
Father:pid : 4899
real uid: 1000
effective uid: 1000
saved uid: 1000
Son:pid : 4900
real uid: 1000
effective uid: 1000
saved uid: 1000
```

实现效果

(3)利用 execl 执行 setuid 程序后，euid、ruid、suid 是否有变化；

这里需要开启一个子进程，否则 execl 程序会覆盖我们原本的程序，执行之后我们再次查看 euid、ruid、suid。

```c
int main()
{
    uid_t ruid,euid,suid;
    pid_t pid;
    if((pid=fork())==0){
        printf("./execl:\n");
        execl("/home/hiter-lxj1170301015/desktop/ex/my_execl","./my_execl",NULL);
    }
    else if (pid > 0){
        printf("This is a execl test.\n");
        getresuid(&ruid,&euid,&suid);
        printf("fpid : %d\n",getpid());
        printf("real uid: %d\n",ruid);
        printf("effective uid: %d\n",euid);
        printf("saved uid: %d\n",suid);
    }
    return 0;
}
```

execl 执行 setuid 程序

```c
int main()
{
    uid_t ruid,euid,suid;
    getresuid(&ruid,&euid,&suid);
    printf("real uid: %d\n",ruid);
    printf("effective uid: %d\n",euid);
    printf("saved uid: %d\n",suid);
    return 0;
}
```

My_execl 程序

```
hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 ./after_execl
This is a execl test.
fpid : 5018
real uid: 1000
effective uid: 1000
saved uid: 1000
./execl:
spid : 5019
real uid: 1000
effective uid: 0
saved uid: 0
```

实现效果

利用 execl 运行 my_execl 后 euid 不变，而 euid 和 suid 变为了 root，是因为文件的拥有者为 root。

(4)程序何时需要临时性放弃 root 权限，何时需要永久性放弃 root 权限，并在程序中分别实现两种放弃权限方法；

创建 my_root（权限 700，除 root 用户外，其他人无任何权限），使用 setresuid(1000,1000,0)临时放弃权限，使用 setresuid(1000,1000,1000)永久放弃权限。

```c
int main()
{
    uid_t ruid,euid,suid;
    printf("1.以root身份运行: \n");
    getresuid(&ruid,&euid,&suid);
    system("./my_root");
    printf("ruid: %d\t euid: %d\t suid: %d\n",ruid,euid,suid);
    printf("2.临时放弃root身份: \n");
    setresuid(1000,1000,0);
    system("./my_root");
    getresuid(&ruid,&euid,&suid);
    printf("ruid: %d\t euid: %d\t suid: %d\n",ruid,euid,suid);
    printf("3.重新获取root身份: \n");
    setresuid(0,0,0);
    system("./my_root");
    getresuid(&ruid,&euid,&suid);
    printf("ruid: %d\t euid: %d\t suid: %d\n",ruid,euid,suid);
    printf("4.永久放弃root身份: \n");
    setresuid(1000,1000,1000);
    system("./my_root");
    getresuid(&ruid,&euid,&suid);
    printf("ruid: %d\t euid: %d\t suid: %d\n",ruid,euid,suid);
    if(setresuid(0,0,0)<0){
        printf("重新获取root失败\n");
    }
    return 0;
}
```

弃权方法实现

```
hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 sudo ./changeroot

1.以root身份运行：
ruid: 0  euid: 0         suid: 0
THis is a change root test.
2.临时放弃root身份：
ruid: 1000       euid: 1000      suid: 0
sh: 1: /home/hiter-lxj1170301015/Desktop/ex/./my_root: Permission denied
3.重新获取root身份：
ruid: 0  euid: 0         suid: 0
THis is a change root test.
4.永久放弃root身份：
ruid: 1000       euid: 1000      suid: 1000
sh: 1: /home/hiter-lxj1170301015/Desktop/ex/./my_root: Permission denied
Can't reset resuid!
```

实现效果

临时放弃权限需求

