计算机系统安全实验

实验二

姓名：卢兑玳

学号：L1703009**01**

实验二　passwd 实现细粒度访问控制及 root 能力位安全应用

2.1 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。（5 分）

说明：（1）配合第 3 章 在基于用户权限管理基础上，进行细粒度的权限管理。（2）1 学时，每人独立完成

1、passwd 程序功能描述

在 Linux 中，passwd 程序是可信任的，修改存储经过加密的密码的影子密码文件（/etc/shadow），passwd 程序执行它自己内部的安全策略，允许普通用户修改属于他们自己的密码，同时允许 root 修改所有密码。为了执行这个受信任的作业，passwd 程序需要有移动和重新创建 shadow 文件的能力，在标准 Linux 中，它有这个特权，因为 passwd 程序可执行文件在执行时被加上了 setuid 位，它作为 root 用户（它能访问所有文件）允许，然而，许多程序都可以作为 root 允许（实际上，所有程序都有可能作为 root 允许）。这就意味着任何程序（当以 root 身份运行时）都有可能能够修改 shadow 文件。

2、实验要求

自己编制文件和程序，仿制 passwd 程序修改/etc/shadow 的功能，包括：

a) 自己设置一个类/etc/shadow 文件 aaa，该文件中约定好内容格式，和读取该文件的程序相配合，文件中包括超级用户及其内容、普通用户及其内容

b)编制程序使得：Root 用户能够读取和修改 aaa 文件中所有用户的内容普通用户仅能够读取和修改 aaa 文件中属于自己用户的内容

c）普通用户能以 root 身份执行所编制的类 passwd 程序

3、编制实验报告，回答上面问题，给出源代码、分析过程和实验结果

实验过程：

（a）首先创建一个名称为 aaa 的文本文件，里面的格式约定为：“用户名:密码”

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ gedit aaa
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ cat aaa
root:root
fengjinghang:1234
test:1235
```

（b）编写程序使得：root 用户能够读取和修改 aaa 文件中所有用户的内容，普通用户仅能够读取和修改 aaa 文件中属于自己用户的内容。

普通用户可以读取和修改自己的内容

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ gedit aaa
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ cat aaa
root:root
fengjinghang:1234
test:1235
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ ./mypasswd nohtaeyun
当前的用户为: fengjinghang
将用户fengjinghang 的内容改为 nohtaeyun
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ cat aaa
root:root
fengjinghang:nohtaeyun
test:1235
```

普通用户不可以修改其他用户的密码

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ cat aaa
root:root
fengjinghang:nohtaeyun
test:1235
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ su root
密码:
root@fengjinghang-ThinkPad-T470p:/home/fengjinghang/oss-lab2# ./mypasswd root 0011
当前的用户为: root
将用户root 的内容改为 0011
root@fengjinghang-ThinkPad-T470p:/home/fengjinghang/oss-lab2# ./mypasswd fengjinghang 0011
当前的用户为: root
将用户fengjinghang 的内容改为 0011
root@fengjinghang-ThinkPad-T470p:/home/fengjinghang/oss-lab2# ./mypasswd test 0011
当前的用户为: root
将用户test 的内容改为 0011
root@fengjinghang-ThinkPad-T470p:/home/fengjinghang/oss-lab2# cat aaa
root:0011
fengjinghang:0011
test:0011
```

（c）普通用户能以 root 身份执行所编制的类 passwd 程序

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ cat aaa
root:0011
fengjinghang:0011
test:0011
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ sudo ./mypasswd test 11
10
[sudo] fengjinghang 的密码:
当前的用户为: root
将用户test 的内容改为 1110
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ cat aaa
root:0011
fengjinghang:0011
test:1110
```

