Lab 5: Kernel Exploitation

CSC 472/583

Kate Nguyen

Lab performed on 12/13/2021

Introduction

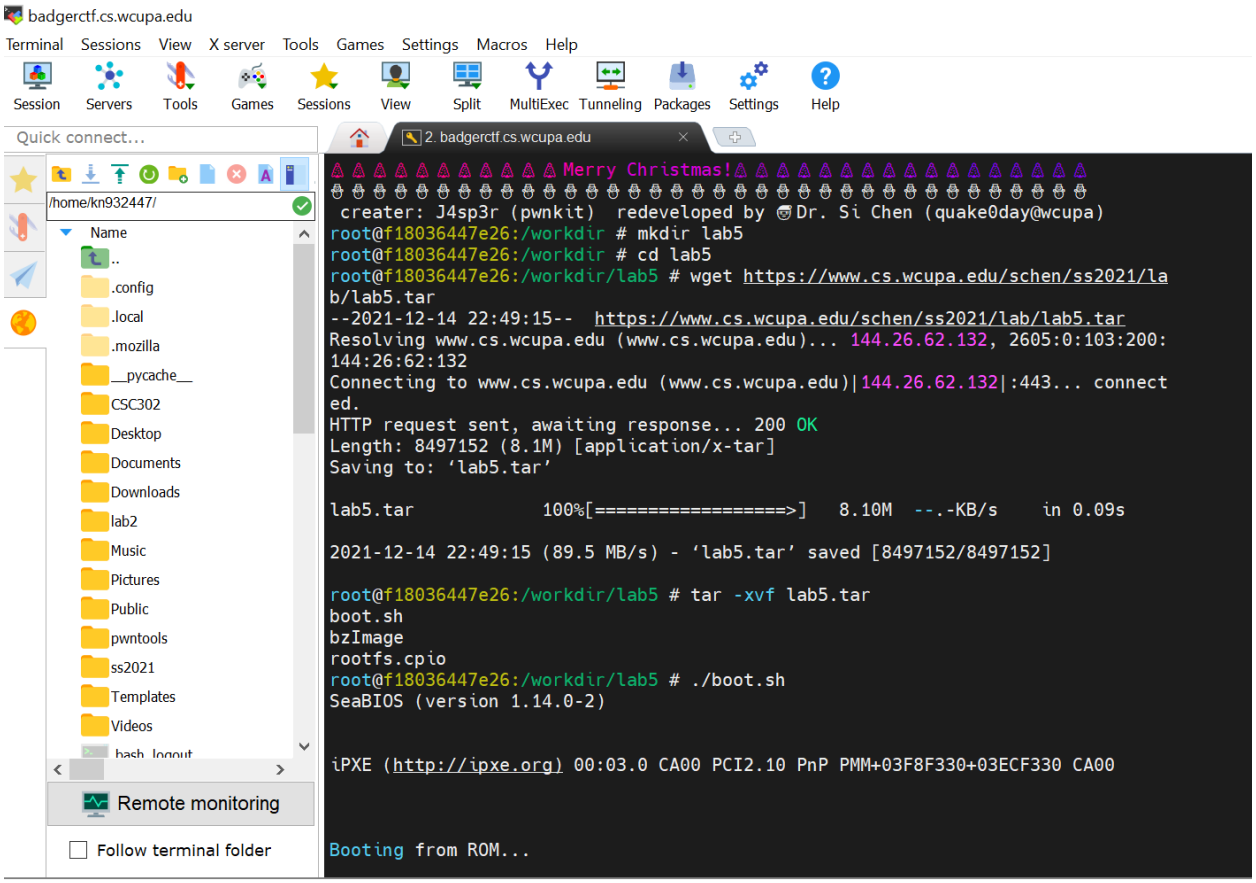
The purpose of this lab is to utilize kernel exploitation and the goals are to under the concept of this exploitation by User After Free (UAF). If after freeing a memory location, a program does not clear the pointer to that memory space, then we could exploit/attack the error to hack into the program.