```
void giveupRootTemporary(uid_t uid_tran)
{
    uid_t ruid, euid,suid;
    getresuid(&ruid, &euid, &suid);
    if (euid == 0)
    {
        // 临时性放弃root权限
        int is seteuid = seteuid(uid_tran);
        getresuid(&ruid, &euid, &suid);
        if(euid > 0)
        {
            printf("------ 1 临时性放弃root权限成功 ------\n");
        }
        else
        {
            printf("------ 1 临时性放弃root权限失败 ------\n");
        }
        printf("ruid = %d, euid = %d, suid = %d\n", ruid, euid, suid);
    }
    else
    {
        printf("1 无 root 权限，无法放弃root权限\n");
    }
}
```

永久放弃权限需求

```c
void giveupRootPermanent(uid_t uid_tran)
{
    uid_t ruid, euid,suid;
    getresuid(&ruid, &euid, &suid);
    if (euid != 0 && (ruid == 0 || suid == 0))
    {
        setuid(0);
        getresuid(&ruid, &euid, &suid);
    }
    if (euid == 0)
    {
        // 永久性放弃root权限
        setresuid(uid_tran, uid_tran, uid_tran);
        getresuid(&ruid, &euid, &suid);
        if(ruid > 0 && euid > 0 && suid > 0)
        {
            printf("------ 2 永久性放弃root权限成功 ------\n");
        }
        else
        {
            printf("------ 2 永久性放弃root权限失败 ------\n");
        }
        printf("ruid = %d, euid = %d, suid = %d\n", ruid, euid, suid);
    }
    else
    {
        printf("2 无 root 权限，无法放弃root权限\n");
    }
}
```

```
hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 sudo ./giveup
[sudo] hiter-lxj1170301015 的密码：
------ 1 临时性放弃root权限成功 ------
ruid = 0, euid = 1001, suid = 0
------ 2 永久性放弃root权限成功 ------
ruid = 1001, euid = 1001, suid = 1001
```

实现效果

(5)execl 函数族中有多个函数，比较有环境变量和无环境变量的函数使用的差异。

在 execl 函数族中 p 的是有环境变量的，不带 p 的是无环境变量的。其中有环境变量的函数可以只输入一个文件名，在执行的时候会自动从环境变量中找与这个文件名相同的程序，而不带环境变量的则需要指名程序的绝对路径或者相对路径。

以下作为验证：

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main()
{
    //test1();
    //test2();
    test3();
    return 0;
}

void test1(){
    printf("1.使用execl函数，不指定路径.\n");
    execl("ls","ls","-l",NULL);
}

void test2(){
    printf("2.使用execlp函数，不指定路径.\n");
    execlp("ls","ls","-l",NULL);
}

void test3(){
    printf("3.使用execl函数，指定路径.\n");
    execl("/bin/ls","ls","-l",NULL);
}
```

```
 hiter-lxj1170301015@hiter-lxj1170301015  ~/Desktop/ex/lab1  ./test1_execl
1.使用execl函数，不指定路径.
 hiter-lxj1170301015@hiter-lxj1170301015  ~/Desktop/ex/lab1  ./test2_execl
```

Test1()实现效果

```
 hiter-lxj1170301015@hiter-lxj1170301015  ~/Desktop/ex/lab1  ./test2_execl
2.使用execlp函数，不指定路径.
总用量 120
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11536 12月   1 13:24 after_execl
-rw-rw-r-- 1 hiter-lxj1170301015 hiter-lxj1170301015    85 12月   3 02:30 cal.c
-rwsrwxr-x 1 hiter-lxj1170301015 hiter-lxj1170301015  8296 12月   3 02:31 cal.exe
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11480 12月   1 13:17 changeroot
-rw-rw-r-- 1 hiter-lxj1170301015 hiter-lxj1170301015    21 12月   3 02:37 demo.txt
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11432 11月  30 10:20 fork
-rwsrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015  8304 12月   1 11:20 http_server.exe
-rwsrwxr-x 1 root                root                8296 12月   3 02:39 netmonitor.exe
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11648 12月   3 02:49 setuid
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11328 12月   3 03:01 test1_execl
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11328 12月   3 03:02 test2_execl
-rwxr----- 1 hiter-lxj1170301015 hiter-lxj1170301015    26 12月   3 02:24 流星雨.txt
```

