

计算机安全实验

实验一

姓名：卢兑琬

学号：L170300901

实验一（10 分）

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：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：2 人一组,分组实现（分工明确,每人讲解自己的部分,并能够相互配合）

1.1 Linux 系统文件权限设置与辨识 setuid 程序 uid 差异（5 分）

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；



```
fengjlnghang@fengjlnghang-ThinkPad-T470p:~$ ls -l /usr/bin/passwd
-rwxr-xr-x 1 root root 59640 3月 23 2019 /usr/bin/passwd
```

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，

例如：sudo, passwd

```
fengjinghang@fengjinghang-ThinkPad-T470p:~$ ls -l /usr/bin/sudo
-rwsr-xr-x 1 root root 149088 2月  1 2020 /usr/bin/sudo
fengjinghang@fengjinghang-ThinkPad-T470p:~$ ls -l /usr/bin/passwd
-rwxr-xr-x 1 root root 59640 3月 23 2019 /usr/bin/passwd
```

去掉 sudo 的 setuid

```
fengjinghang@fengjinghang-ThinkPad-T470p:~$ sudo chmod u-s /usr/bin/sudo
[sudo] fengjinghang 的密码:
fengjinghang@fengjinghang-ThinkPad-T470p:~$ sudo apt-get update
sudo: /usr/bin/sudo 必须属于用户 ID 0(的用户)并且设置 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。

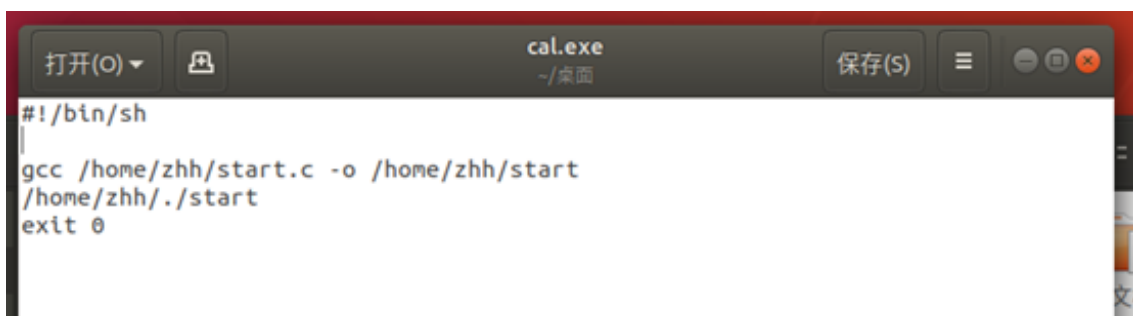
(2) 设置文件或目录权限

a)用户 A 具有文本文件“流星雨.txt”，该用户允许别人下载；

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01$ sudo chmod 744 lxy.txt
[sudo] fengjinghang 的密码:
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01$ ls -l lxy.txt
-rwxr--r-- 1 fengjinghang fengjinghang 31 12月 12 15:38 lxy.txt
```

注意执行权限，组和其他用户只有读权限

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；



创建 cal.exe 脚本文件，文件里内容如上图（按照脚本文件的格式来写），该脚本文件将会编译运行一个 start.c 的文件，该文件将会在根目录下创建一个 start.txt 文件，并且在里面写入一些文字

start.c 文件内容如下:

```
start.c
打开(O) 保存(S)
#include "stdio.h"
#include "time.h"

void main(){
    FILE *fp = fopen("/home/zhh/startout.txt", "w");
    fprintf(fp, "This line will be written when the computer is turned on\n");
    fclose(fp);
}
```

修改脚本文件的权限, 使其成为可执行文件:

```
fengjinghang@fengjinghang-ThinkPad-T470p:~$ ls -l /home/fengjinghang/cal.exe
-rwxr-xr-x 1 fengjinghang fengjinghang 104 12月 12 15:56 /home/fengjinghang/cal.exe
```

利用一下命令, 将脚本文件放置在启动目录 init.d 下:

```
sudo mv cal.exe /etc/init.d/
```

```
活动 终端
fengjinghang@fengjinghang-ThinkPad-T470p:~$ ls /etc/init.d/
acpid cups lightdm speech-dispatcher
alsa-utils cups-browsed networking spice-vdagent
anacron dbus network-manager thermald
apparmor dns-clean openvpn udev
apport gdm3 plymouth ufw
avahi-daemon grub-common plymouth-log unattended-upgrades
binfmt-support hwclock.sh pppd-dns uuidd
bluetooth irqbalance procps whoopsie
brltty kerneloops resolvconf x11-common
cal.exe keyboard-setup.dpkg-bak rsync
console-setup.sh keyboard-setup.sh rsyslog
cron kmod saned
```

打开启动目录, 将脚本添加到启动脚本 (90 表示优先级):

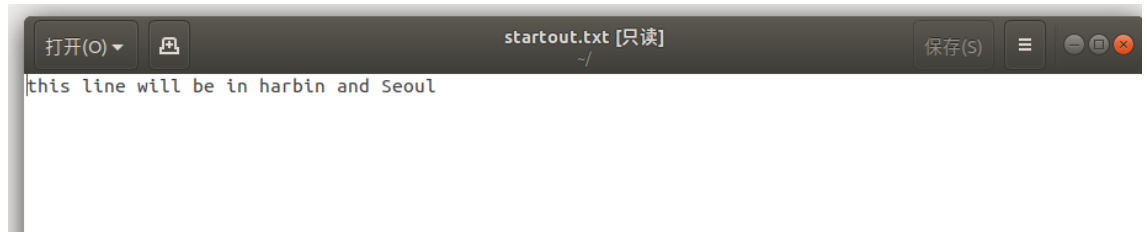
```
cd /etc/init.d/
```

```
sudo update-rc.d cal.exe defaults 90
```

