# 攻防世界 echo-server writeup

原创

流风_霜 于 2021-01-07 17:04:09 发布 202 收藏 1

## 攻防世界 echo-server writeup

## 一. 知识点

### （1）花指令

0x01 概念
花指令是，由设计者特别构思，希望使反汇编的时候出错，让破解者无法清楚正确地反汇编程序的内容，迷失方向。经典的是，目标位置是另一条指令的中间，这样在反汇编的时候便会出现混乱。花指令有可能利用各种指令：jmp, call, ret的一些堆栈技巧，位置运算，等等。

0x02 常见花指令

| 机器码 | 汇编语言 |
|---|---|
| 9A | CALL immed32 |
| E8 | CALL immed16 |
| E9 | JMP immed16 |
| EB | JMP immed8 |

0x03 如何去除花指令
第一步：

- 识别哪些是有用的数据，哪些是垃圾数据。

第二步：

- 把垃圾数据用nop(0x90h)填充。

### （2）IDA出现"sp-analysis failed"

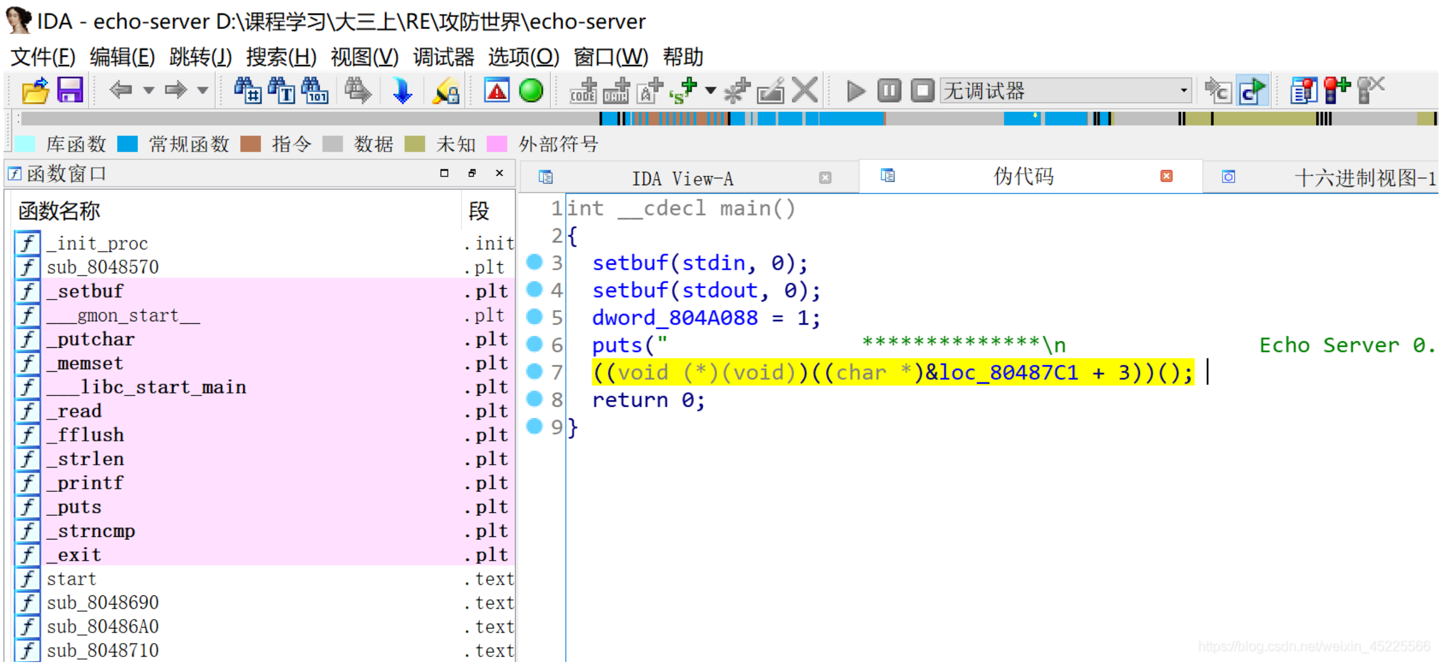
IDA官网解释：

```
Problem: Failed to trace the value of the stack pointer

 Description:
        The value of the stack pointer at the end of the function is different
        from its value at the start of the function. IDA checks for the
        difference only if the function is ended by a "return" instruction.
        The most probable cause is that stack tracing has failed.
        This problem is displayed in the disassembly listing with
        the "sp-analysis failed" comment.
 What to do:
        1. Examine the value of stack pointer at various
           locations of the function and try to find out why the stack tracing
           has failed. Usually, it fails because some called function changed the
           stack pointer (by purging the input parameters, for example)
        2. If you have found the offending function, change its
           attributes (namely, number of bytes purged upon return).
        3. Another way is to specify manually how the stack pointer is
           modified. See Change stack pointer command
```

