Challenge: Regularity

Challenge Description :

Nothing much changes from day to day. Famine, conflict, hatred - it's all part and parcel of the lives we live now. We've grown used to the animosity that we experience every day, and that's why it's so nice to have a useful program that asks how I'm doing. It's not the most talkative, though, but it's the highest level of tech most of us will ever see....

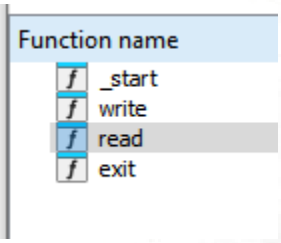
Context :

- You are given a compiled file, analyze it and get the flag.

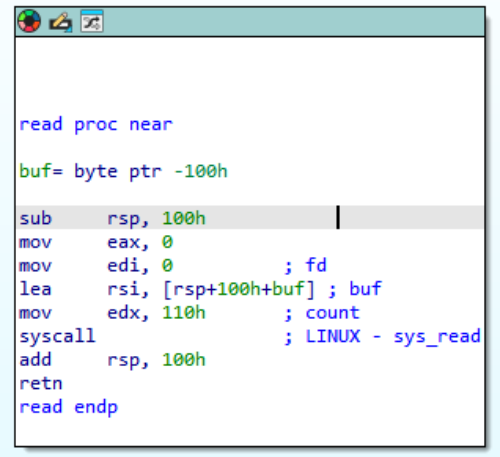
Flag :

- First Install the files and start the instance. Connecting to the server as soon as possible.
- The interface given to you isn't much, when you run the compiled file with Ltrace or Strace not much is given.

```
birdo@DESKTOP-0ENQDDA:~$ nc 94.237.59.63 56902
Hello, Survivor. Anything new these days?
4
Yup, same old same old here as well...
2
42
birdo@DESKTOP-0ENQDDA:~$
```



- On Hex-Ida there only seems to be a limited amount of functions and not a lot of strings either, within the file.
- Analyzing the Binary code we find a Buffer-Overflow exploit present, meaning we can potentially gain a RCE abusing this.

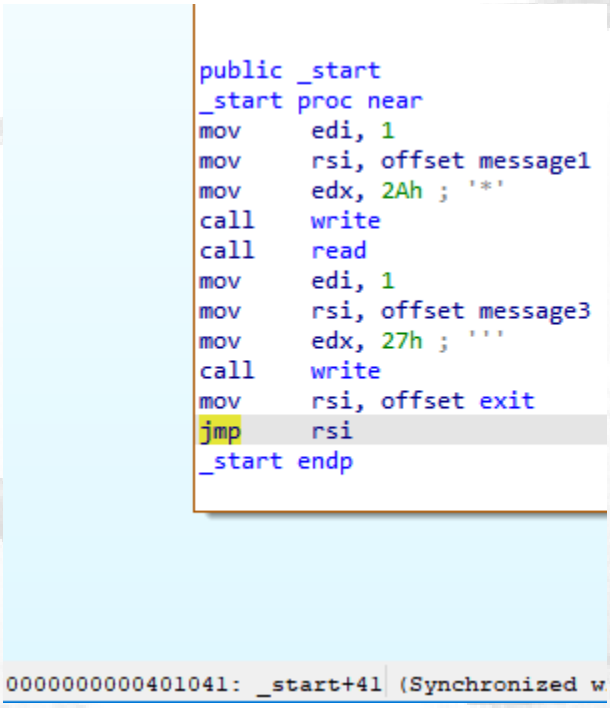


```
read proc near

buf= byte ptr -100h

sub     rsp, 100h
mov     eax, 0
mov     edi, 0          ; fd
lea     rsi, [rsp+100h+buf] ; buf
mov     edx, 110h       ; count
syscall                 ; LINUX - sys_read
add     rsp, 100h
retn
read endp
```

- The Buffer-Overflow exploit is within the read function where we create a buffer on a stack of 100 bytes that we can read, but it's changed to read 110 bytes of data.
- That gives us enough space to write some shell code on the stack more possible due to no protection on the file.
- Meaning we can also overwrite the address that reaches the stack, we need to find where the stack is, so we can jump there.
- In the start function the read function gets the address from the [rsi] binary, making it the stack address that is storing it. It also contains a [jmp rsi] binary we can use on the address 0x401041



- To exploit the server with this Buffer-Overflow I'm going to create a script in python using PWN to send over a Buffer-Overflow payload and hopefully give me a easy shell using that address.

```
birdo@DESKTOP-0ENQDDA:/mnt/c/Users/drobo/Music/htb-challanges$ python3 regularity.py
[+] Opening connection to 94.237.59.63 on port 56902: Done
[DEBUG] Connected to remote server
[DEBUG] Sending payload
[DEBUG] Payload sent
[DEBUG] Shell successfully spawned
Flag: HTB{jMp_rSi_jUmP_aLl_tH3_w4y!}
[*] Switching to interactive mode
$
```

- Making the script with the exploit as a poc finally worked and got a shell.

- The script will be on my github if you interested here is the {link}
- The Flag given to us is : HTB{jMp_rSi_jUmP_aLl_tH3_w4y!}