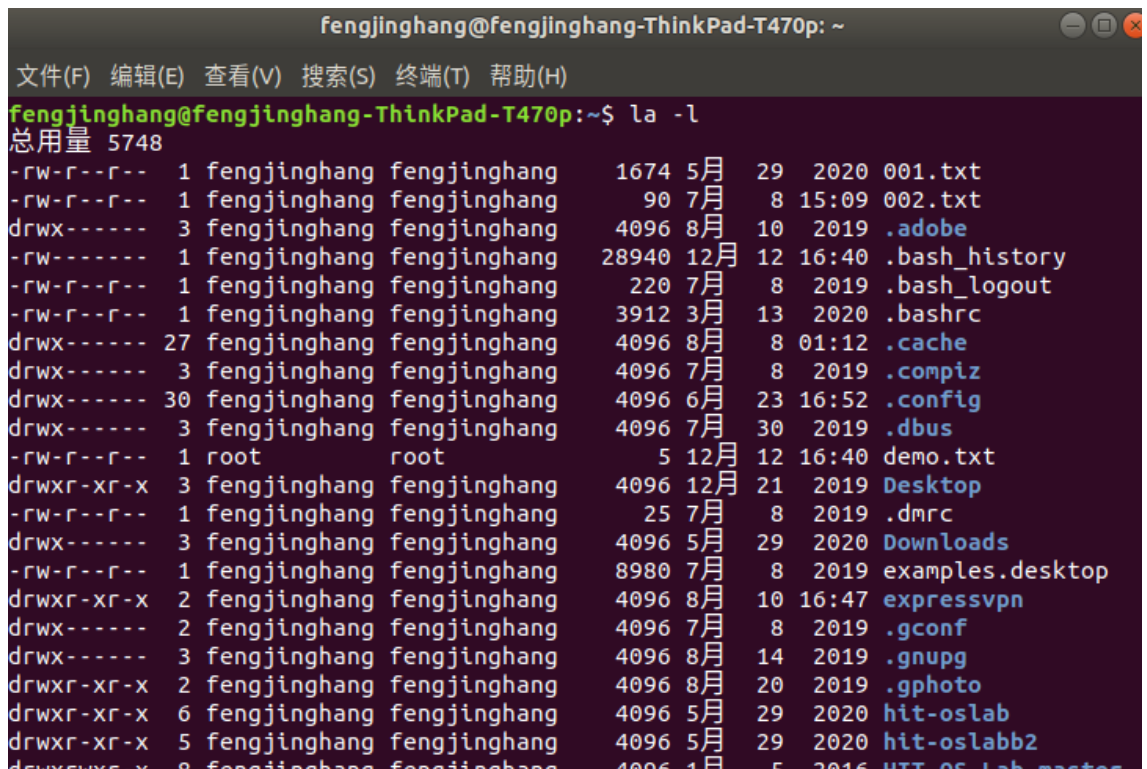
查看启动目录表，如果上述操作后 cal.exe 然没有在启动目录表 0-5 中，则执行如下步骤（创建软连接）：

```
ln -s /etc/init.d/cal.exe /etc/rc2.d/S99cal.exe
```

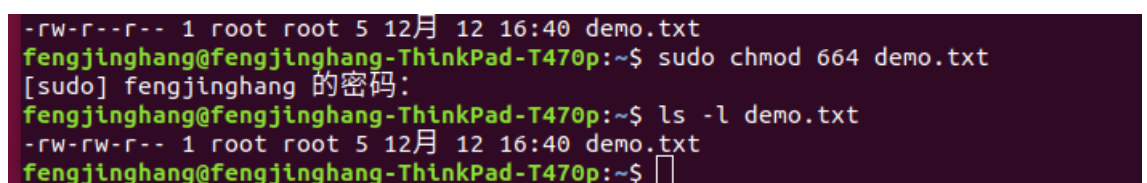
重启，根目录出现相应文件，文件内容和预期一致



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；
用户创建文件的初始权限：



修改权限，让同组用户也可以修改该文件：



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。

创建 netmonitor.exe 脚本：初始权限如下：

```
fengjinghang@fengjinghang-ThinkPad-T470p:~$ vim netmonitor.exe
fengjinghang@fengjinghang-ThinkPad-T470p:~$ ls -l netmonitor.exe
-rw-r--r-- 1 fengjinghang fengjinghang 7 12月 12 16:46 netmonitor.exe
```

修改权限，设置 setuid 位：

```
fengjinghang@fengjinghang-ThinkPad-T470p:~$ sudo chmod 4744 netmonitor.exe
fengjinghang@fengjinghang-ThinkPad-T470p:~$ ls -l netmonitor.exe
-rwsr--r-- 1 fengjinghang fengjinghang 7 12月 12 16:46 netmonitor.exe
fengjinghang@fengjinghang-ThinkPad-T470p:~$
```

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；

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

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

(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；

最后设计的程序权限如下：

client, http, root_only



```
活动 终端
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01$ ls -l
总用量 52
-rwxr-xr-x 1 fengjinghang fengjinghang 8600 12月 12 17:07 client
-rw-rw-rw- 1 fengjinghang fengjinghang 1465 12月 12 17:07 client.c
-rwsr-xr-x 1 fengjinghang fengjinghang 8608 12月 12 17:12 http
-rw-rw-rw- 1 fengjinghang fengjinghang 1455 12月 12 17:12 http.c
drwxr-xr-x 2 fengjinghang fengjinghang 4096 12月 12 17:10 p
-rwxr--r-- 1 fengjinghang fengjinghang 8304 12月 12 16:58 root_only
-rw-rw-rw- 1 fengjinghang fengjinghang 100 12月 12 08:55 root_only.c
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01$ ./client
```