编写的程序如下：

```c
#define _GNU_SOURCE
#include <stdio.h>
#include <string.h>
#include <pwd.h>
#include <unistd.h>
#include <sys/types.h>


//改变文件中 某用户的内容
void change(char *user, char *context);

int main(int argc, char **argv){
    uid_t ruid, euid, suid;
    struct passwd *user;
    getresuid(&ruid, &euid, &suid);
    user = getpwuid(ruid);
    printf("当前的用户为：%s\n", user->pw_name);

    switch(argc){
        case 2:
            change(user->pw_name, argv[1]);
            break;
        case 3:
            if(strcmp(user->pw_name, "root") == 0){
                change(argv[1], argv[2]);
            }
            break;
        break;
    }
    return 0;
}
```

```c
//改变文件中 某用户的内容
void change(char *user, char *context){
    FILE *fp;
    char *line = NULL;
    size_t len = 0;
    ssize_t read_line;  //ssize_t 是signed_size_t
    long offset;
    char *p = NULL;
    int same;
    char buf_after[1000] = {0};
    char buf_before[100][100];


    fp = fopen("aaa", "r+");  //打开可读写的文件，该文件必须存在
    offset = ftell(fp);  //得到文件位置指针，当前位置相对于文件收不的偏移量（字节）
    int i = 0;

    //read file
    while((read_line = getline(&line, &len, fp)) != -1){
        strcpy(buf_before[i], line);

        p = strstr(line, ":" );  //判断字符串str2是否是str1的子串，如果是，则返回str2在str1中首次出现的地址，否则，返回NULL
        if(p == NULL){
            continue;
        }
```

2.2 利用 root 的能力机制实现系统加固，有效实现 root 能力的分发和管理。提供程序比较进行 root 能力管理前后系统安全性的差异。

说明：（1）配合第 4 章 实现 root 的多种能力的有效管理，提高 root 用户权利的合理分发，测试 root 能力管理的安全性和有效性。（2）3 学时，2 人一组合作完成。

1、学习和理解 root 的 capability 能力位功能。修改系统内核，配置 capability 的能力位，实现几种能力位的设置可验证。以 redhat 2.4 下的能力为例实现能力位的配置实现。

1）函数说明

getcap 可以获得程序文件所具有的能力(CAP).

getpcaps 可以获得进程所具有的能力(CAP).

setcap 可以设置程序文件的能力(CAP).

注:

1)cap_chown=eip 是将 chown 的能力以 cap_effective(e),cap_inheritable(i),cap_permitted(p)三种位图的方式授权给相关的程序文件.

2)如果改变文件名,则能力保留到新文件.

3)用 setcap -r /bin/chown 可以删除掉文件的能力.

4)重新用 setcap 授权将覆盖之前的能力.

能力位：CAP_SYS_NICE 23(允许提升优先级,设置其它进程的优先级)

{

对于普通用户程序的 NICE 优先级,不能超过 ulimit 对它的限制,如下:

nice -n -5 ls

nice: cannot set niceness: Permission denied

而 CAP_SYS_NICE 可以帮助普通用户设置一个想要的一个任意优先级.

setcap cap_sys_nice=eip /usr/bin/nice

切换到普通用户,指定优先级,如下:

nice -n -5 ls

log mnt mount.c mounttest pacct psacct psacct.c reboot1 reboot1.c test

[root@localhost zy]# setcap cap_sys_nice=eip /home/tttt/test

[root@localhost zy]# getcap /home/tttt/test

/home/tttt/test = cap_sys_nice+eip

}

2、实验要求：

（1）实现 3 种基本能力位的授权和查看，并分析授权前和授权后的差异；

（2）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，比如 cap_sys_module，cap_linux_immutable

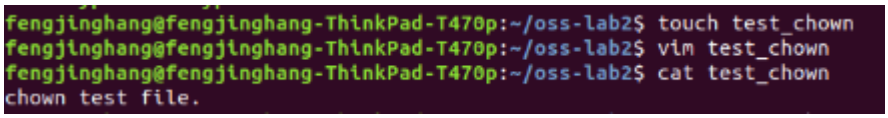
（3）组合系统的部分能力位，实现系统的网络管理功能，或用户管理功能、文件管理功能。

（4）编制攻击程序，测试能力位的安全性。

（1）实现 3 种基本能力位的授权和查看，并分析授权前和授权后的差异；

① cap_chown 能力：允许改变文件的所有权。

新建一个 test_chown 文件，用于测试



查看其用户和组的情

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ ls -l test_chown
-rw-r--r-- 1 fengjinghang fengjinghang 17 12月 17 21:09 test_chown
```

直接更改其所属组和用户为 root，发现是不被允许的操作

```
2$ chown root:root test_chown chown: 正在更改'test_chown' 的所有者: 不允许的操作
```

我们为设置/bin/chown 能力位，并且再次使用 chown 来修改其所属组合用户为 root，发现可以修改

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ sudo setcap cap_chown=eip /bin/chown
[sudo] fengjinghang 的密码:
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ getcap /bin/chown
/bin/chown = cap_chown+eip
```

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ chown root:root test_chown
```

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ ls -l test_chown
```

```
-rw-r--r-- 1 root root 17 12月 17 21:09 test_chown
```

