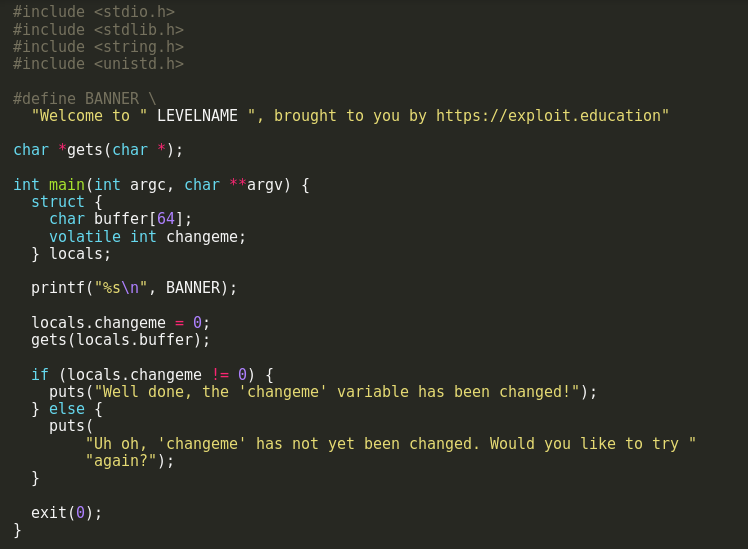
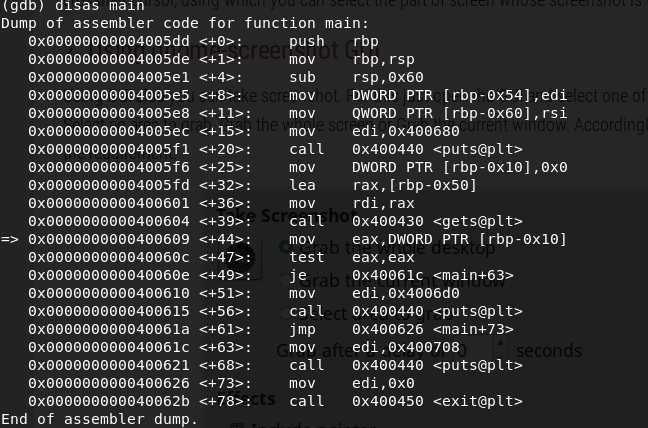
Pheonix stack-zero write-up

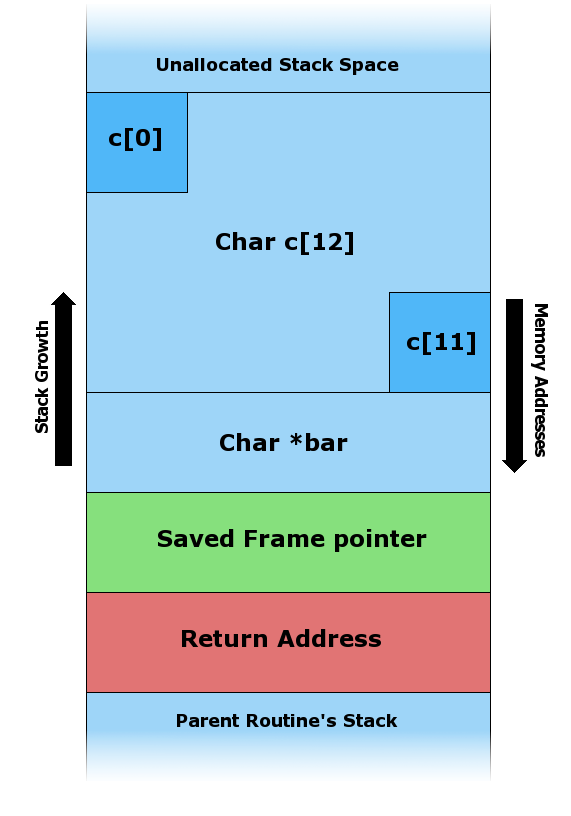
**Intro**

Basically this level introduces the concept of [buffer-overflow](https://en.wikipedia.org/wiki/Buffer_overflow) vulnerability, the program was easy to read but the point is that we have to look for the vulnerable function, 

the vulnerable function here obviously is gets function which has no limit to copy the input from the user so if we give a sufficient input we can change the value of the locals.changeme.

**Solution**

I approached this problem by determining the address of the start of the buffer and the address of the locals.changeme 

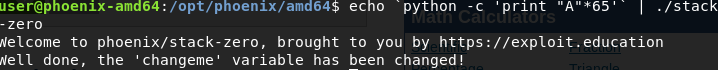
the first two lines of the assembler dump is function prologue, what we are interested in is the address of the locals.changeme and the buffer, I already assume that you know how stack functions and that these are local variables located on the stack here’s a quick reminder how the stack works the stack grows towards lower memory address as shown in the picture 

so if we can get the address of the buffer which is located below the locals.changeme and growth towards higher memory address we can change it’s value.

as we can see from the disassembler that the address of the locals.changeme is at [rbp-0x10] and we know that because it clear from the disassembler that we are moving 0x0 to this address which it’s value = 0x7fffffffe630 

and the start of the buffer is at [rbp-0x50] we got this address from the disassembler above because it’s often to see [lea instruction](https://www.felixcloutier.com/x86/lea) when we want to get the buffer address another way to know this address is to give input in gdb like (AAAAAAAAAAAAAAAAAAAAA) and to check where there hex (\x41) starts when examining the stack with x/50wx $rsp.

now we have the two address so we subtract the address of locals.changeme from the address of the buffer we got 0x40 which how many bytes we want to write to to overwrite the value of locals.changeme

we can see that we got 0x40 as result to reach locals.changeme which equal to 64 in hex and now we will craft our final exploit script

and this how I approached this problem.

I hope you enjoyed my write-up any comment or suggest is appreciated you can contact me via my e-mail address: omaroobaniessa@gmail.com