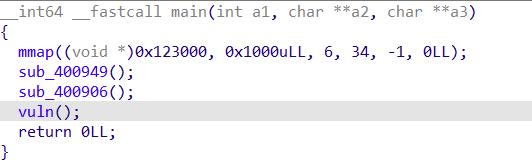
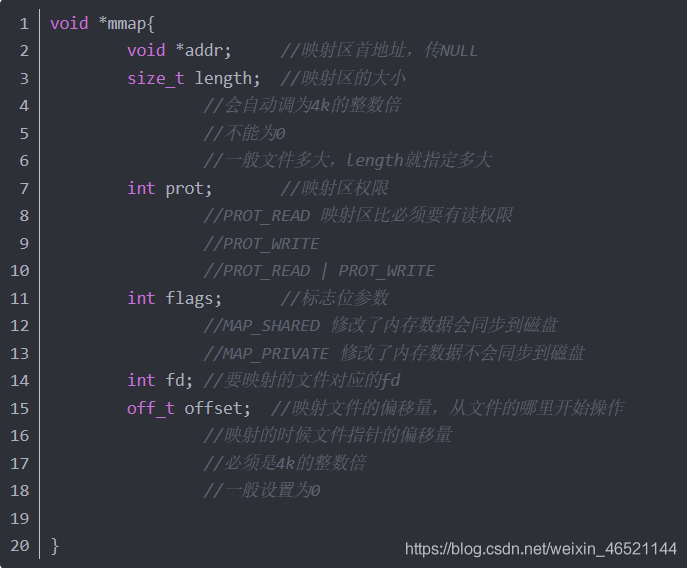
用vmmap开启了一段内存



这里mmap参数类型是（起始地址，大小，保护类，文件描述符等）



我们只需要关注保护类权限

PROT\_READ是0x1

PROT\_WRITE是0x2

PROT\_EXEC是0x4

也就是read,write,execute的顺序,那么这里是6，也就是write,execute权限都有

程序逻辑很简单，存在0x18字节溢出，看是否能进行栈迁移，目前可控输入只有buf，但是buf只有0x20大小不够rop，所以尝试别的方法。程序在最开始mmap了一块内存，可写可执行，那么接下来目标就是1.在这块内存上写shellcode

2.跳转过来执行

**由于不能rop且栈是可执行的想到能不能进行jmp rsp。**

<https://blog.csdn.net/weixin_46521144/article/details/115196495>

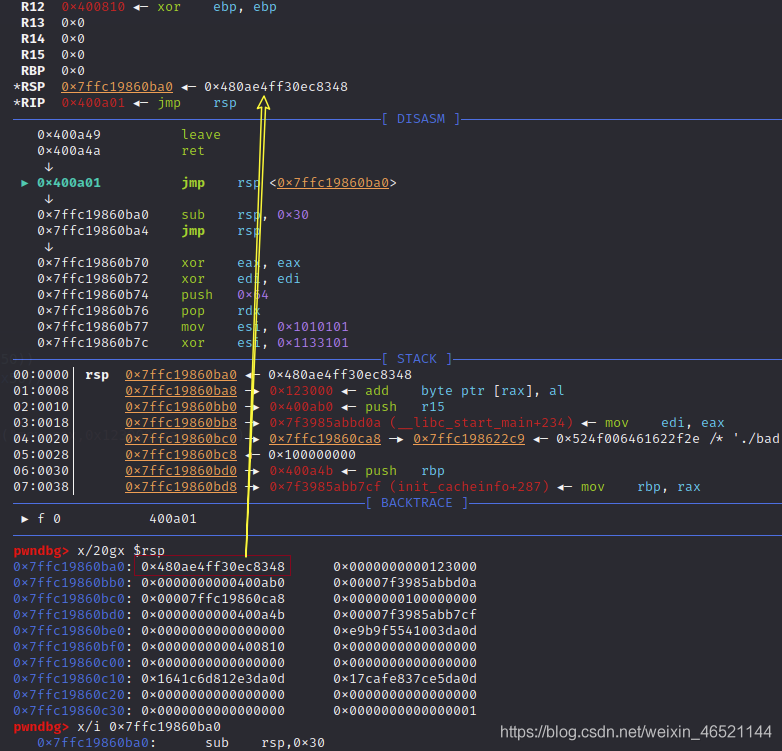
为什么要写jmp\_rsp？我们用gdb看一下



溢出时rsp



注意红框的内容



为什么要写jmp rsp呢？

看上面两张图，如果不写，rip会自动到0x480ae4ff…这个地方，解析这个地址的内容，发现是0x7ffc19860b98，但是这一句话不是任何指令！因此需要jmp rsp把rip跳到被jmp取地址过后的rsp上面，完成了一步取地址，那么执行前执行下一步时就可以执行shellcode了

总的来说，如果直接栈溢出，想要一溢出就执行shellcode是不可行的。要么使用call,要么使用像这里的jmp rsp，要么把shellcode的起始地址写到溢出的位置。

之后再sub rsp 40,jmp rsp就很好理解了。这个有点像attacklab里面的前三题。

在此时执行的buffer里面写的时read,read进入的也是shellcode,之后用到call rax就是之前说到的