首先以普通身份运行 client，client 中会调用设置了 setuid 位的 http 文件，从而会获得 root 的 uid，http 文件中会反复放弃 root 权限，通过测试能否调用 root_only 文件，可以来体现 http 文件中收放 root 权限的过程。

代码如下:

client:

```
client.c
~/lab1/final
保存(S)

#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

/*客户函数所有者为用户*/
/*调用http程序, http的所有者为root, 同时拥有setuid位*/

void show_id(uid_t *ruid, uid_t *euid, uid_t *suid){
    getresuid(ruid, euid, suid); //获取r, e, suid
    printf("ruid: %d, euid: %d, suid: %d\n", *ruid, *euid, *suid); //打印当前uid号
}

int main(int argc, char *argv[]){
    uid_t ruid, euid, suid;
    pid_t pid;
    printf("***** 客户端进程 *****\n");

    //第一次开启多进程, 测试fork和uid的关系
    printf("**** 测试fork ****\n");

    if( (pid=fork()) < 0 ){
        printf(" fork error ! \n");
    }
    else if (pid == 0){
        printf("子进程1: ");
        show_id(&ruid, &euid, &suid);
        exit(0);
    }
    else{
        printf("父进程1:");
        show_id(&ruid, &euid, &suid);
    }

    sleep(2);
    //第2次开启多进程, 测试execl和setuid的关系, 同时测试root权限的收放
    printf("**** 测试execl ****\n");

    if( (pid=fork()) < 0 ){
        printf(" fork error ! \n");
    }
    else if (pid == 0){
        printf("子进程2: \n");
        execl("http", "./http", NULL);
    }
    else{
        printf("父进程2:");
        show_id(&ruid, &euid, &suid);
    }
    sleep(5);

    printf("***** 客户端进程结束 *****\n");

    return 0;
}
```

http:

```
打开(O)  http.c 保存(S)
~/lab1/final

#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

void show_id(uid_t *ruid, uid_t *euid, uid_t *suid){
    getresuid(ruid, euid, suid); //获取r, e, suid
    printf("ruid: %d, euid: %d, suid: %d\n", *ruid, *euid, *suid); //打印
}

int main(int argc, char *argv[]){
    uid_t ruid, euid, suid;
    pid_t pid;
    printf("***** http server START! *****\n");

    if( (pid=fork()) < 0 ){
        printf(" fork error ! \n");
    }
    else if (pid == 0){
        show_id(&ruid, &euid, &suid); //打印初始uid
        printf("以root身份运行root_only.exe\n");
        execl("root_only", "root_only", NULL);
        exit (0);
    }
    sleep(1);

    seteuid(1000); //临时放弃root权限
    printf("\n临时放弃root权限\n");
    if( (pid=fork()) < 0 ){
        printf(" fork error ! \n");
    }
    else if (pid == 0){
        show_id(&ruid, &euid, &suid);
        printf("以普通身份运行root_only.exe\n");
        execl("root_only", "root_only", NULL);
        exit (0);
    }
    sleep(1);
}
```



```

seteuid(0); //重新获取root权限
printf("\n重新获取root权限\n");
if( (pid=fork()) < 0 ){
    printf(" fork error ! \n");
}
else if (pid == 0){
    show_id(&ruid, &euid, &suid);
    printf("以root身份运行root_only.exe\n");
    execl("root_only", "root_only", NULL);
    exit (0);
}
sleep(1);

setuid(1000); //永久放弃root权限
printf("\n永久放弃root权限\n");
if( (pid=fork()) < 0 ){
    printf(" fork error ! \n");
}
else if (pid == 0){
    show_id(&ruid, &euid, &suid);
    printf("以普通身份运行root_o.exe\n");
    execl("root_only", "root_only", NULL);
    exit (0);
}
sleep(1);

printf("\n尝试重新获取root权限\n");
show_id(&ruid, &euid, &suid);
if(setuid(0)<0){
    printf("尝试重新获取root权限 失败! \n");
}

printf("***** http server END! *****\n");
return 0;

```

root_only:



```

root_only.c
~/lab1/final

#include <stdio.h>
#include <stdlib.h>

void main(){
    printf("root_only is running!\n");
}

```

运行结果如下:

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01$ ./client
***** 客户端进程 *****
**** 测试fork ****
父进程:ruid: 1000,  euid: 1000,  suid: 1000
**** 测试execl ****
父进程:ruid: 1000,  euid: 1000,  suid: 1000
子进程1: ruid: 1000,  euid: 1000,  suid: 1000
***** 客户端进程结束 *****
子进程2:
***** http server START! *****
ruid: 1000,  euid: 1000,  suid: 1000
以root身份运行root_only.exe
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01$ root_only is running!

临时放弃root权限
ruid: 1000,  euid: 1000,  suid: 1000
以普通身份运行root_only.exe
root_only is running!

重新获取root权限
ruid: 1000,  euid: 1000,  suid: 1000
以root身份运行root_only.exe
root_only is running!

永久放弃root权限
ruid: 1000,  euid: 1000,  suid: 1000
以普通身份运行root_o.exe
root_only is running!