我们为设置/bin/chown 能力位，并且再次使用 chown 来修改其所属组合用户为 root，发现可以修改

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ chown root:root test_chown fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ ls -l test_chown
-rw-r--r-- 1 root root 17 12月 17 21:09 test_chown
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ sudo setcap -r /bin/chown
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ getcap /bin/chown
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ chown root:root test_chown chown: 正在更改'test_chown' 的所有者: 不允许的操作
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$
```

当我们再次取消掉/bin/chown 能力位后，再次使用 chown 来修改其所属的组和用户，发现是不被允许的

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ date
2020年 12月 17日 星期四 21:15:48 CST
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ date -s 2020-12-16
date: 无法设置日期: 不允许的操作
2020年 12月 16日 星期三 00:00:00 CST
```

② cap_sys_time 能力：允许改变系统时钟

查看系统时间，并且尝试直接修改，发现是不允许的

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ sudo setcap cap_sys_time=ei
p /bin/date
[sudo] fengjinghang 的密码:
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ date -s 2020-12-16
2020年 12月 16日 星期三 00:00:00 CST
```

我们设置/bin/date 能力位，再次尝试使用 date 修改时间，修改成功

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ getcap /bin/date
/bin/date = cap_sys_time+eip
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ sudo setcap -r /bin/date
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ getcap /bin/date
```

实验过后，我们取消掉能力位

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ getcap /bin/date
/bin/date = cap_sys_time+eip
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ sudo setcap -r /bin/date
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ getcap /bin/date
```

② cap_dac_read_search 能力：忽略所有对读、搜索操作的限制

我们以/etc/shadow 为例，先用 root 权限查看其内容

```
dnsmasq:*:17954:0:99999:7:::
colord:*:17954:0:99999:7:::
speech-dispatcher:!:17954:0:99999:7:::
hplip:*:17954:0:99999:7:::
kernoops:*:17954:0:99999:7:::
pulse:*:17954:0:99999:7:::
rtkit:*:17954:0:99999:7:::
saned:*:17954:0:99999:7:::
usbmux:*:17954:0:99999:7:::
fengjinghang:$6$Ot.q9sBs$gLu2al4dzPIQclAcTc5CBqMl2PR2DL5WIhUJMKGlNIAOq5CnO.GjDYj
atfmRKlDxkZvk8mDc4JY6UA1j7ml3P.:18608:0:99999:7:::
cups-pk-helper:*:18117:0:99999:7:::
geoclue:*:18117:0:99999:7:::
gdm:*:18117:0:99999:7:::
gnome-initial-setup:*:18117:0:99999:7:::
nvidia-persistenced:*:18334:0:99999:7:::
openvpn:!:18492:0:99999:7:::
openvpn_as:!:18492:0:99999:7:::
test:$6$E/IHnY0m$k3ZeVe03DtZ/E3Q1dNwTTb.v4VW7zalpy5oOAeNIRvJG1M/QvCpLpvvM7KZlnjc
QjfmmyHXTlhuyfK6y9tQ0a.:18608:0:99999:7:::
ftpuser:!:18608:0:99999:7:::
```

我们以普通的权限再次查看，发现不允许访问

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ cat /etc/shadow
cat: /etc/shadow: 权限不够
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$
```

我们设置/bin/cat 能力位，再次尝试查看/etc/shadow，发现可以查看

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ sudo setcap cap_dac_read_search=eip /bin/cat
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ getcap /bin/cat
/bin/cat = cap_dac_read_search+eip
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ cat /etc/shadow
root:$6$lq0h8YBZ$HND8Iih9.l34Bb4tBvERqcjUghVwTNpH.HGR.mmabGqcFjR6UeDrCTqjq/hdX/I9Yhthn4hDwg2vXKnH2CRCu1:18608:0:99999:7:::
daemon:*:17953:0:99999:7:::
bin:*:17953:0:99999:7:::
sys:*:17953:0:99999:7:::
sync:*:17953:0:99999:7:::
```

取消其能力位，就不能查看了

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ sudo setcap -r /bin/cat
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ getcap /bin/cat
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ cat /etc/shadow
cat: /etc/shadow: 权限不够
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$
```

（2）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，比如 cap_sys_module，cap_linux_immutable