Test2()实现效果

```
3.使用execl函数，指定路径.
总用量 132
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11536 12月   1 13:24 after_execl
-rw-rw-r-- 1 hiter-lxj1170301015 hiter-lxj1170301015    85 12月   3 02:30 cal.c
-rwsrwxr-x 1 hiter-lxj1170301015 hiter-lxj1170301015  8296 12月   3 02:31 cal.exe
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11480 12月   1 13:17 changeroot
-rw-rw-r-- 1 hiter-lxj1170301015 hiter-lxj1170301015    21 12月   3 02:37 demo.txt
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11432 11月  30 10:20 fork
-rwsrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015  8304 12月   1 11:20 http_server.exe
-rwsrwxr-x 1 root                root                8296 12月   3 02:39 netmonitor.exe
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11648 12月   3 02:49 setuid
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11328 12月   3 03:01 test1_execl
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11328 12月   3 03:02 test2_execl
-rwxrwxrwx 1 hiter-lxj1170301015 hiter-lxj1170301015 11328 12月   3 03:03 test3_execl
-rwxr----- 1 hiter-lxj1170301015 hiter-lxj1170301015    26 12月   3 02:24 流星雨.txt
```

Test3()实现效果

3、 编制实验报告，对问题一说明原理，对问题 2 说明设计过程和实验步骤。并写出心得体会。

心得体会：本次实验自己动手操作了有关于 linux 系统中文件和目录的权限管理部分的内容，更加深刻地理解了 ruid，euid，suid 之间的区别与联系；清楚了通过 fork 创建的子进程与父进程之间的关系。进一步学习了 execl 函数族的相关知识。通过本次实验，复习了课上老师讲过的内容，知其然还要知其所以然。

1.2 chroot 的配置

1、 利用 chroot 工具来虚拟化管理

1) 实现 bash 或 ps 的配置使用；

添加新用户 test，创建用户必要的文件夹，并更改其属主。

```
root@hiter-lxj1170301015:/home/hiter-lxj1170301015# sudo useradd -M test
root@hiter-lxj1170301015:/home/hiter-lxj1170301015# passwd test
输入新的 UNIX 密码：
重新输入新的 UNIX 密码：
passwd：已成功更新密码
root@hiter-lxj1170301015:/home/hiter-lxj1170301015# mkdir -p /var/chroot/home/te
st
root@hiter-lxj1170301015:/home/hiter-lxj1170301015# chown -R test.test /var/chro
ot/home/test
root@hiter-lxj1170301015:/home/hiter-lxj1170301015# chmod 700 /var/chroot/home/t
est
root@hiter-lxj1170301015:/home/hiter-lxj1170301015# cd /var/chroot
root@hiter-lxj1170301015:/var/chroot# mkdir -p {etc,dev,proc,lib,usr}
root@hiter-lxj1170301015:/var/chroot# cd home
root@hiter-lxj1170301015:/var/chroot/home# ls
test
root@hiter-lxj1170301015:/var/chroot/home# cp -a /etc/passwd /var/chroot/etc/pas
swd
root@hiter-lxj1170301015:/var/chroot/home# cp -a /etc/group /var/chroot/etc
```

```
root@hiter-lxj1170301015:/var/chroot/etc# vim group
root@hiter-lxj1170301015:/var/chroot/etc# vim passwd
root@hiter-lxj1170301015:/var/chroot/etc# cat group
root:x:0:
test:x:1001:
root@hiter-lxj1170301015:/var/chroot/etc# cat passwd
root:x:0:0:root:/root:/bin/bash
test:x:1001:1001::/home/test:/bin/sh
```

查看 bash 命令需要哪些.so 文件，拷贝到相应目录中

```
root@hiter-lxj1170301015:/# cd /var/chroot/usr
root@hiter-lxj1170301015:/var/chroot/usr# mkdir -p {bin,lib,libexec}
mkdir: 无法创建目录"bin": 文件已存在
root@hiter-lxj1170301015:/var/chroot/usr# rm -rf bin
root@hiter-lxj1170301015:/var/chroot/usr# mkdir -p {bin,lib,libexec}
root@hiter-lxj1170301015:/var/chroot/usr# cp -a /usr/bin/ftp /var/chroot/usr/bin
/
```

```
root@hiter-lxj1170301015:/var/chroot/etc# ldd /bin/bash
        linux-vdso.so.1 (0x00007ffe627c2000)
        libtinfo.so.5 => /lib/x86_64-linux-gnu/libtinfo.so.5 (0x00007f7302e42000
)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f7302c3e000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f730284d000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f7303386000)
```