尝试重新获取root权限
ruid: 1000,  euid: 1000,  suid: 1000
尝试重新获取root权限 失败!
***** http server END! *****
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01$
```

首先父进程 1 和子进程 1 比较了 fork 之后的 uid 变化, 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。

其次, 父进程 2 和子进程 2 比较了 execl 之后的 uid 变化情况, 利用 execl 运行 http 文件后 ruid 不变, 而 euid 和 suid 变为了 root, 是因为文件的拥有者为 root。

(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。

exec 家族一共有六个函数, 分别是:

```
#include <unistd.h>

int execl(const char *pathname, const char *arg0, ... , (char*)0 );

int execv(const char *pathname, char *const argv[]);

int execl(const char *pathname, const char *arg0, ..., (char *)0, char *const envp[]);
```

```
int execve(const char *pathname, char *const argv[], char *const envp[]);  
int execlp(const char *filename, const char *arg0, ..., (char*)0);  
int execvp(const char *filename, char *const argv);
```

//6 个函数返回值：若出错则返回-1，成功则不返回值。

这几个函数的区别如下：

1. 前 4 个函数取路径名为参数，后 2 个取文件名作为参数。当指定 filename 作为参数时：

(1) 如果 filename 中包含/，则将其作为路径名。

(2) 否则就按 PATH 环境变量，在它所指的各目录中搜索可执行文件。

如果 execlp 和 execvp 使用路径前缀中的一个找到了一个可执行文件，但是该文件不是由连接编辑器产生的机器可执行文件，则认为该文件是一个 shell 脚本，于是试着调用 /bin/sh，并以该 filename 作为 shell 的输入。

2. 第 2 个区别于参数表的传递有关，l 表示 list，v 表示矢量 vector。函数 execl,execlp 和 execl 要求将新程序的每个命令行参数都说明为一个单独的参数，参数表以空指针结尾。另外 3 个函数则应该先构造一个指向各参数的指针数组，然后将该数组地址作为这三个函数的参数。

3. 以 e 结尾的两个函数（execl 和 execve）可以传递一个指向环境字符串指针数组的指针（envp）。其他四个函数则使用调用进程中的 environ 变量为新程序复制现有的环境。

其中以 p 结尾的函数，可以向函数传递一个指向环境字符串指针数组的指针。即自个定义各个环境变量，而其它四个则使用进程中的环境变量。例如，execlp,execvp，表示第一个参数 path 不用输入完整路径，只有给出命令名即可，它会在环境变量 PATH 当中查找命令。

于是我们设计以下程序：

(1)使用 execl 函数，不指定路径

```
打开(O)  execl_null.c
~/lab1/exec

#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

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

运行结果如下：

我们发现对于 ls 指令，execl 函数无法发现路径。

(2)使用 execlp 函数，不指定路径

程序如下：

```
打开(O)  execlp_null.c
~/lab1/exec

#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

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

运行结果如下：

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/02$ sudo ./execlp_null
[sudo] fengjinghang 的密码:
使用execlp函数, 不指定路径
总用量 48
-rwxr-xr-x 1 fengjinghang fengjinghang 8344 12月 12 17:36 execl
-rw-rw-rw- 1 fengjinghang fengjinghang 218 12月 12 08:55 execl.c
-rwxr-xr-x 1 fengjinghang fengjinghang 8352 12月 12 17:33 execl_null
-rw-rw-rw- 1 fengjinghang fengjinghang 215 12月 12 08:55 execl_null.c
-rwxr-xr-x 1 fengjinghang fengjinghang 8352 12月 12 17:35 execlp_null
-rw-rw-rw- 1 fengjinghang fengjinghang 217 12月 12 08:55 execlp_null.c
```

我们可以发现，对于 execlp 函数，可以使用系统环境变量找到 ls 文件。

(3)使用 execl 函数，指定路径

程序如下：

```
打开(O)  execl.c
~/lab1/exec

#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

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

运行结果如下:

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/02$ sudo ./execl
使用execl函数, 指定路径
总用量 48
-rwxr-xr-x 1 fengjinghang fengjinghang 8344 12月 12 17:36 execl
-rw-rw-rw- 1 fengjinghang fengjinghang 218 12月 12 08:55 execl.c
-rwxr-xr-x 1 fengjinghang fengjinghang 8352 12月 12 17:33 execl_null
-rw-rw-rw- 1 fengjinghang fengjinghang 215 12月 12 08:55 execl_null.c
-rwxr-xr-x 1 fengjinghang fengjinghang 8352 12月 12 17:35 execlp_null
-rw-rw-rw- 1 fengjinghang fengjinghang 217 12月 12 08:55 execlp_null.c
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/02$
```

通过结果我们发现加入绝对路径的 `execl` 函数与 `execlp` 函数的结果相同。

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。

本次实验通过，对 `setuid` 位的使用和控制有了更加深入的了解，同时也对 `ruid`, `euid`, `suid` 的区别和作用有了更加清晰的认知。并且通过 `fork` 和 `execl` 函数的使用，对函数的功能和作用，父子进程之间的关系有了更加清晰的了解。

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;

创建 `test` 用户。创建 `var/chroot`，创建必要的目录，拷贝主机的用户信息到 `chroot` 目录下。

```
fengjinghang@fengjinghang-ThinkPad-T470p:~$ su root
密码:
root@fengjinghang-ThinkPad-T470p:/home/fengjinghang# sudo useradd -M test
root@fengjinghang-ThinkPad-T470p:/home/fengjinghang# 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@fengjinghang-ThinkPad-T470p:/home/fengjinghang# mkdir -p /var/chroot/home/test
root@fengjinghang-ThinkPad-T470p:/home/fengjinghang# chown -R test.test /var/chroot/home/test
root@fengjinghang-ThinkPad-T470p:/home/fengjinghang# chmod 700 /var/chroot/home/test/
root@fengjinghang-ThinkPad-T470p:/home/fengjinghang# cd /var/chroot
root@fengjinghang-ThinkPad-T470p:/var/chroot# mkdir -p {etc,dev,proc,lib,usr}
root@fengjinghang-ThinkPad-T470p:/var/chroot# cd home
root@fengjinghang-ThinkPad-T470p:/var/chroot/home# ls
test
root@fengjinghang-ThinkPad-T470p:/var/chroot/home# cp -a /etc/passwd /var/chroot/etc/passwd
root@fengjinghang-ThinkPad-T470p:/var/chroot/home# cp -a /etc/group /var/chroot/etc/group
root@fengjinghang-ThinkPad-T470p:/var/chroot/home#
```

