计算机系统安全实验报告

实验名称：passwd 实现细粒度访问控制及 root 能力安全使用

班级：1703101

学号：1170300520

姓名：郭子阳

计算机学院

一 分析 passwd 程序实现过程，模拟系统中密码修改机制，在自主访问控制系统中实现细粒度的权限管理。

类似/etc/shadow 文件 aaa 格式设置为，用户名+空格+密码，如下：

```
[ziyang@ziyang-pc CSS_Lab2]$ cat aaa
root 12345678
ziyang 7654321
exp 123
[ziyang@ziyang-pc CSS_Lab2]$
```

编写 passwd.c 文件模拟/usr/bin/passwd 的功能，功能描述如下：

- 所有用户都可使用 passwd+密码修改自己的密码
- root 用户可使用 passwd+用户名+密码修改任意用户的密码

编译后的 passwd 程序的文件所有者是 root，并且设置了 setuid 位，使任何用户都可以以 root 身份执行，aaa 文件设置了只允许 root 读写，不允许其他用户读写。

passwd 程序首先要获取进程的 ruid，以判断执行的用户，由于获取的仅仅只是用户的 id，而不是用户的用户名，于是需要调用 getpwuid()函数获取对应的用户名：

```
uid_t ruid, euid, suid;
struct passwd* userStruct;

getresuid(&ruid, &euid, &suid);
userStruct = getpwuid(ruid);
```

并根据 argc，即参数的个数来判断，如果参数只有一个（argc == 2），说明只是修改该用户自己的密码，则无需进行权限判断；如果参数有两个（argc == 3），说明调用者想修改其他用户的密码，此时需要判断调用用户是否是 root，如果是 root 才允许，否则设置 errno 位，输出错误信息：

```
switch(argc)
{
    case 2:
        changePassword(userStruct->pw_name, argv[1]);
        break;
    case 3:
        if(strcmp("root", userStruct->pw_name) == 0)
        {
            changePassword(argv[1], argv[2]);
        } else
        {
            errno = EPERM;
            perror("passwd");
        }
        break;
    default:
        errno = EINVAL;
        perror("passwd");
}
```

修改密码用的函数 changePassword()函数，接受两个参数：username 和 password。首先使用 getline 函数按行读取 aaa 文件，按照空格拆分每一行，将空格前的内容与 username 相比较，如果相同，则说明需要修改该行：

```
ssize_t bytesNum;
size_t n = 0;
char* line;
long line_start = ftell(fp);
while((bytesNum = getline(&line, &n, fp)) != -1)
{
    if(strlen(line) == 0)
    {
        continue;
    }
    char* currentLineUser = strsep(&line, " ");
    if(strcmp(username, currentLineUser) == 0)
    {
```

发现密码所在行后，就从下一行开始一直读到文件尾，将读到的内容保存在一个字符串中，用于修改后再次写入。

接着将文件指针移到密码所在行开头，按照 username+空格+password 的格式写入后加换行，接着将刚刚保存的字符串接着写入到文件，写完后，使用 ftruncate()函数删除后续的内容，防止修改前比修改后长而出现修改不完全的情况。

```c
if(strcmp(username, currentLineUser) == 0)
{
    char after[1024] = {0};
    int c = 0;
    while(!feof(fp))
    {
        after[c] = fgetc(fp);
        c ++;
    }
    if(c != 0)
    {
        after[c - 1] = '\0';
    }
    fseek(fp, line_start, SEEK_SET);
    fprintf(fp, "%s %s\n", username, password);
    if(c != 0)
    {
        fprintf(fp, "%s", after);
    }
    long total_length = ftell(fp);
    int fd = fileno(fp);
    if(ftruncate(fd, total_length))
    {
        perror("passwd");
    }
    fclose(fp);
    return;
}
line_start = ftell(fp);
```

修改密码测试：

初始密码文件如下：

```
[ziyang@ziyang-pc CSS_Lab2]$ cat aaa
root 12345678
ziyang 7654321
exp 123
[ziyang@ziyang-pc CSS_Lab2]$
```