拷贝动态库文件，拷贝完成后，发现拷贝的是软链接文件

```
root@hiter-lxj1170301015:/var/chroot/lib# sudo cp -a /lib/x86_64-linux-gnu/{libtinfo.so.
5,libdl.so.2,libc.so.6} /var/chroot/lib/x86_64-linux-gnu
root@hiter-lxj1170301015:/var/chroot/lib# ls
x86_64-linux-gnu
root@hiter-lxj1170301015:/var/chroot/lib# cd x86_64-linux-gnu
root@hiter-lxj1170301015:/var/chroot/lib/x86_64-linux-gnu# ls
libc.so.6  libdl.so.2  libtinfo.so.5
root@hiter-lxj1170301015:/var/chroot/lib/x86_64-linux-gnu# ls -l
总用量 0
lrwxrwxrwx 1 root root 12 11月  2 23:44 libc.so.6 -> libc-2.27.so
lrwxrwxrwx 1 root root 13 11月  2 23:44 libdl.so.2 -> libdl-2.27.so
lrwxrwxrwx 1 root root 15 11月  2 23:44 libtinfo.so.5 -> libtinfo.so.5.9
```

拷贝原始文件，然后测试一下，正常显示 bash 信息

```
root@hiter-lxj1170301015:/var/chroot/lib/x86_64-linux-gnu# sudo cp -a /lib/x86_64-linux-
gnu/{libc-2.27.so,libdl-2.27.so,libtinfo.so.5.9} /var/chroot/lib/x86_64-linux-gnu
root@hiter-lxj1170301015:/var/chroot/lib/x86_64-linux-gnu# ls -l
总用量 2168
-rwxr-xr-x 1 root root 2030544 4月  17  2018 libc-2.27.so
lrwxrwxrwx 1 root root      12 11月  2 23:44 libc.so.6 -> libc-2.27.so
-rw-r--r-- 1 root root   14560 4月  17  2018 libdl-2.27.so
lrwxrwxrwx 1 root root      13 11月  2 23:44 libdl.so.2 -> libdl-2.27.so
lrwxrwxrwx 1 root root      15 11月  2 23:44 libtinfo.so.5 -> libtinfo.so.5.9
-rw-r--r-- 1 root root  170784 5月  23  2018 libtinfo.so.5.9
root@hiter-lxj1170301015:/var/chroot/lib/x86_64-linux-gnu# cd ..
root@hiter-lxj1170301015:/var/chroot/lib# cd ..
root@hiter-lxj1170301015:/var/chroot# mkdir lib64
root@hiter-lxj1170301015:/var/chroot# cp -a /lib64/ld-linux-x86-64.so.2 /var/chroot/lib6
4
root@hiter-lxj1170301015:/var/chroot# cd lib64
root@hiter-lxj1170301015:/var/chroot/lib64# ls -l
总用量 0
lrwxrwxrwx 1 root root 32 11月  2 23:44 ld-linux-x86-64.so.2 -> /lib/x86_64-linux-gnu/ld
-2.27.so
root@hiter-lxj1170301015:/var/chroot/lib64# cp -a /lib/x86_64-linux-gnu/ld-2.27.so /var/
chroot/lib/x86_64-linux-gnu
```

此时 chroot 在/var/chroot 文件夹的环境已经配置完成

```
root@hiter-lxj1170301015:/var/chroot/lib64# cd ..
root@hiter-lxj1170301015:/var/chroot# sudo chroot /var/chroot
bash-4.4# exit
exit
```

2) 利用 chroot 实现 SSH 服务或 FTP 服务的虚拟化隔离；

这里使用 pure-ftpd

```
root@hiter-lxj1170301015:/var/chroot# apt-get install pure-ftpd
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
将会同时安装下列软件：
  pure-ftpd-common
下列软件包将被【卸载】：
  ftpd
下列【新】软件包将被安装：
  pure-ftpd pure-ftpd-common
升级了 0 个软件包，新安装了 2 个软件包，要卸载 1 个软件包，有 0 个软件包未被升级
。
```