更改 chroot 下的 group, passwd, 删除无用信息, 只保留 root 和 test

```
root@fengjinghang-ThinkPad-T470p:/var/chroot/etc# cat group
root:x:0:
test:x:1003:
```

再创建相应的刚才未创建的部分文件, 并且拷贝一些缺少的文件

```
root@fengjinghang-ThinkPad-T470p:/var/chroot/etc# cd /var/chroot/usr
root@fengjinghang-ThinkPad-T470p:/var/chroot/usr# mkdir -p {bin,lib,libexec}
root@fengjinghang-ThinkPad-T470p:/var/chroot/usr# rm -rf bin
root@fengjinghang-ThinkPad-T470p:/var/chroot/usr# mkdir -p {bin,lib,libexec}
root@fengjinghang-ThinkPad-T470p:/var/chroot/usr# cp -a /usr/bin/ftp /var/chroot/usr/bin/
```

在 chroot 创建相应目录, 并且将 bash 中的内容拷贝过去

```
root@fengjinghang-ThinkPad-T470p:/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@fengjinghang-ThinkPad-T470p:/var/chroot/lib# ls
x86_64-linux-gnu
root@fengjinghang-ThinkPad-T470p:/var/chroot/lib# cd x86_64-linux-gnu
root@fengjinghang-ThinkPad-T470p:/var/chroot/lib/x86_64-linux-gnu# ls
libc.so.6 libdl.so.2 libtinfo.so.5
```

发现都是软连接, 并且飘红, 说明软连接缺失

```
root@fengjinghang-ThinkPad-T470p:/var/chroot/lib/x86_64-linux-gnu# ls -l
总用量 0
lrwxrwxrwx 1 root root 12 6月 5 2020 libc.so.6 -> libc-2.27.so
lrwxrwxrwx 1 root root 13 6月 5 2020 libdl.so.2 -> libdl-2.27.so
lrwxrwxrwx 1 root root 15 5月 23 2018 libtinfo.so.5 -> libtinfo.so.5.9
```

拷贝这些软连接

```
root@fengjinghang-ThinkPad-T470p:/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
```


这回 bash 全了

```
root@fengjinghang-ThinkPad-T470p:/var/chroot/lib/x86_64-linux-gnu# ls -l
总用量 2168
-rwxr-xr-x 1 root root 2030544 6月 5 2020 libc-2.27.so
lrwxrwxrwx 1 root root 12 6月 5 2020 libc.so.6 -> libc-2.27.so
-rw-r--r-- 1 root root 14560 6月 5 2020 libdl-2.27.so
lrwxrwxrwx 1 root root 13 6月 5 2020 libdl.so.2 -> libdl-2.27.so
lrwxrwxrwx 1 root root 15 5月 23 2018 libtinfo.so.5 -> libtinfo.so.5.9
-rw-r--r-- 1 root root 170784 5月 23 2018 libtinfo.so.5.9
root@fengjinghang-ThinkPad-T470p:/var/chroot/lib/x86_64-linux-gnu#
```

在 chroot 创建 lib64，拷贝相应文件

```
root@fengjinghang-ThinkPad-T470p:/var/chroot# mkdir lib64
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp -a /lib64/ld-linux-x86-64.so.2
/var/chroot/lib64
root@fengjinghang-ThinkPad-T470p:/var/chroot#
```

发现有软连接，拷贝相应软连接

```
root@fengjinghang-ThinkPad-T470p:/var/chroot/lib64# ls -l
总用量 0
lrwxrwxrwx 1 root root 32 6月 5 2020 ld-linux-x86-64.so.2 -> /lib/x86_64-linux-gnu/ld-2.27.so
root@fengjinghang-ThinkPad-T470p:/var/chroot/lib64# sudo cp -a /lib/x86_64-linux-gnu/ld-2.27.so /var/chroot/lib/x86_64-linux-gnu
root@fengjinghang-ThinkPad-T470p:/var/chroot/lib64#
```

竟然忘了拷贝 bash，拷贝一下

```
root@fengjinghang-ThinkPad-T470p:/var/chroot/lib64# cd ..
root@fengjinghang-ThinkPad-T470p:/var/chroot# cd bin
root@fengjinghang-ThinkPad-T470p:/var/chroot/bin# ls
root@fengjinghang-ThinkPad-T470p:/var/chroot/bin# sudo cp -a /bin/bash /var/chroot/bin/
root@fengjinghang-ThinkPad-T470p:/var/chroot/bin#
```

测试一下，bash 好用，顺便用它正式创建 chroot 沙箱

```
root@fengjinghang-ThinkPad-T470p:/var/chroot/bin# sudo cp -a /bin/bash /var/chroot/bin/
root@fengjinghang-ThinkPad-T470p:/var/chroot/bin# sudo chroot /var/chroot
bash-4.4# exit
exit
root@fengjinghang-ThinkPad-T470p:/var/chroot/bin#
```

2)利用 chroot 实现 SSH 服务或 FTP 服务的虚拟化隔离；

安装 pure-ftpd

```
root@fengjinghang-ThinkPad-T470p:/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
下列软件包是自动安装的并且现在不需要了:
```