In this lab, there are five stages in kernel exploitation. A brief overview of kernel exploitation:

1) Find the vulnerability into the kernel code (ie: loadable kernel module (LMK)- device drivers)
2) Manipulate to gain code execution
3) Elevate our process privilege level
4) Survive the "trip" back to userland
5) Lastly, get root privileges

Lab Execution:

Target 1: Boot Up QEMU. Follow the Lab 5 pdf procedures to successfully accomplish this task.



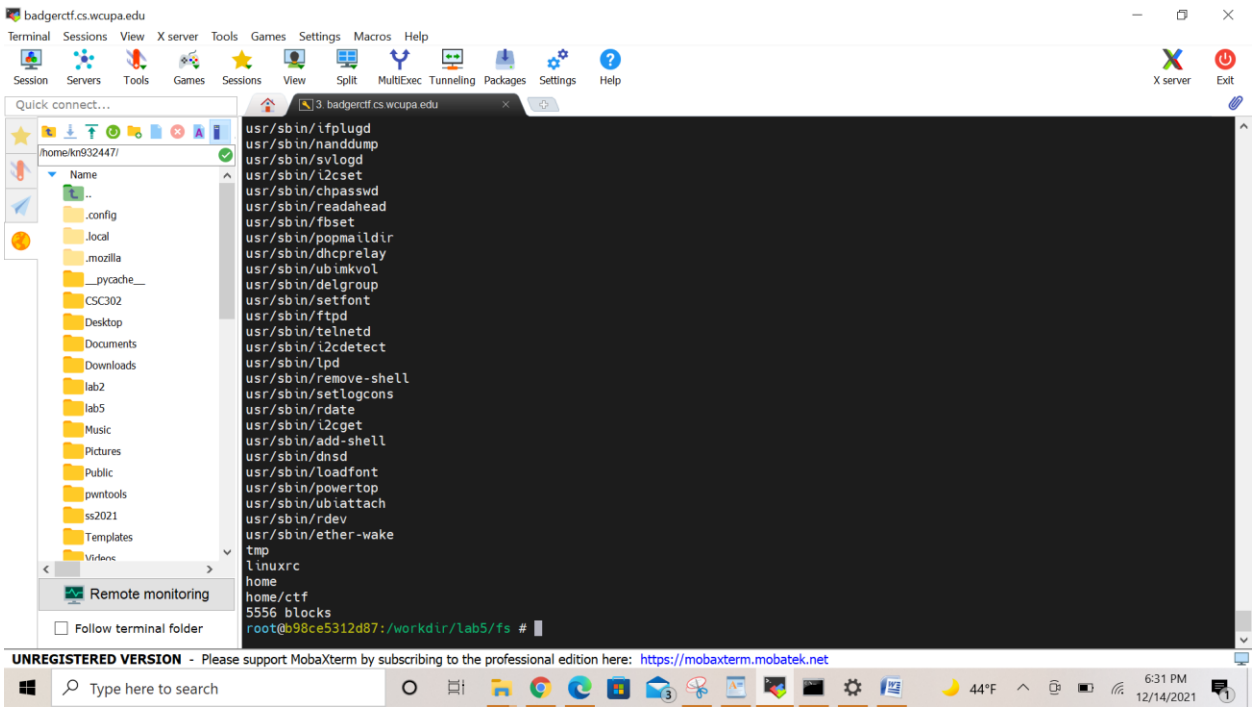
Question 1: how many folders are there inside the root (/) folder?

Answer 1: There are 11 folders. Note that init and linuxrc are not folders.