系统启动时删除部分能力，可以保护系统。系统管理员通常为了系统的安全，完全可以剥夺 root 用户的能力，这样即使是 root 用户，也将无法进行某些操作，并且 root 也不能立即恢复被删除的能力，只有 init 进程能够添加能力，这从一定程度上提高了系统的安全性。
通常，一个能力如果从能力边界集中被删除，只有系统重新启动才能恢复。因此 root 用户可以删除系统保留的能力，但这个过程不可逆。

（1）cap_linux_immutable

允许修改文件的不可修改(IMMUTABLE)和只添加(APPEND-ONLY)属性。普通用户不能通过 chattr 对文件设置 IMMUTABLE 和 APPEND-ONLY 权限,而通过 CAP_LINUX_IMMUTABLE 可以使普通用户通过自己增减(immutable/append-only)权限。
系统启动时没有 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）cap_sys_module

如果拥有此能力位，表示用户能够加载(或卸载)内核模块的特权操作。
系统启动时没有 CAP_SYS_MODULE 能力，攻击者不能修改系统的内核。系统内核被改动，需要重新启动系统才能使用新内核。

（3）cap_chown

允许改变文件的所有权。
系统启动时没有 cap_chown 能力位，攻击者不能修改文件属主，可以避免其访问或执行一些重要文件。

（4）cap_sys_time

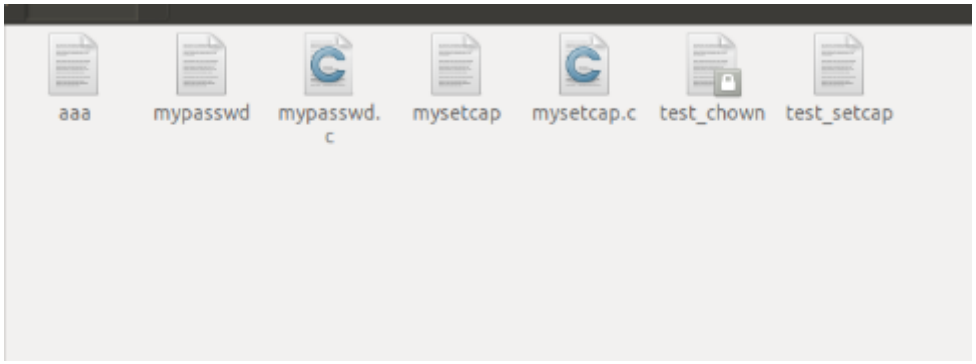
允许改变系统时钟。
系统启动时没有 cap_sys_time 能力，攻击者不能修改系统的时间，防止造成系统的时间错乱。

（5）cap_dac_read_search

忽略所有对读、搜索操作的限制。
系统启动时没有 cap_dac_read_search 能力，防止攻击者对重要的文件的读取。

（3）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。

创建 test_setcap 文件，并且赋予其可执行权限




```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ ls -l test_setcap
-rwxr--r-- 1 fengjinghang fengjinghang 16 12月 17 21:28 test_setcap
```

创建 mysetcap 文件，编写相应代码，想要实现的功能：更改 test_setcap 的 setuid 位，更改其所属的用户和组

分析，我们需要的相应的能力位有：

CAP_CHOWN　允许改变文件的所有权
CAP_FOWNER　如果文件属于进程的 UID，就取消对文件的限制
CAP_FSETID　允许设置 setuid 位
CAP_SETGID　允许改变组 ID
CAP_SETUID　允许改变用户 ID

代码实现如下

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

#undef _POSIX_SOURCE
#include <sys/capability.h>

void list_cap(){
    cap_user_header_t cap_header = malloc(8);
    cap_user_data_t cap_data = malloc(12);
    cap_header->pid = getpid();
    cap_header->version = _LINUX_CAPABILITY_VERSION;

    if (capget(cap_header, cap_data)<0)
    {
        perror("failed capget");
        exit(1);
    }
    printf("cap data permitted: 0x%x, effective: 0x%x, inheritable: 0x%x\n",
                cap_data->permitted, cap_data->effective, cap_data->inheritable);
}

void listcaps(){
    cap_t caps = cap_get_proc();
    ssize_t y = 0;
    printf("the peocess %d was give capabilities %s\n", (int)getpid(), cap_to_text(caps, &y));
    fflush(0);
    cap_free(caps);
}
```