创建 用户

```

root@fengjinghang-ThinkPad-T470p:/var/chroot# groupadd ftpgroup
root@fengjinghang-ThinkPad-T470p:/var/chroot# ls
bin dev etc home lib lib64 proc usr
root@fengjinghang-ThinkPad-T470p:/var/chroot# cd usr
root@fengjinghang-ThinkPad-T470p:/var/chroot/usr# ls
bin lib libexec lib.libexec
root@fengjinghang-ThinkPad-T470p:/var/chroot/usr# cd ..
root@fengjinghang-ThinkPad-T470p:/var/chroot# useradd -g ftpgroup -d /home/ftpuser -s /sbin/nologin ftpuser
root@fengjinghang-ThinkPad-T470p:/var/chroot# pure-pw useradd fengjinghang -u ftpuser -d /home/ftpuser/fengjinghang
Password:
Enter it again:
root@fengjinghang-ThinkPad-T470p:/var/chroot# pure-pw mkdb
root@fengjinghang-ThinkPad-T470p:/var/chroot#

```

试运行 ftp

```

root@fengjinghang-ThinkPad-T470p:/var/chroot# pure-pw mkdb
root@fengjinghang-ThinkPad-T470p:/var/chroot# service pure-ftpd start
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 18:16. 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.
Name (127.0.0.1:fengjinghang):

```

试运行：登陆

```

Name (127.0.0.1:fengjinghang): fengjinghang
331 User fengjinghang OK. Password required
Password:
230 OK. Current directory is /home/fengjinghang
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>

```

试运行：退出

```

ftp> exit
221-Goodbye. You uploaded 0 and downloaded 0 kbytes.
221 Logout.

```

查看 ftp 需要的链接

```

libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fcf7a6f3000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007fcf7a4d4000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007fcf7a2d0000)
libaudit.so.1 => /lib/x86_64-linux-gnu/libaudit.so.1 (0x00007fcf7a0a7000)
/lib64/ld-linux-x86-64.so.2 (0x00007fcf7bacc000)
libcap-ng.so.0 => /lib/x86_64-linux-gnu/libcap-ng.so.0 (0x00007fcf79ea2000)
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /lib/x86_64-linux-gnu/libssl.so.1.1 ./ --parent
cp: 无法获取'/lib/x86_64-linux-gnu/libssl.so.1.1' 的文件状态(stat): 没有那个文件或目录
root@fengjinghang-ThinkPad-T470p:/var/chroot# ls /lib/x86_64-linux-gnu/

```

把这些链接也拷贝过来


```

root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /lib/x86_64-linux-gnu/libssl.so.2.35 ./ --parent
cp: 无法获取'/lib/x86_64-linux-gnu/libssl.so.2.35' 的文件状态(stat): 没有那个文件或目录
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /usr/lib/x86_64-linux-gnu/libssl.so.1.1 ./ --parent
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /usr/lib/x86_64-linux-gnu/libcrypto.so.1.1 ./ --parent
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /lib/x86_64-linux-gnu/libcrypt.so.1 ./ --parent
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /lib/x86_64-linux-gnu/libcap.so.2 ./ --parent
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /lib/x86_64-linux-gnu/libpam.so.0 ./ --parent
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /lib/x86_64-linux-gnu/libc.so.6 ./ --parent
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /lib/x86_64-linux-gnu/libpthread.so.0 ./ --parent
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /lib/x86_64-linux-gnu/libdl.so.2 ./ --parent
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /lib/x86_64-linux-gnu/libaudit.so.1 ./ --parent
root@fengjinghang-ThinkPad-T470p:/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@fengjinghang-ThinkPad-T470p:/var/chroot# cp /lib/x86_64-linux-gnu/libcap-ng.so.0 ./ --parent

```

后续操作:

```

root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /etc/x86_64-linux-gnu/ctb
root@fengjinghang-ThinkPad-T470p:/var/chroot# ls
bin dev etc home lib lib64 proc usr
root@fengjinghang-ThinkPad-T470p:/var/chroot# mknod -m 666 dev/null c 1 3
root@fengjinghang-ThinkPad-T470p:/var/chroot# mknod dev/urandom c 1 9
root@fengjinghang-ThinkPad-T470p:/var/chroot# chroot ./

```

现在的问题: bash 这一步, 没有办法运行接下来的内容

重新进入沙箱

```

fengjinghang@fengjinghang-ThinkPad-T470p:~$ sudo chroot /var/chroot
[sudo] fengjinghang 的密码:

```

用沙箱的 bash 进行之前卡住的操作, 发现端口被占用

```

bash-4.4# pure-ftpd -j -l puredb:/etc/pure-ftpd/pureftpd.pdb
Unable to start a standalone server: Address already in use

```

查找占用该端口所在的进程, 把他 kill 掉 (有点暴力, 我在网上查的, 不知道好不好使), 然后

重新再 chroot 中启动 ftp

```

fengjinghang@fengjinghang-ThinkPad-T470p:~$ sudo fuser -n tcp 21
21/tcp: 7793
fengjinghang@fengjinghang-ThinkPad-T470p:~$ sudo kill 7793
fengjinghang@fengjinghang-ThinkPad-T470p:~$ sudo chroot /var/chroot
bash-4.4# pure-ftpd -j -l puredb:/etc/pure-ftpd/pureftpd.pdb

```

打开一个新的终端, 使用 ftp 服务

```

fengjinghang@fengjinghang-ThinkPad-T470p:~$ 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 10:49. 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:fengjinghang): fengjinghang
331 User fengjinghang OK. Password required
Password:
230 OK. Current directory is /

```

目录下, 创建一个 test 文件夹, 再 ls 一下, 发现只有一个 test 文件夹

```
230 OK. Current directory is /  
Remote system type is UNIX.  
Using binary mode to transfer files.  
ftp>
```

```
ftp> mkdir test  
257 "test" : The directory was successfully created  
ftp> ls  
200 PORT command successful  
150 Connecting to port 41701  
drwxr-xr-x  2 1004      1004      4096 Dec 12 10:50 test  
226-Options: -l
```