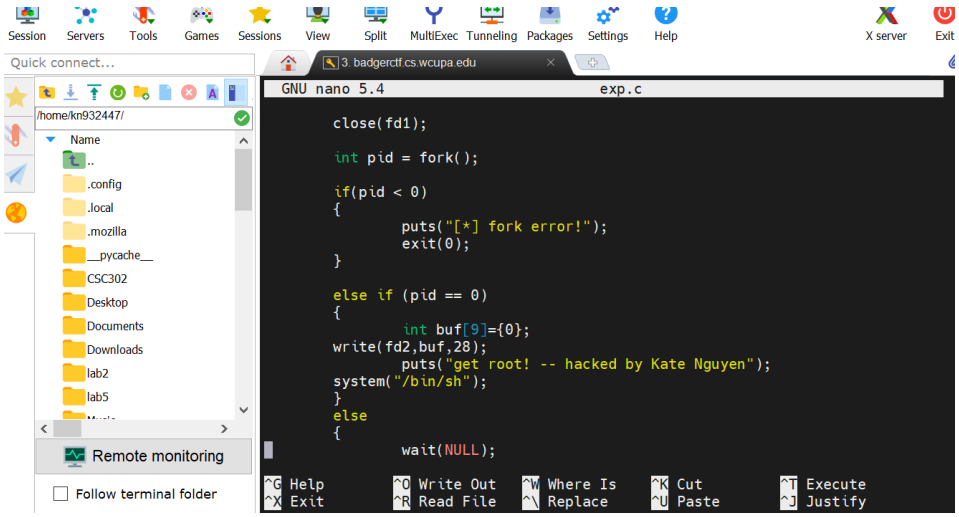
```
Boot took 1.17 seconds

/ $ ls
bin      etc      init     linuxrc  root     sys      usr
dev      home     lib      proc     sbin     tmp
/ $
```

Target 2: Tweaking the Default File System. Follow the Lab 5 pdf procedures to successfully accomplish this task.

Question 2 : Please take a screenshot and show me the output after typing the command "cpio –idmv < rootfs.cpio". Can see that the file system of rootfs.cpio is now unpacked into the fs folder.



Target 3: Compile and Execute the Kernel Exploitation Shellcode. Follow the Lab 5 pdf procedures to successfully accomplish this task. Note: instead of vim to edit exp.c , nano was used.




```
root@b98ce5312d87:/workdir/lab5/fs # find . | cpio -o --format=newc > rootfs.co
cpio: File ./rootfs.cpio grew, 3644416 new bytes not copied
14237 blocks
root@b98ce5312d87:/workdir/lab5/fs #
```

Question 3: (Inside of the QEMU linux virtual machine) Please take a screenshot of the output after typing the command : ./exp

```
Boot took 0.96 seconds

/ $ ./exp
[    37.880711] device open
[    37.882651] device open
[    37.884677] alloc done
[    37.886552] device release
get root! -- hacked by Kate Nguyen
/ #
```

Target 4: Understand UAF.

Please read **exp.c** file and answer the following questions:

Question4 (2 points): In the shellcode (exp.c) why we want to open the device (/dev/babydev) twice?
Question5 (2 points): In the shellcode (exp.c), what's the meaning of ioctl(fd1, 0x1001, 0xa8)? Why use 0xa8?
Question6 (1 point): In the shellcode (exp.c), what's the meaning of write(fd2, zeros, 28)?

Answer 4: In the shellcode (exp.c) we want to open the device (/dev/babydev) twice. By opening the device twice and at the same time, the second time will overwrite the first allocated space because babydev_struct is global. So, when the first one is closed then the second one is released. The second instance is babydev_struct can be re-used.

Answer 5: In the shellcode (exp.c), the meaning of ioctl(fd1, 0x1001, 0xa8). The oi control function, 0x1001 is a hard coded value that will check the command and then call the kfree function, kmalloc function which is the size of the 0xa8 and will allocate a new memory address that is 0xa8. Why use 0xa8? 0xa8 is the device buffer length when it calls baby open – telling it how much space we want to reallocate.

Answer 6: In the shellcode (exp.c), the meaning of write(fd2, zeros, 28) – This will go/write to babydev_struct and put in 28 zeros in the cred_struct. Why? Putting the uid and gid to zero will turn this process from a user to a root user.

Discussion & Conclusion

The goal of this lab was to understand the concepts of kernel exploitation and UAF vulnerabilities in the kernel land. Subjectively, the hardest part of this lab was to log into badger, I have tried to ssh and molly into badger on multiple attempts and also using the university's vpn, either was successful. I was not able to perform this lab remotely and resorted to going on site to perform the lab.