创建用户

```
root@hiter-lxj1170301015:/var/chroot# groupadd ftpgroup
root@hiter-lxj1170301015:/var/chroot# useradd -g ftpgroup -d /home/ftpuser -s /s
bin/nologin ftpuser
root@hiter-lxj1170301015:/var/chroot# pure-pw useradd lxj -u ftpuser -d /home/ft
puser/lxj
Password:
Enter it again:
root@hiter-lxj1170301015:/var/chroot# pure-pw mkdb
```

打开服务

```
root@hiter-lxj1170301015:/var/chroot# service pure-ftpd start
root@hiter-lxj1170301015:/var/chroot# ftp 127.0.0.1
Connected to 127.0.0.1.
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 1 of 50 allowed.
220-Local time is now 01:36. Server port: 21.
220-This is a private system - No anonymous login
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
```

登录查看

```
Name (127.0.0.1:hiter-lxj1170301015): hiter-lxj1170301015
331 User hiter-lxj1170301015 OK. Password required
Password:
230 OK. Current directory is /home/hiter-lxj1170301015
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful
150 Connecting to port 41141
drwxr-xr-x    5 hiter-lxj1 hiter-lxj1       4096 Dec  1 11:27 Desktop
drwxr-xr-x    2 hiter-lxj1 hiter-lxj1       4096 Nov  2 23:48 Documents
drwxr-xr-x    2 hiter-lxj1 hiter-lxj1       4096 Nov  2 23:48 Downloads
drwxr-xr-x    2 hiter-lxj1 hiter-lxj1       4096 Nov  2 23:48 Music
drwxr-xr-x    2 hiter-lxj1 hiter-lxj1       4096 Dec  1 11:09 Pictures
drwxr-xr-x    2 hiter-lxj1 hiter-lxj1       4096 Nov  2 23:48 Public
drwxr-xr-x    2 hiter-lxj1 hiter-lxj1       4096 Nov  2 23:48 Templates
drwxr-xr-x    2 hiter-lxj1 hiter-lxj1       4096 Nov  2 23:48 Videos
226-Options: -l
226 8 matches total
ftp> exit
221-Goodbye. You uploaded 0 and downloaded 0 kbytes.
221 Logout.
root@hiter-lxj1170301015:/var/chroot# service pure-ftpd stop
```

复制所需文件

```
root@hiter-lxj1170301015:/var/chroot# whereis pure-ftpd
pure-ftpd: /usr/sbin/pure-ftpd /etc/pure-ftpd /usr/share/man/man8/pure-ftpd.8.gz
root@hiter-lxj1170301015:/var/chroot# cp /usr/sbin/pure-ftpd ./ --parent
root@hiter-lxj1170301015:/var/chroot# cp -r /etc/pure-ftpd/* ./ --parent
root@hiter-lxj1170301015:/var/chroot# cp /usr/share/man/man8/pure-ftpd.8.gz ./ -
-parent
```

查看所需链接

```
root@hiter-lxj1170301015:/var/chroot# ldd /usr/sbin/pure-ftpd
        linux-vdso.so.1 (0x00007ffde5ff7000)
        libssl.so.1.1 => /usr/lib/x86_64-linux-gnu/libssl.so.1.1 (0x00007f480e54
0000)
        libcrypto.so.1.1 => /usr/lib/x86_64-linux-gnu/libcrypto.so.1.1 (0x00007f
480e075000)
        libcrypt.so.1 => /lib/x86_64-linux-gnu/libcrypt.so.1 (0x00007f480de3d000
)
        libcap.so.2 => /lib/x86_64-linux-gnu/libcap.so.2 (0x00007f480dc37000)
        libpam.so.0 => /lib/x86_64-linux-gnu/libpam.so.0 (0x00007f480da29000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f480d638000)
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f480d41
9000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f480d215000)
        libaudit.so.1 => /lib/x86_64-linux-gnu/libaudit.so.1 (0x00007f480cfec000
)
        /lib64/ld-linux-x86-64.so.2 (0x00007f480ea11000)
        libcap-ng.so.0 => /lib/x86_64-linux-gnu/libcap-ng.so.0 (0x00007f480cde70
00)
```

