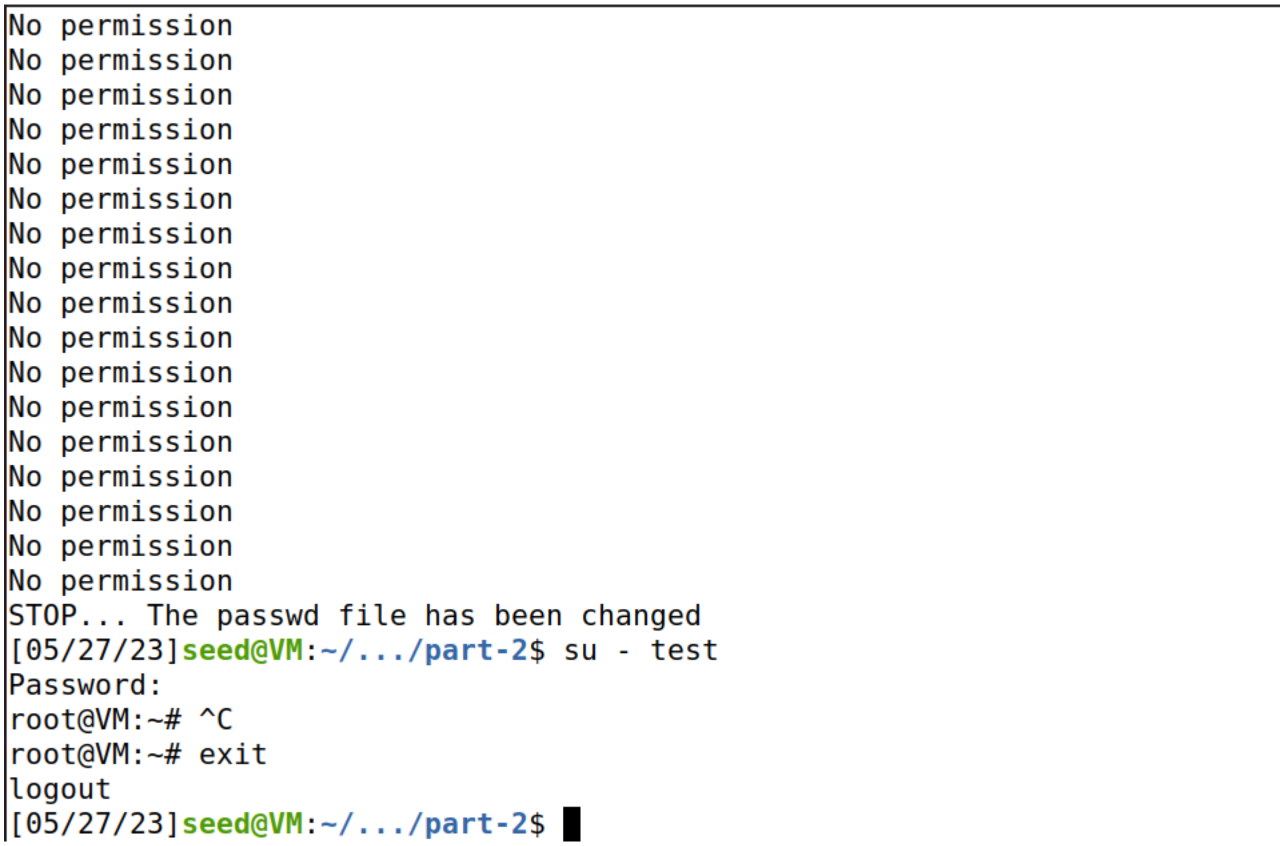
CS489 A1 write up

Skylar Samson Liang

June 2023

1 2.4: rc_attack.c, target_process.sh

The race condition attack is a probabilistic attack so it is not guaranteed to work. The way this attack works is by running the attack program and the vulnerable program in parallel. The hope here is that there is a race condition between the attack program and the vuln program such that a link is made between TOCTOU in the vuln program. More specifically, the attack program will link the password file and the tmp file so that a new user with a new password can be added to the password file. Proof of an attacker (me) logging in as a root user named "test" is shown below.



```
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
STOP... The passwd file has been changed
[05/27/23]seed@VM:~/.../part-2$ su - test
Password:
root@VM:~# ^C
root@VM:~# exit
logout
[05/27/23]seed@VM:~/.../part-2$ 
```

2 4.6: fmt2.py

This attack is complex. I believe the idea here is similar to a buffer overflow attack. I was able to print out the memory using the line containing "%.8x"*400. From this I was able to pinpoint the place in memory that we have to overwrite. However I was unable to overwrite this piece in memory.

If I could overwrite this piece then I believe the attack would go like the following. First I would over write this piece with an address that points to the NOPs slide in the content array. However, we are dealing with a 64-bit memory so we want the address to be at the end of the content array. This address would point **back** to the beginning of the content array to reach the NOP slide. This NOP slide would direct the pointer to go straight to the shellcode which would open a reverse shell. On another terminal I would have ”nc -nv -l 9090” running so that the reverse shell can connect to the netcat instance. This would then allow me root access on the docker server.

Note: The NOP slide is not necessary as I could also just make the shellcode start at 0 and point the the buff_addr. This would be the very beginning of the shellcode and would execute it. So essentially the order in the payload would be:

shellcode — NOPs — buff_addr