1. **0-run-command**

In this case, I got two ways to achieve the target.

1. The first one, the concepts of **redirection** and **command substitution** were applied.

**$(cat<flag)**

The content of flag will output to cat then $() will capture the result of the commend of cat.

**\*\*\*\*\***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Program\*\*\*\*\***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

$ ./run-command

THIS IS A SECURE LS MACHINE

Which directory do you want to take a look at?

$(cat<flag)

ls: cannot access **'cand{use\_$(<>)}**': No such file or directory

1. The second concept is **Internal Field Separator (IFS).** The Internal Field Separator (IFS) that is used for word splitting after expansion and to split lines into words with the read builtin command.

**$(cat${IFS}flag)**

**\*\*\*\*\***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Program\*\*\*\*\***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**$ ./run-command**

THIS IS A SECURE LS MACHINE

Which directory do you want to take a look at?

$(cat${IFS}flag)

ls: cannot access **'cand{use\_$(<>)}**': No such file or directory

1. **1-guess-my-random**

In this case, the vulnerability exists in the function of sprint(). After tracking it by using gdb, I found the certain size of buffer from ebp-0x88 was copied to ebp-0x48. The function also adds the hard coded characters. Thus, there is extra characters that go over the edge of the size of the buffer.

**guess.py**

Text

Description automatically generated

1. **2-one-format-string**

In this case, we can find that there exists the stack cookie by using “pwn checksec one-format-string”. In addition, the max\_read\_size was assigned as 63 to limit the size we can read. Thus, two things we have to do. First, pass the stack cookie. Second one is to modify the size of max\_read\_size.

By inspect the target function, we know the stack cookie will be generated then stored at ebp-0x8. Furthermore, using “print &max\_read\_size” get the exact location of max\_read\_size then use the idea of arbitrary write to change the size of it.

In order to align the code, the format string using will be filled as the multiple of eight. After we set them down, we can put the stack cookie at the ebp-0x8 to pass it then put the please\_run\_this function at the return address because we crack the size limit (63→128).

**fs.py**

**Text

Description automatically generated**

1. **3-2048**

In this case, I found one thing that we can put a bunch of illegal characters like 9 except ‘a’, ‘d’, ‘s’, ‘w’ to make the game keep do the judgement. The size of input is assigned as 20 bytes. However, there exist a vulnerability allowed us to achieve our goal. The first of every 20 characters will be read then judged like 1st, 21th, 41th and so on.

The range of type of short is from 0 to 4294967295. When the score is deducted under 0, the score will go to 4294967295. Thus, while a bunch of illegal characters are inputted and keep the score down. In the end, we can get the score greater than 3932156. The last step is to trigger the judgement to judge whether I achieve the requirement.

**exploit.py**

**Text

Description automatically generated**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Program\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**$ ./2048 < input.txt**

**A screenshot of a computer

Description automatically generated**

1. **4-where-to-jump/**
2. **5-get-flag-without-write-nor-exec**

In this case, we will make buffer overflow then use the ROP to achieve the target.

First, use the open function to open the file then use the read function to put the content into the store zone. Then the existed instruction is used to load the data step by step. In addition, the extra space need to be filled up due to two lines “add $0x20,%rsp” and “pop %rbp”. Thus, we must give 0x28 padding before we called the exit function. In the end, the exit function will be used to return the ASCII code.

**get.py**

**Text

Description automatically generated**

1. **6-deprivileged**

In this case, the program has opened a file called good luck then read and print it out. First, we create symbolic link flag to “good luck”. After that, when the program opens the file, the flag file will be opened instead. Next, we use padding to fill the buffer then use ROP to achieve the target. Read the content into the store zone then print it out.

**dep.py**

**Text

Description automatically generated**

1. **7-tocttou**

In this case, the concept we use here is Time-of-check to time-of-use. In the beginning, the program examines whether the group id match the requirement. Thus, we create a symbolic link “A” to “fake\_flag” then sending “A” to the program to examine. After it pass the examination, we change the link “A” to “flag” which we want to get.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Process\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Text

Description automatically generated

Give “A” to run the program then change the symbolic link before the processing finish.

Text

Description automatically generated



Text

Description automatically generated

1. **8-caffeinated-tocttou**

In this case, we use the same concept as same as the last challenge. However, here has no sleep time provided. Thus, we need to write a script then run the while loop to change the symbolic quickly to successfully get the flag.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Process\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Create a symbolic “A” to “fake\_flag” first.Graphical user interface, text

Description automatically generated

Then, write a script do the symbolic change.

Text

Description automatically generated

**Text

Description automatically generated**

1. **9-guess-passwd**

In this case, we guess the password by determining the process time. The program will sleep 0.02 seconds if a character matches the answer. Thus, the time consume is a way to guess the answer. I used the while loop and exception handling to try the password automatically.

**guess.py**

**Text

Description automatically generated**

1. **a-rop-static**

In this case, we do not use library, we use system call to achieve the target. First, open the file by putting 5 into register eax because its system call number is 0x5. Then, the concept of ROP is used here to run the read and write function which are already existed in the program to store the file content into the store zone and write it out.

**rop.py**

**Text

Description automatically generated**