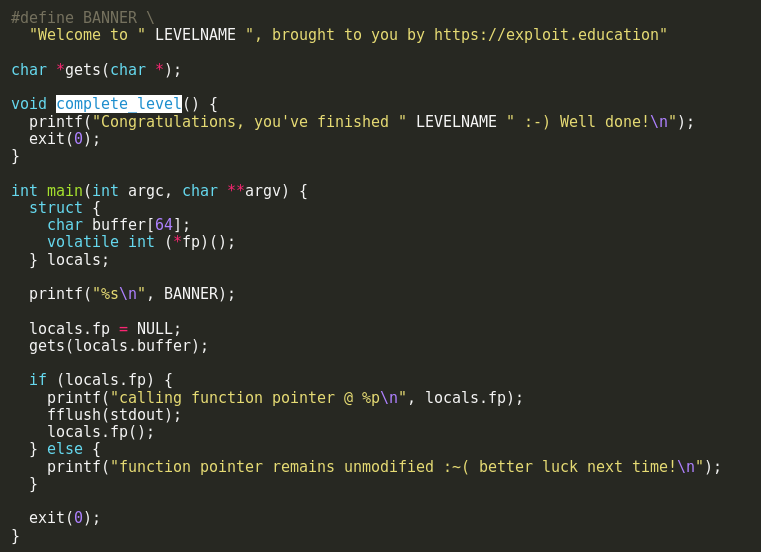
Pheonix stack-three write-up

**Intro**

this level is a bit advanced from the previous level because here we introduce the concept of redirecting code execution to another function by overwriting the EIP(32-bit instruction pointer register).In our case we will redirect code excution to complete\_level function.



**Solution**

As always we start by analyzing our code carefully and check out the vulnerable function again here it’s the gets function which copies undetermined amount of bytes so we know from this that our attack vector will be overflowing the stack and to overwrite the address of EIP by putting the address of the complete\_level function, there are three steps we will follow to solve this level:

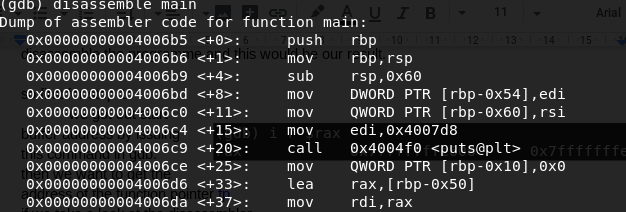
1. find the difference between the start of the buffer and the function pointer fp
2. get the address of complete\_level function
3. craft our exploit

One resource that I found very useful was a [gdb cheat sheet](http://darkdust.net/files/GDB%20Cheat%20Sheet.pdf) I found online.so first we will disassemble the programme and this would be our result.

so first for step :

we got out start buffer address by issuing this command in gdb. 

then we want to get the address of the function pointer fp

if we take a look at the disassembler 

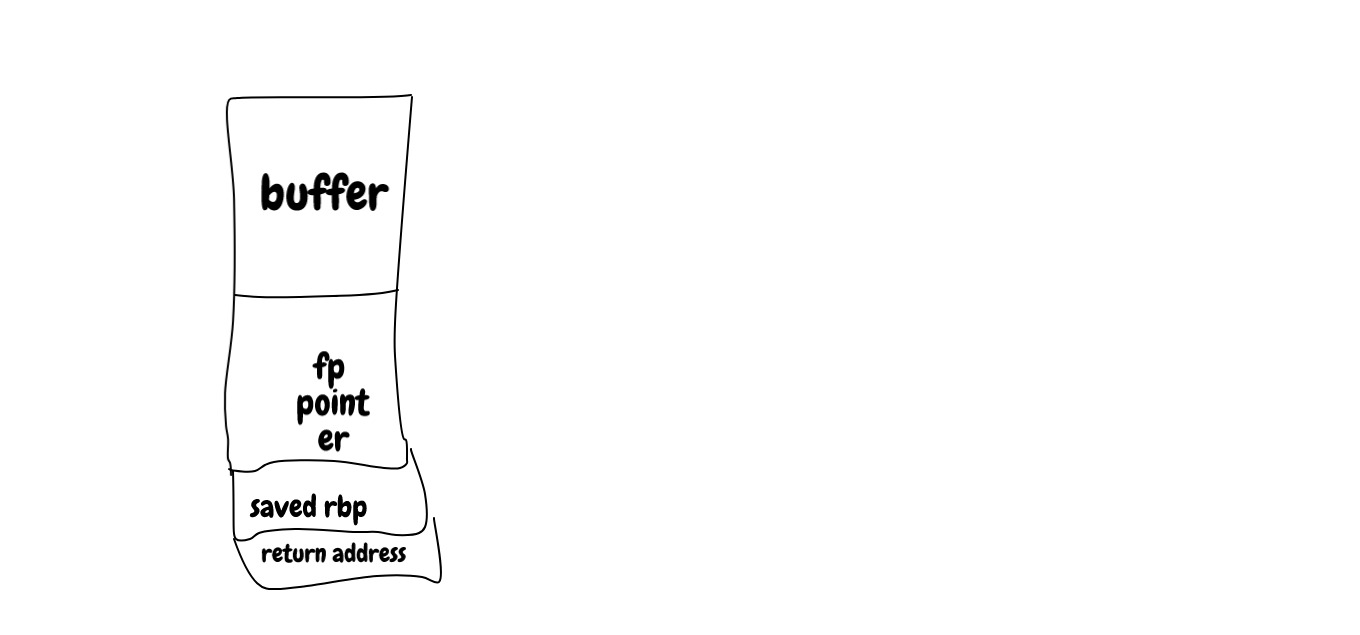
in this line 0x00000000004006ce <+25>: mov QWORD PTR [rbp-0x10],0x0 we can get the address of the address of the function pointer fp by issuing this command in gdb x/x $rbp-0x10 so the result is 0x7fffffffe640 now we subtract the two addresses

0x7fffffffe640 - 0x7fffffffe600 = 0x40 base 16 = 64 base 10

second step:

we will learn now about a nice command called [objdump](https://en.wikipedia.org/wiki/Objdump) we will use it to get the address of the complete function by typing objdump -t ./stack-three | grep complete\_level





third step:

now our exploit is ready to be crafted our stack looks something like this

so our exploite is