大意是ida检测到，IDA有栈跟踪的功能，它在函数内部遇到ret(retn)指令时会做判断：栈指针的值在函数的开头/结尾是否一致，如果不一致就会在函数的结尾标注"sp-analysis failed" 。一般编程中，不同的函数调用约定(如stdcall&_cdcel call)可能会出现这种情况；另外，为了实现代码保护而加入代码混淆(特指用push/push+ret实现函数调用)技术也会出现这种情况。

## 二. 解题步骤

（1）拖进IDA反编译，查看伪代码，可以看到这里的函数调用很奇怪

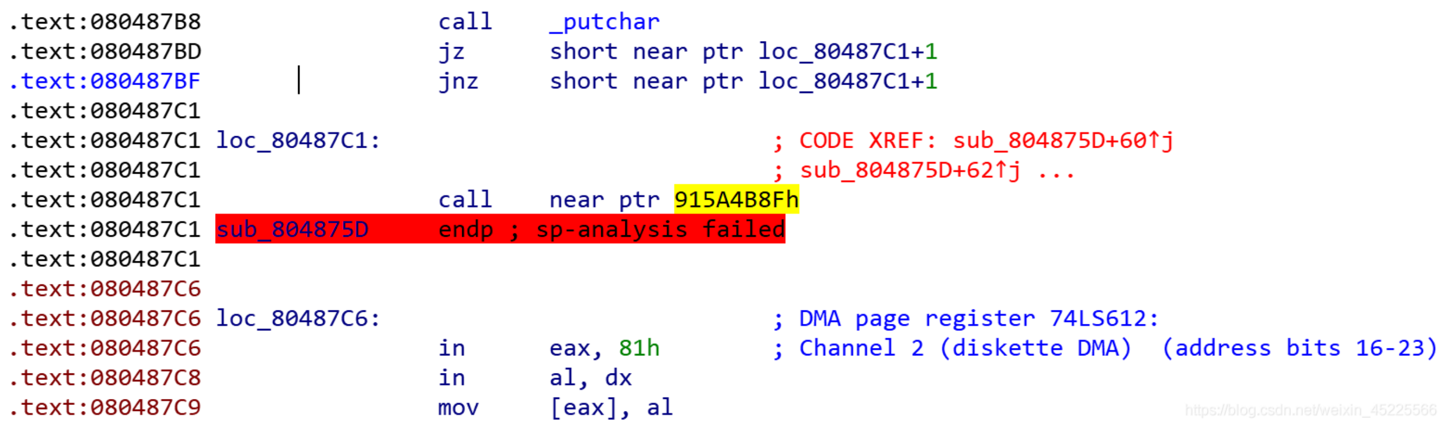

IDA - echo-server D:\课程学习\大三上\RE\攻防世界\echo-server

文件(F) 编辑(E) 跳转(J) 搜索(H) 视图(V) 调试器 选项(O) 窗口(W) 帮助
无调试器
库函数 常规函数 指令 数据 未知 外部符号
函数窗口
IDA View-A
伪代码
十六进制视图-1

| 函数名称 | 段 |
|---|---|
| _init_proc | .init |
| sub_8048570 | .plt |
| _setbuf | .plt |
| ___gmon_start__ | .plt |
| _putchar | .plt |
| _memset | .plt |
| ___libc_start_main | .plt |
| _read | .plt |
| _fflush | .plt |
| _strlen | .plt |
| _printf | .plt |
| _puts | .plt |
| _strncmp | .plt |
| _exit | .plt |
| start | .text |
| sub_8048690 | .text |
| sub_80486A0 | .text |
| sub_8048710 | .text |