```c
int main(){
    cap_t caps = cap_init();
    //允许改变组id，允许设置setuid位，如果文件属于进程的UID就取消对文件的限制 ，允许改变用户ID  允许改变文件的所有权
    cap_value_t caplist[5] = {CAP_SETGID, CAP_FSETID, CAP_FOWNER, CAP_SETUID, CAP_CHOWN};
    pid_t parentpid = getpid();
    if(!parentpid){
        printf("pid error!\n");
        return 0;
    }
    unsigned num_caps = 5;
    cap_set_flag(caps, CAP_EFFECTIVE, num_caps, caplist, CAP_SET);
    cap_set_flag(caps, CAP_INHERITABLE, num_caps, caplist, CAP_SET);
    cap_set_flag(caps, CAP_PERMITTED, num_caps, caplist, CAP_SET);
    if(cap_set_proc(caps)){
        perror("cap_set_proc");
        return 0;
    }
    list_cap();
    listcaps();
    if(chown("/home/zhh/lab2/test_setcap",0 ,0)==0){
        printf("chown succeed\n");
    }
    else
    {
        perror("chown()");
        return 0;
    }
    if(chmod("/home/zhh/lab2/test_setcap", 04777)==0){
        printf("chmod succeed\n");
    }
    else
    {
        perror("chmod()");
        return 0;
    }
    cap_clear(caps);
    if(cap_set_proc(caps)){
        perror("cap_set_proc");
        return 0;
    }
    list_cap();
    listcaps();
    cap_free(caps);
    return 0;
}
```

编译过程中发现有如下错误，对’cap_get_proc’未定义的引用问题

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ rm mysetcap
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ gcc mysetcap.c -o mysetcap
/tmp/ccOzXt0X.o：在函数‘listcaps’中：
mysetcap.c:(.text+0xac)：对‘cap_get_proc’未定义的引用
mysetcap.c:(.text+0xcb)：对‘cap_to_text’未定义的引用
mysetcap.c:(.text+0xff)：对‘cap_free’未定义的引用
/tmp/ccOzXt0X.o：在函数‘main’中：
mysetcap.c:(.text+0x137)：对‘cap_init’未定义的引用
mysetcap.c:(.text+0x1a7)：对‘cap_set_flag’未定义的引用
mysetcap.c:(.text+0x1c5)：对‘cap_set_flag’未定义的引用
mysetcap.c:(.text+0x1e3)：对‘cap_set_flag’未定义的引用
mysetcap.c:(.text+0x1ef)：对‘cap_set_proc’未定义的引用
mysetcap.c:(.text+0x285)：对‘cap_clear’未定义的引用
mysetcap.c:(.text+0x291)：对‘cap_set_proc’未定义的引用
mysetcap.c:(.text+0x2dd)：对‘cap_free’未定义的引用
collect2: error: ld returned 1 exit status
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ gcc mysetcap.c -o mysetcap
-lcap
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$
```

使用-lcap 参数解决

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ gcc mysetcap.c -o mysetcap
-lcap
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$
```

可以发现在经过设置能力位之后，这个程序能够成功的为 test_setcap 这个文件设置 setuid 位，且更改其所属用户和组

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ ls -l test_setcap
-rwxr--r-- 1 fengjinghang fengjinghang 16 12月 17 21:28 test_setcap
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ sudo ./mysetcap
[sudo] fengjinghang 的密码：
cap data permitted: 0xd9, effective: 0xd9, inheritable: 0xd9
the peocess 3480 was give capabilities = cap_chown,cap_fowner,cap_fsetid,cap_set
gid,cap_setuid+eip
chown succeed
chmod succeed
cap data permitted: 0x0, effective: 0x0, inheritable: 0x0
the peocess 3480 was give capabilities =
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ ls -l test_setcap
-rwsrwxrwx 1 root root 16 12月 17 21:28 test_setcap
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$
```

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

int main(){

//测试cap_sys_time能力位是否开启，如果开启，就攻击修改系统时间


    //测试cap_dac_read_search能力位是否开启，如果开启就读取/etc/shadow中的密码
    printf("测试cap_dac_read_search能力位，查看密码：\n");
    system("cat /etc/shadow");
    printf("\n");