复制链接

```
root@hiter-lxj1170301015:/var/chroot# cp /usr/lib/x86_64-linux-gnu/libssl.so.1.1
 ./ --parent
root@hiter-lxj1170301015:/var/chroot# cp /usr/lib/x86_64-linux-gnu/libcrypto.so.
1.1 ./ --parent
root@hiter-lxj1170301015:/var/chroot# cp /lib/x86_64-linux-gnu/libcrypt.so.1 ./
--parent
root@hiter-lxj1170301015:/var/chroot# cp /lib/x86_64-linux-gnu/libcap.so.2 ./ --
parent
root@hiter-lxj1170301015:/var/chroot# cp /lib/x86_64-linux-gnu/libpam.so.0 ./ --
parent
root@hiter-lxj1170301015:/var/chroot# cp /lib/x86_64-linux-gnu/libc.so.6 ./ --pa
rent
root@hiter-lxj1170301015:/var/chroot# cp /lib/x86_64-linux-gnu/libpthread.so.0 .
/ --parent
root@hiter-lxj1170301015:/var/chroot# cp /lib/x86_64-linux-gnu/libdl.so.2 ./ --p
arent
root@hiter-lxj1170301015:/var/chroot# cp /lib/x86_64-linux-gnu/libaudit.so.1 ./
--parent
root@hiter-lxj1170301015:/var/chroot# cp /lib64/ld-linux-x86-64.so.2 ./ --parent
cp: '/lib64/ld-linux-x86-64.so.2' 与 './lib64/ld-linux-x86-64.so.2' 为同一文件
root@hiter-lxj1170301015:/var/chroot# cp /lib/x86_64-linux-gnu/libcap-ng.so.0 ./
 --parent
```

后续操作

```
root@hiter-lxj1170301015:/var/chroot# mkdir dev
mkdir: 无法创建目录"dev": 文件已存在
root@hiter-lxj1170301015:/var/chroot# ls
bin  dev  etc  home  lib  lib64  proc  usr
root@hiter-lxj1170301015:/var/chroot# mknod -m 666 dev/null c 1 3
root@hiter-lxj1170301015:/var/chroot# mknod dev/urandom c 1 9
root@hiter-lxj1170301015:/var/chroot# chroot ./
bash-4.4# pure-ftpd -j -lpuredb:/etc/pure-ftpd/pureftpd.pdb
```

此时已打开服务

```
✗ hiter-lxj1170301015@hiter-lxj1170301015 ~ ftp 127.0.0.1
Connected to 127.0.0.1.
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 1 of 50 allowed.
220-Local time is now 17:43. Server port: 21.
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
Name (127.0.0.1:hiter-lxj1170301015): lxj
331 User lxj OK. Password required
Password:
230 OK. Current directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful
150 Connecting to port 47065
226-Options: -l
226 0 matches total
```

新建 test 文件夹，用于测试

```
ftp> mkdir test
257 "test" : The directory was successfully created
ftp> ls
200 PORT command successful
150 Connecting to port 39565
drwxr-xr-x    2 1002       1002            4096 Dec  2 17:44 test
226-Options: -l
226 1 matches total
ftp> cd ../../../../
250 OK. Current directory is /
ftp> ls
200 PORT command successful
150 Connecting to port 39145
drwxr-xr-x    2 1002       1002            4096 Dec  2 17:44 test
226-Options: -l
226 1 matches total
ftp> exit
221-Goodbye. You uploaded 0 and downloaded 0 kbytes.
221 Logout.
```

此时已实现虚拟化隔离。

3) chroot 后如何降低权限，利用实验一中编制的程序检查权限的合理性；

因为 chroot 是以 root 身份运行的，所以我们在 chroot 进入环境的时候还是 root 权限，这是很不安全的行为。所以，应该在 chroot 之后应该主动的放弃权限，即应该快速的恢复到普通用户的身份，防止攻击者攻破 jail 后获得 root 权限并使用 root 权限来损坏我们的电脑。