再猛往上一级文件夹退，发现还是只有 test 文件夹，没有其他文件夹，说明我们的沙箱应该创建成功了。

```
ftp> cd ../../../../  
250 OK. Current directory is /  
ftp> ls  
200 PORT command successful  
150 Connecting to port 44297  
drwxr-xr-x  2 1004      1004      4096 Dec 12 10:50 test  
226-Options: -l
```

之后由尝试创建了文件夹，在 test 文件夹内

```
ftp> cd /test  
250 OK. Current directory is /test  
ftp> mkdir testfengjinghang  
257 "testfengjinghang" : The directory was successfully created  
ftp>
```

通过打开文件管理，我们也可以发现创建成功

```
fengjinghang@fengjinghang-ThinkPad-T470p:/var/chroot/home/ftpuser/fengjinghang/t  
est$ ls  
testfengjinghang
```

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；

chroot 后会将 uid 设置为非 root，这里测试设置成 1000

运行 sudo ./code1 命令：

```
before:  
ruid:1000      euid:0   suid:0  
now give up root permission.....  
now:  
ruid:1000      euid:1000  suid:1000
```

```

root@fengjinghang-ThinkPad-T470p:/var/chroot/bin# ls -l /var/chroot/lib/x86_64-l
inux-gnu
总用量 3352
-rwxr-xr-x 1 root root 170960 6月 5 2020 ld-2.27.so
lrwxrwxrwx 1 root root 32 6月 5 2020 ld-linux-x86-64.so.2 -> /lib/x86_64
-linux-gnu/ld-2.27.so
-rw-r--r-- 1 root root 124848 12月 12 18:41 libaudit.so.1
-rwxr-xr-x 1 root root 2030544 12月 12 18:40 libc-2.27.so
-rw-r--r-- 1 root root 18712 12月 12 18:42 libcap-ng.so.0
-rw-r--r-- 1 root root 22768 12月 12 18:39 libcap.so.2
-rw-r--r-- 1 root root 39208 12月 12 18:39 libcrypt.so.1
lrwxrwxrwx 1 root root 12 6月 5 2020 libc.so.6 -> libc-2.27.so
-rw-r--r-- 1 root root 14560 12月 12 18:40 libdl-2.27.so
lrwxrwxrwx 1 root root 13 6月 5 2020 libdl.so.2 -> libdl-2.27.so
-rw-r--r-- 1 root root 55848 12月 12 18:39 libpam.so.0
lrwxrwxrwx 1 root root 17 2月 4 2018 libpcre.so.3 -> libpcre.so.3.13.3
-rw-r--r-- 1 root root 464824 2月 4 2018 libpcre.so.3.13.3
-rwxr-xr-x 1 root root 144976 6月 5 2020 libpthread-2.27.so
lrwxrwxrwx 1 root root 18 6月 5 2020 libpthread.so.0 -> libpthread-2.27.
so
-rw-r--r-- 1 root root 154832 3月 1 2018 libselinux.so.1
lrwxrwxrwx 1 root root 15 5月 23 2018 libtinfo.so.5 -> libtinfo.so.5.9
-rw-r--r-- 1 root root 170784 5月 23 2018 libtinfo.so.5.9
root@fengjinghang-ThinkPad-T470p:/var/chroot/bin#

```

代码如下：

```

#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:\n ruid:%d\t euid:%d\t suid:%d\n", ruid, euid, suid);
    printf("now chdir\n");
    chdir("/var/chroot\n");
    if(chroot("/var/chroot")==0){
        printf("chroot succeed!\n");
    }

    getresuid(&ruid, &euid, &suid);
    printf("before:\n ruid:%d\t euid:%d\t suid:%d\n", ruid, euid, suid);
    printf("now give up root permission....\n");
    setresuid(1000, 1000, 1000); //test的uid
    getresuid(&ruid, &euid, &suid);
    printf("now: \nruid:%d\teuid:%d\tsuid:%d\n", ruid, euid, suid);
    execlp("ls", "ls", NULL);
    return 0;
}