    //测试cap_chown能力位是否开启，如果开启就读取/etc/shadow中的密码

    system( "ls -l attacked");
    printf("\n");
    printf("测试cap_chown能力位，修改文件attacked文件的用户，组为root：\n");
    system("chown root:root attacked");
    system( "ls -l attacked");
    printf("\n");

    if (fork()==0)
    {
        printf("当前时间：\n");
        execlp("date", "date", NULL);
    }
    else
    {
        sleep(1);
        printf("设置新的时间：\n");
        execlp("date", "date", "-s", "2019-11-25",NULL);
    }
    sleep(1);

    return 0;
}
```

我们以测试 cap_sys_time 能力位，cap_chown 能力位，cap_dac_read_search 能力位为例，如果用户设置了能力位并且没有及时收回，则攻击者可以利用调用系统函数，利用这一能力位修改系统时间，修改文件的所有者，读取一些不能读取的机密文件。

没有相应能力位时，则攻击者将不会成功

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ ./attack
testing cap_dac_read_search, want to get access to the password
cat: /ets/shadow: 没有那个文件或目录

ls: 无法访问'attacked': 没有那个文件或目录

testing cap_chown, change attack
chown: 无法访问'attacked': 没有那个文件或目录
ls: 无法访问'attacked': 没有那个文件或目录

当前时间:
2020年 12月 17日 星期四 22:02:29 CST
设置新的时间:
date: 无法设置日期: 不允许的操作
2019年 11月 25日 星期一 00:00:00 CST
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$
```

如果设置了相应的能力位，没有收回，则会导致相应攻击实现：
查看了不可查看的密码

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ ./attack
testing cap_dac_read_search, want to get access to the password
root:$6$lq0h8YBZ$HND8Ilh9.l34Bb4tBvERqcjUghVwTNpH.HGR.mmabGqcFjR6UeDrCTqjq/hdX/I9Yhthn4hDwg2vXKnH2CRCu1:18608:0:99999:7:::
daemon:*:17953:0:99999:7:::
bin:*:17953:0:99999:7:::
sys:*:17953:0:99999:7:::
sync:*:17953:0:99999:7:::
games:*:17953:0:99999:7:::
man:*:17953:0:99999:7:::
lp:*:17953:0:99999:7:::
mail:*:17953:0:99999:7:::
news:*:17953:0:99999:7:::
uucp:*:17953:0:99999:7:::
proxy:*:17953:0:99999:7:::
www-data:*:17953:0:99999:7:::
backup:*:17953:0:99999:7:::
list:*:17953:0:99999:7:::
irc:*:17953:0:99999:7:::
gnats:*:17953:0:99999:7:::
nobody:*:17953:0:99999:7:::
systemd-timesync:*:17953:0:99999:7:::
systemd-network:*:17953:0:99999:7:::
systemd-resolve:*:17953:0:99999:7:::
syslog:*:17953:0:99999:7:::
_apt:*:17953:0:99999:7:::
messagebus:*:17954:0:99999:7:::
uuidd:*:17954:0:99999:7:::
lightdm:*:17954:0:99999:7:::
whoopsie:*:17954:0:99999:7:::
avahi-autoipd:*:17954:0:99999:7:::
avahi:*:17954:0:99999:7:::
dnsmasq:*:17954:0:99999:7:::
```

修改了文件权限，系统时间

```
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$ ./attack
testing cap_dac_read_search, want to get access to the password
cat: /etc/shadow: 权限不够

-rw-r--r-- 1 root root 0 12月 17 22:10 attacked

testing cap_chown, change attack
chown: 正在更改'attacked' 的所有者: 不允许的操作
-rw-r--r-- 1 root root 0 12月 17 22:10 attacked

当前时间:
2020年 12月 17日 星期四 22:11:39 CST
设置新的时间:
date: 无法设置日期: 不允许的操作
2019年 11月 25日 星期一 00:00:00 CST
fengjinghang@fengjinghang-ThinkPad-T470p:~/oss-lab2$
```

代码如下：

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

int main(){

//测试cap_sys_time能力位是否开启，如果开启，就攻击修改系统时间


    //测试cap_dac_read_search能力位是否开启，如果开启就读取/etc/shadow中的密码
    printf("测试cap_dac_read_search能力位，查看密码：\n");
    system("cat /etc/shadow");
    printf("\n");


    //测试cap_chown能力位是否开启，如果开启就读取/etc/shadow中的密码

    system( "ls -l attacked");
    printf("\n");
    printf("测试cap_chown能力位，修改文件attacked文件的用户，组为root：\n");
    system("chown root:root attacked");
    system( "ls -l attacked");
    printf("\n");

    if (fork()==0)
    {
        printf("当前时间：\n");
        execlp("date", "date", NULL);
    }
    else
    {
        sleep(1);
        printf("设置新的时间：\n");
        execlp("date", "date", "-s", "2019-11-25",NULL);
    }
    sleep(1);

    return 0;
}
```