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

int main()
{
    uid_t ruid, euid, suid;
    getresuid(&ruid, &euid, &suid);
    printf("before:\nruid:%d\teuid:%d\tsuid:%d\n", ruid, euid, suid);
    printf("now chdir\n");
    chdir("/var/chroot");
    printf("now chroot\n");
    if(chroot("/var/chroot")==0){
        printf("chroot succeed!\n");
    }
    getresuid(&ruid, &euid, &suid);
    printf("now:\nruid:%d\teuid:%d\tsuid:%d\n", ruid, euid, suid);
    printf("now give up root permission...\n");
    setresuid(1001, 1001, 1001);
    getresuid(&ruid, &euid, &suid);
    printf("now:\nruid:%d\teuid:%d\tsuid:%d\n", ruid, euid, suid);
    execlp("ls","ls",NULL);
    return 0;

}
```

代码实现

成功 chroot 后，立即降低权限

```
✗ hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 sudo ./mychroot

before:
ruid:0  euid:0  suid:0
now chdir
now chroot
chroot succeed!
now:
ruid:0  euid:0  suid:0
now give up root permission...
now:
ruid:1001       euid:1001       suid:1001
bin  dev  etc  home  lib  lib64  proc  usr
```

实现效果

4) 在 chroot 之前没有采用 cd xx 目录，会对系统有何影响，编制程序分析其影响。

注释掉 3）中的 chdir 部分，代码如下：

```c
int main()
{
    uid_t ruid, euid, suid;
    getresuid(&ruid, &euid, &suid);
    printf("before:\nruid:%d\teuid:%d\tsuid:%d\n", ruid, euid, suid);
    printf("now chdir\n");
    //chdir("/var/chroot");
    printf("now chroot\n");
    if(chroot("/var/chroot")==0){
        printf("chroot succeed!\n");
    }
    getresuid(&ruid, &euid, &suid);
    printf("now:\nruid:%d\teuid:%d\tsuid:%d\n", ruid, euid, suid);
    printf("now give up root permission...\n");
    setresuid(1001, 1001, 1001);
    getresuid(&ruid, &euid, &suid);
    printf("now:\nruid:%d\teuid:%d\tsuid:%d\n", ruid, euid, suid);
    execlp("ls","ls",NULL);
    return 0;