以普通用户身份（ziyang）修改自己的密码：

```
[ziyang@ziyang-pc CSS_Lab2]$ ./passwd 666666
The user name of the ruid is ziyang
change ziyang's password to 666666
[ziyang@ziyang-pc CSS_Lab2]$ sudo cat aaa
root 12345678
ziyang 666666
exp 123
[ziyang@ziyang-pc CSS_Lab2]$
```

试图修改 exp 的密码，会提示 Operation not permitted，无法修改：

```
[ziyang@ziyang-pc CSS_Lab2]$ ./passwd exp 888888
The user name of the ruid is ziyang
passwd: Operation not permitted
[ziyang@ziyang-pc CSS_Lab2]$ sudo cat aaa
root 12345678
ziyang 666666
exp 123
[ziyang@ziyang-pc CSS_Lab2]$
```

当以 root 用户执行时，修改 exp 的密码为 888888:

```
[ziyang@ziyang-pc CSS_Lab2]$ sudo ./passwd exp 888888
The user name of the ruid is root
change exp's password to 888888
[ziyang@ziyang-pc CSS_Lab2]$ sudo cat aaa
root 12345678
ziyang 666666
exp 888888
[ziyang@ziyang-pc CSS_Lab2]$
```

可以看到被成功修改。

二 root 的 capability 位使用

1. cap_chown 能力位

该能力允许用户任意修改文件的拥有者。

首先创建一个文件，其所属用户为 ziyang：

```
[ziyang@ziyang-pc ~]$ touch test.txt
[ziyang@ziyang-pc ~]$ ls -l test.txt
-rw-r--r-- 1 ziyang ziyang 0 12月  8 18:08 test.txt
[ziyang@ziyang-pc ~]$
```

修改能力位之前，试图文件拥有者设置为 root：

```
[ziyang@ziyang-pc ~]$ chown root test.txt
chown: 正在更改'test.txt' 的所有者：不允许的操作
[ziyang@ziyang-pc ~]$
```

执行 setcap 命令后，再尝试修改所有者：

```
[ziyang@ziyang-pc ~]$ sudo setcap cap_chown=eip /bin/chown
[ziyang@ziyang-pc ~]$ chown root:root test.txt
[ziyang@ziyang-pc ~]$ ls -l test.txt
-rw-r--r-- 1 root root 0 12月  8 18:08 test.txt
[ziyang@ziyang-pc ~]$
```

即可成功修改。

2. cap_kill 能力位

该能力允许用户结束其他用户执行的进行。

切换到用户 exp，并在后台执行一个进程，当切换到其他用户（ziyang）时，可以使用 ps 看到这个进程，但是无法结束它：

```
[exp@ziyang-pc ~]$ nohup ./deamon > /dev/null 2>&1 &
[1] 17306
[exp@ziyang-pc ~]$ su ziyang
密码：
[ziyang@ziyang-pc exp]$ ps -A | grep deamon
17306 pts/0    00:00:00 deamon
[ziyang@ziyang-pc exp]$ kill -9 17306
bash: kill: (17306) - 不允许的操作
[ziyang@ziyang-pc exp]$
```

当给 bash 程序赋予 cap_kill 位后（kill 信号由 bash 发出），ziyang 即可 kill 掉其

他用户的进程：

```
[ziyang@ziyang-pc exp]$ sudo setcap cap_kill=eip /usr/bin/bash
[ziyang@ziyang-pc exp]$ ps -A | grep deamon
17306 pts/0    00:00:00 deamon
[ziyang@ziyang-pc exp]$ kill -9 17306
[ziyang@ziyang-pc exp]$ ps -A | grep deamon
[ziyang@ziyang-pc exp]$
```

3. cap_dac_override 能力位
该能力允许用户无视文件能力位设置读写执行文件。
首先尝试使用 cat 命令读/etc/shadow 文件：

```
[ziyang@ziyang-pc ~]$ cat /etc/shadow
cat: /etc/shadow: 权限不够
[ziyang@ziyang-pc ~]$
```