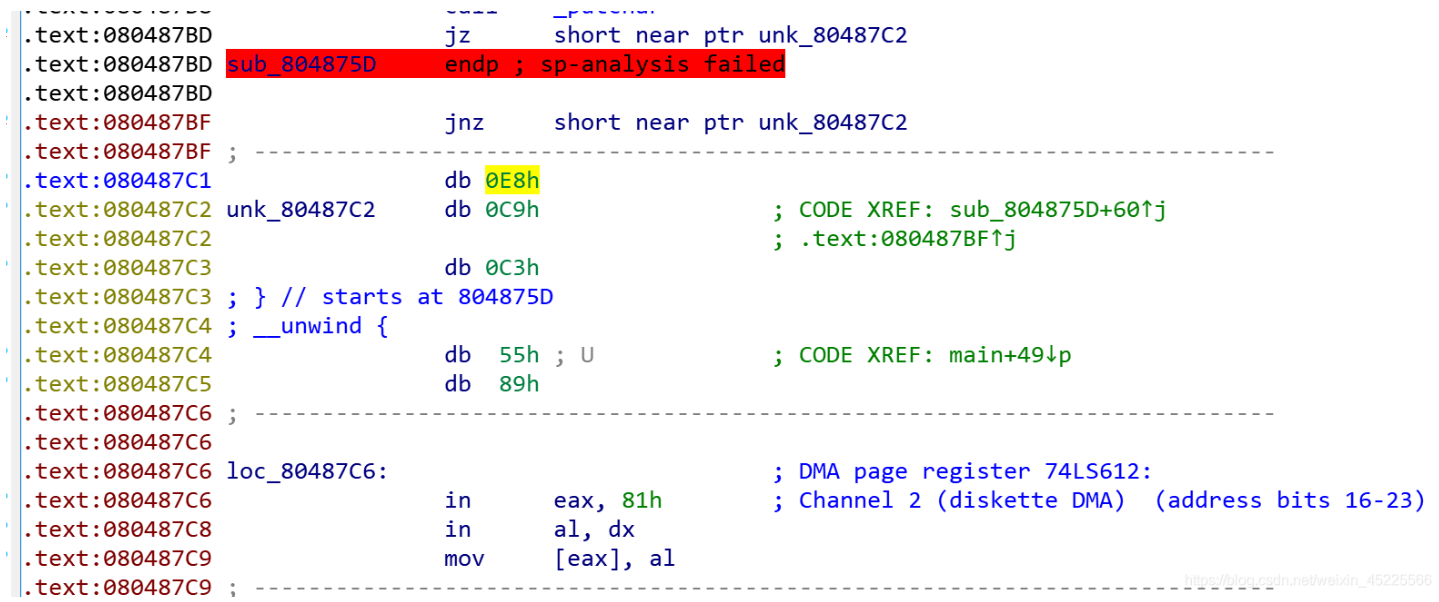
```
int __cdecl main()
{
  setbuf(stdin, 0);
  setbuf(stdout, 0);
  dword_804A088 = 1;
  puts("              **************\n          Echo Server 0.
  ((void (*)(void))((char *)&loc_80487C1 + 3))();
  return 0;
}
```

（2）跟去这个函数，发现汇编代码很乱，有花指令插入
程序本应该跳转到loc_80487C1+3处，但这里并没有显示出来，由此判断程序被混淆，混入了花指令，需要去除花指令。