}
```

可以看到，在 chroot 之前没有到相应目录下一定会导致越狱事件的产生，因为 chroot 只会改变根目录不会改变工作目录。

```
hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1 sudo ./mychrootwi
thourchdir
before:
ruid:0  euid:0  suid:0
now chdir
now chroot
chroot succeed!
now:
ruid:0  euid:0  suid:0
now give up root permission...
now:
ruid:1001       euid:1001       suid:1001
after_execl         mychroot
cal.c               mychrootwithourchdir
cal.exe             netmonitor.exe
changeroot          setuid
demo.txt            test1_execl
fork                test2_execl
giveup              test3_execl
http_server.exe  ''$'\346\265\201\346\230\237\351\233\250''.txt'
hiter-lxj1170301015@hiter-lxj1170301015 ~/Desktop/ex/lab1
```

实现效果

可参考后面的文件内容进行处理。

2、 编制实验报告，说明原理，设计过程和实验步骤。并写出心得体会。

尝试了三天，终于学会了如何配置 chroot，实现了 ftpd 的虚拟化隔离。刚接触 linux 系统，对于一些命令行还不熟悉，所以觉得这次实验真的很麻烦，还好最后做出来了。

Ubuntu 下 chroot FTP 服务

1、准备基本的 chroot 环境

在进入 chroot 环境之前要先准备好相应的设置，在本例中是将 ftpd chroot 到/var/chroot 目录中。因为系统自带的 ftpd 在/usr/libexec/目录，所以需要在/var/chroot 中执行以下操作：

#mkdir -p /var/chroot/usr/libexec

然后将 ftpd 复制到该目录中：

#install -C /usr/libexec/ftpd /var/chroot/usr/libexec

将 ftpd 需要的库也复制到 chroot 目录中，使用 ldd 来检测 ftpd 运行时需要哪些库文件：

ldd /usr/libexec/ftpd

/usr/libexec/ftpd:

libskey.so.2 => /usr/lib/libskey.so.2 (0x28074000)

libmd.so.2 => /usr/lib/libmd.so.2 (0x2807b000)

……

ldd 的运行结果显示了 ftpd 运行时需要库，把这些库安装到 chroot 的相应目录中：

mkdir -p /var/chroot/usr/lib

install -C /usr/lib/libskey.so.2 /var/chroot/usr/lib

install -C /usr/lib/libmd.so.2 /var/chroot/usr/lib

……

2、配置 chroot 环境

2.1 检查 ftpd 是否能在 chroot 环境中运行：

chroot /var/chroot /usr/libexec/ftpd

ELF interpreter /usr/libexec/ld-elf.so.1 not found

程序出错，根据提示在/usr/libexec 中还缺少文件 ld-elf.so.1，由于 ftpd 是在 chroot 环境中运行，所以应将 ld-elf.so.1 复制到 chroot 环境中，即/var/chroot/usr/libexec 中：

install -C /usr/libexec/ld-elf.so.1 /var/chroot/usr/libexec

2.2 再次尝试进入 chroot 环境：

chroot /var/chroot /usr/libexec/ftp

这次没有任何提示说明运行库已经准备好了，但是由于 ftpd 在不带-D 参数的时候运行完后就会自动退出，所以现在还是无法从远程登录 ftp 服务，试着在 ftpd 后面加上参数-D：

chroot /var/chroot /usr/libexec/ftpd –D

结果与上次一样，通过查阅 chroot(8)的手册，可以看到 chroot 的语法是：

chroot newroot [command]

也就是说 command 后面不能带参数,即然这样可以写一个简单的 shell 脚本来运行 ftpd，脚本命名为 ftpd.sh，存放于/var/chroot/usr/libexec 中，内容为：

#!/bin/sh

#/usr/libexec/ftpd -D -4

由于不需要支持 IPv6，所以这里加上了参数-4 只对 IPv4 提供支持，当然也可以加上一些其它参数。接下来为脚本加上执行权限：

#chmod a+x /var/chroot/usr/libexec/ftpd.sh

为了要运行这个脚本程序，还需要将/bin/sh 到的 chroot 环境中：

#mkdir /var/chroot/bin

#install -C /bin/sh /var/chroot/bin

接下来就要为 chroot 环境准备/etc 目录了。首先要复制的就是/etc/services 文件，因为它定义了 ftpd 使用的端口号和协议：

mkdir /var/chroot/etc

cp /etc/services /var/chroot/etc

因为需要验证用户，所以需要复制 master.passwd 和 group：

cp /etc/group /var/chroot/etc

cp /etc/master.passwd /var/chroot/etc

编辑/var/chroot/etc/master.passwd 和/var/chroot/etc/group，删除不需要使用 ftp 的用户和不必要的组，注意，当更改了 master.passwd 后一定要使用 pwd_mkdb 来生成密码数据库，由于此时需要将密码数据库文件存放在/var/chroot/etc 中，而不是默认/etc 中，所以在 pwd_mkdb 后面加上-d 参数来指定数据库存放位置：

#pwd_mkdb -d /var/chroot/etc /var/chroot/etc/master.passwd

此时如果执行成功的话将会在/var/chroot/etc/目录中多“pwd.db、spwd.db”两个文件。

2.3 再次进入 chroot 环境：

#chroot /var/chroot /usr/libexec/ftpd.sh

现在便可以登录到 chroot 环境下的 ftp 服务器了。

3、结尾工作

为每一个用户建立 home 目录，注意是在建在/var/chroot/home 之中。在/var/chroot/etc/中生成 ftpusers 文件，将禁止登录 ftp 的用户的用户名加入其中，以禁止部分用户登录。

在/var/chroot/etc/中生成 ftpchroot 文件，它的作用是限制用户只能访问自己的 home 目录中的文件，而不能访问 home 外的任何内容。将要限制的用户用户名加入其中。

在/var/chroot/etc/中生成 ftpwelcome 文件，它的作用是当用户连接上服务器的时候显示欢迎信息。在/var/chroot/etc/中生成 ftpmotd 文件，它的作用是当用户登录进服务器的时候显示欢迎信息。