给/usr/bin/cat 设置了 cap_dac_override 能力位后，即可无视权限读取文件：

```
[ziyang@ziyang-pc ~]$ sudo setcap cap_dac_override=eip /usr/bin/cat
[ziyang@ziyang-pc ~]$ cat /etc/shadow
root:$6$qq.fR1GH2HFy8QKX$kbugzgK.ZGA3DA/KrDCjuDbHg1Z.Z7UPqmIMXA99XJqQoAyv2UiLmjk
ugX9GNMT2zNRJZszCcVE.cWCilaj4s0:18233::::::
nobody:!!:18214::::::
dbus:!!:18214::::::
bin:!!:18214::::::
daemon:!!:18214::::::
mail:!!:18214::::::
ftp:!!:18214::::::
http:!!:18214::::::
systemd-journal-remote:!!:18214::::::
systemd-network:!!:18214::::::
systemd-resolve:!!:18214::::::
systemd-timesync:!!:18214::::::
systemd-coredump:!!:18214::::::
```

4. 系统启动时关闭某能力位，对系统的应用和安全性有何影响，以具体能力位为例说明。
例如关闭 cap_chown 位，则任何用户都可以随意改变任意文件的所属用户，相当于可以访问或者执行任何文件。
若关闭 cap_kill 位，任何用户可以关闭任何进程，可能关闭系统的重要进程，造成系统不稳定。
若关闭 cap_dac_override 位，则任何用户可以无视权限地读写执行任何文件。

5. 组合系统的部分能力位，实现系统的网络管理功能，或用户管理功能、文件管理功能。
编写 capset.c，以实现绑定端口的权限的分配。
绑定较低的端口需要的 CAP 权限有 cap_net_bind_service, cap_net_broadcast, cap_net_admin, cap_net_raw，共四个。
capset.c 编译后的 setcap 需要设置 setuid 位，是因为分配 CAP 权限需要 root 权限。
capset 运行结果如下：

```
[ziyang@ziyang-pc CSS_Lab2]$ ./capset
ruid=1000
euid=0
suid=0
Bind port 80 successfully!
The process was given capabilities = cap_net_bind_service,cap_net_broadcast,cap_
net_admin,cap_net_raw+eip
Bind port 81 successfully!
dropping caps
The process was given capabilities =
Error bind port 82
[ziyang@ziyang-pc CSS_Lab2]$
```

capset 程序首先会试图绑定 80 端口，结果是成功绑定，原因是程序设置了 setuid 位，以 root 的身份运行。之后将上述提到的四个权限分配给进程：

```
cap_t caps = cap_init();
cap_value_t capList[4] = {CAP_NET_BIND_SERVICE,
CAP_NET_BROADCAST,CAP_NET_ADMIN, CAP_NET_RAW};

cap_set_flag(caps, CAP_EFFECTIVE, 4, capList, CAP_SET);
cap_set_flag(caps, CAP_INHERITABLE, 4, capList, CAP_SET);
cap_set_flag(caps, CAP_PERMITTED, 4, capList, CAP_SET);
```

```
if(cap_set_proc(caps) != 0)
{
    perror("capset");
    return -1;
}
```

随后尝试绑定 81 端口，成功绑定。
最后清除 caps 数组并将 caps 设置给进程 CAP，相当于剥夺上述四个权限：

```
cap_clear(caps);
if(cap_set_proc(caps) != 0)
{
    perror("capset");
    return -1;
}
listCaps();
```

所以最后再尝试绑定端口 82 时就会失败。

6. 编制攻击程序，测试能力位的安全性。
编制程序 tryChangeTime.c，陷入死循环持续尝试修改当前时间：

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

int main()
{
    pid_t pid;
    while(1)
    {
        sleep(1);

        if((pid = fork()) == 0)
        {
            execlp("date", "date", "-s", "2020-6-9", (int *)0);
            return 0;
        } else
        {
            int stat_val;
            waitpid(pid, &stat_val, 0);
        }
    }
}

```

一旦/usr/bin/date 被设置为 cap_sys_time=eip，那么攻击程序立刻就可以成功修改系统时间，达到了攻击的目的。