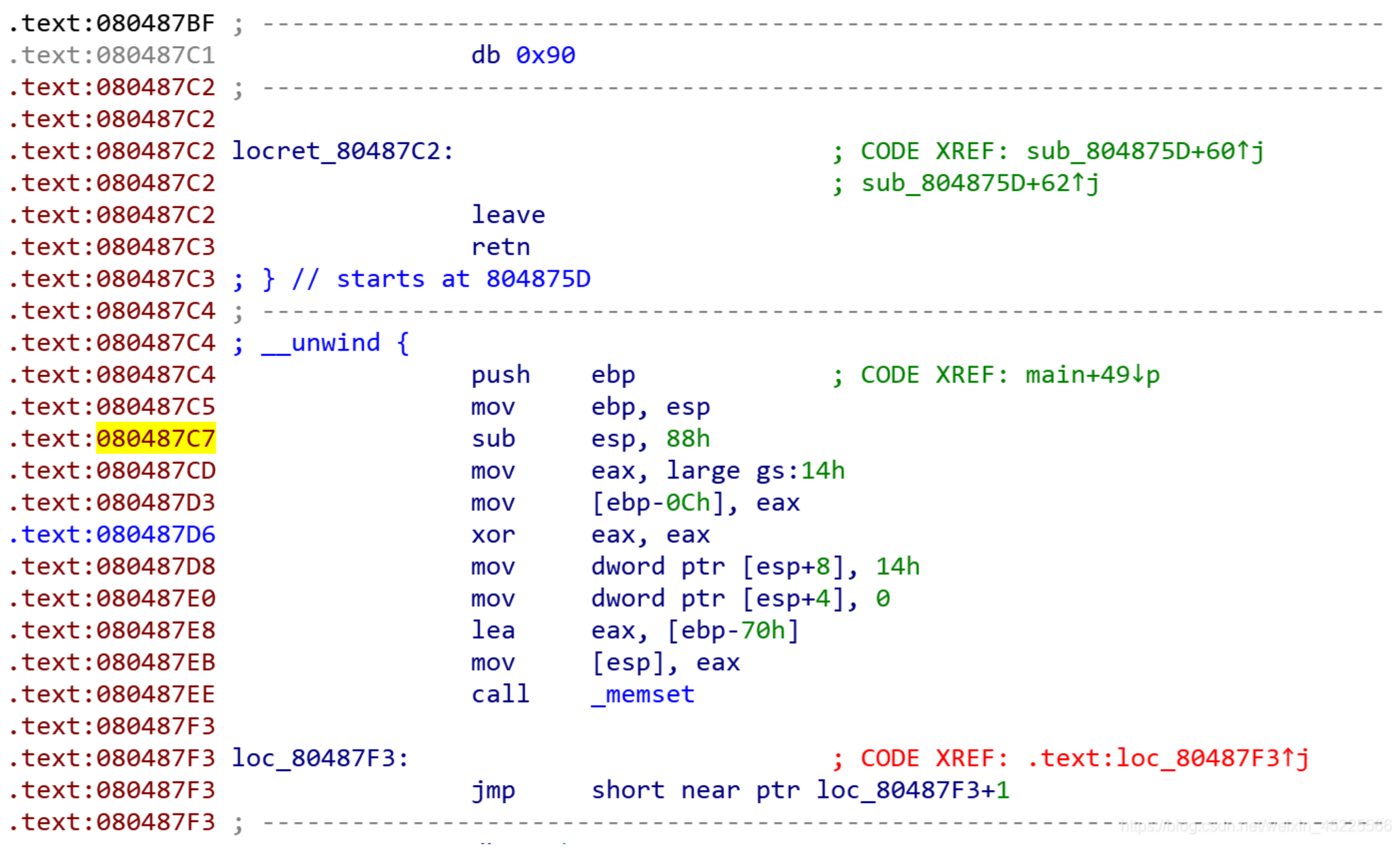
```
.text:080487B8                 call    _putchar
.text:080487BD                 jz      short near ptr loc_80487C1+1
.text:080487BF                 jnz     short near ptr loc_80487C1+1
.text:080487C1
.text:080487C1 loc_80487C1:                            ; CODE XREF: sub_804875D+60↑j
.text:080487C1                                         ; sub_804875D+62↑j ...
.text:080487C1                 call    near ptr 915A4B8Fh
.text:080487C1 sub_804875D     endp ; sp-analysis failed
.text:080487C1
.text:080487C6
.text:080487C6 loc_80487C6:                            ; DMA page register 74LS612:
.text:080487C6                 in      eax, 81h        ; Channel 2 (diskette DMA)  (address bits 16-23)
.text:080487C8                 in      al, dx
.text:080487C9                 mov     [eax], al
```

（3）将汇编代码转换成数据，发现0x80487C1处的字节是0xE8.