```



```
root@fengjinghang-ThinkPad-T470p:/var/chroot# cp /lib/x86_64-linux-gnu/libcap-ng
.so.0 ./ --parent
root@fengjinghang-ThinkPad-T470p:/var/chroot# ls
bin dev etc home lib lib64 proc usr
root@fengjinghang-ThinkPad-T470p:/var/chroot# mknod -m 666 dev/null c 1 3
root@fengjinghang-ThinkPad-T470p:/var/chroot# mknod dev/urandom c 1 9
root@fengjinghang-ThinkPad-T470p:/var/chroot# chroot ./
bash-4.4#
bash-4.4# exit
exit
root@fengjinghang-ThinkPad-T470p:/var/chroot# exit
exit
fengjinghang@fengjinghang-ThinkPad-T470p:~$ sudo chroot /var/chroot
[sudo] fengjinghang 的密码:
bash-4.4# pure-ftpd -j -l puredb:/etc/pure-ftpd/pureftpd.pdb
Unable to start a standalone server: Address already in use
bash-4.4# exit
exit
fengjinghang@fengjinghang-ThinkPad-T470p:~$ sudo fuser -n tcp 21
21/tcp: 7793
fengjinghang@fengjinghang-ThinkPad-T470p:~$ sudo kill 7793
fengjinghang@fengjinghang-ThinkPad-T470p:~$ sudo chroot /var/chroot
bash-4.4# pure-ftpd -j -l puredb:/etc/pure-ftpd/pureftpd.pdb
```

```
fengjinghang@fengjinghang-ThinkPad-T470p: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful
150 Connecting to port 36817
226-Options: -l
226 0 matches total
ftp> mkdir test
257 "test" : The directory was successfully created
ftp> ls
200 PORT command successful
150 Connecting to port 41701
drwxr-xr-x  2 1004      1004              4096 Dec 12 10:50 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 44297
drwxr-xr-x  2 1004      1004              4096 Dec 12 10:50 test
226-Options: -l
226 1 matches total
ftp> 
```



```

fengjinghang@fengjinghang-ThinkPad-T470p:/home$ ls
fengjinghang  lost+found
fengjinghang@fengjinghang-ThinkPad-T470p:/home$ cd ..
fengjinghang@fengjinghang-ThinkPad-T470p:/ $ ls
bin      etc          lib          lost+found  proc      snap      usr
boot     home         lib32        media       root      srv       var
cdrom    initrd.img   lib64        mnt         run       sys       vmlinuz
dev       initrd.img.old  libx32      opt         sbin      tmp       vmlinuz.old
fengjinghang@fengjinghang-ThinkPad-T470p:/ $ cd var
fengjinghang@fengjinghang-ThinkPad-T470p:/var$ ls
backups  crash      local      mail      run      tmp
cache    cuda-repo-10-1-local-10.1.105-418.39  lock      metrics  snap
chroot   lib        log        opt       spool
fengjinghang@fengjinghang-ThinkPad-T470p:/var$ cd /chroot
bash: cd: /chroot: 没有那个文件或目录
fengjinghang@fengjinghang-ThinkPad-T470p:/var$ cd chroot/home/ftpuser/fengjinghang/test/
fengjinghang@fengjinghang-ThinkPad-T470p:/var/chroot/home/ftpuser/fengjinghang/test$ ls
fengjinghang@fengjinghang-ThinkPad-T470p:/var/chroot/home/ftpuser/fengjinghang/test$ ls
testfengjinghang
fengjinghang@fengjinghang-ThinkPad-T470p:/var/chroot/home/ftpuser/fengjinghang/test$ 

```

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：



```

打开(O)  chr_nc.c  保存(S)
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

/*本程序前提设定为：将该文件编译产生的文件设置为root用户所有，并且设置setuid位*/

int main(){
    uid_t ruid, euid, suid;
    getresuid(&ruid, &euid, &suid);
    printf("最开始:\n ruid:%d\t euid:%d\t suid:%d\n", ruid, euid, suid);
    printf("执行 chdir\n");
    //chdir("/var/chroot");
    if(chroot("/var/chroot")==0){
        printf("chroot succeed!\n");
    }
    getresuid(&ruid, &euid, &suid);
    printf("现在:\n ruid:%d\t euid:%d\t suid:%d\n", ruid, euid, suid);
    printf("give up root permission.....\n");
    setresuid(ruid, ruid, ruid); //本程序的用户uid
    getresuid(&ruid, &euid, &suid);
    printf("now: \nruid:%d\teuid:%d\tsuid:%d\n", ruid, euid, suid);
    execlp("ls", "ls", "-l", NULL);
    return 0;
}

```

编译一下，更改权限

```

fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/022$ sudo chmod 4755 chr
chmod: 无法访问'chr': 没有那个文件或目录
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/022$ sudo gcc chr.c -o chr
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/022$ sudo chmod 4755 chr
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/022$ ./chr
before:
  ruid:1000      euid:0   suid:0
now chdir
chroot succeed!
before:
  ruid:1000      euid:0   suid:0
now give up root permission.....
now:
ruid:1000      euid:1000   suid:1000
total 32
drwxr-xr-x 2 0 0 4096 Dec 12 10:58 bin
drwxr-xr-x 2 0 0 4096 Dec 12 10:43 dev
drwxr-xr-x 3 0 0 4096 Dec 12 10:19 etc
drwxr-xr-x 4 0 0 4096 Dec 12 10:49 home
drwxr-xr-x 3 0 0 4096 Dec 12 09:59 lib
drwxr-xr-x 2 0 0 4096 Dec 12 10:06 lib64
drwxr-xr-x 2 0 0 4096 Dec 12 09:49 proc
drwxr-xr-x 8 0 0 4096 Dec 12 10:19 usr
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/022$ █

```

以普通身份跑一下，我们发现，他越狱了（ls 命令，出来了系统根目录内容）

```

-rwsr-xr-x 1 0 0 8576 Dec 12 11:30 chr2
-rw-r--r-- 1 1000 1000 777 Dec 12 11:29 chr2.c
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/022$ ./chr2
before:
  ruid:1000      euid:0   suid:0
now chdir
chroot succeed!
before:
  ruid:1000      euid:0   suid:0
now give up root permission.....
now:
ruid:1000      euid:1000   suid:1000
total 20
-rwxrwxrwx 1 1000 1000 830 Dec 12 11:28 chr.c
-rwsr-xr-x 1 0 0 8576 Dec 12 11:30 chr2
-rw-r--r-- 1 1000 1000 777 Dec 12 11:29 chr2.c
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/022$ sudo gcc chr.c -o chr
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/022$ ls -l
总用量 32
-rwxr-xr-x 1 root      root      8616 12月 12 19:31 chr
-rwsr-xr-x 1 root      root      8576 12月 12 19:30 chr2
-rw-r--r-- 1 fengjinghang fengjinghang 777 12月 12 19:29 chr2.c
-rwxrwxrwx 1 fengjinghang fengjinghang 830 12月 12 19:28 chr.c
fengjinghang@fengjinghang-ThinkPad-T470p:~/lab01/022$ █

```

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。

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。

通过 chroot 沙箱的配置，熟悉了 chroot 的运作过程，了解了沙箱的安全机制，同时通过程序攻击的方式，熟悉了如何防止沙箱被突破的漏洞，这对以后的安全编程有很大的指导意义。