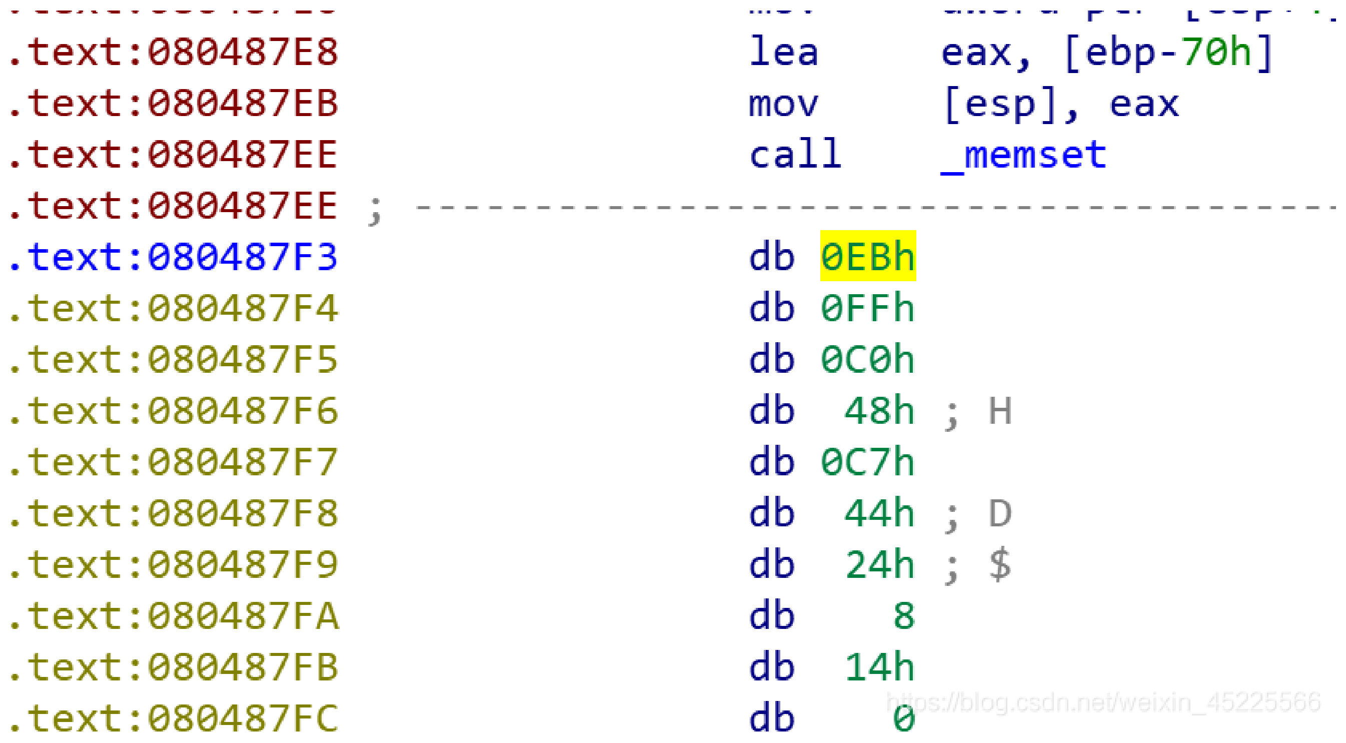
```
.text:080487B8                 call    _putchar
.text:080487BD                 jz      short near ptr unk_80487C2
.text:080487BD sub_804875D     endp ; sp-analysis failed
.text:080487BD
.text:080487BF                 jnz     short near ptr unk_80487C2
.text:080487BF ; ---------------------------------------------------------------------------
.text:080487C1                 db 0E8h
.text:080487C2 unk_80487C2     db 0C9h                 ; CODE XREF: sub_804875D+60↑j
.text:080487C2                                         ; .text:080487BF↑j
.text:080487C3                 db 0C3h
.text:080487C3 ; } // starts at 804875D
.text:080487C4 ; __unwind {
.text:080487C4                 db  55h ; U             ; CODE XREF: main+49↓p
.text:080487C5                 db  89h
.text:080487C6 ; ---------------------------------------------------------------------------
.text:080487C6
.text:080487C6 loc_80487C6:                            ; DMA page register 74LS612:
.text:080487C6                 in      eax, 81h        ; Channel 2 (diskette DMA)  (address bits 16-23)
.text:080487C8                 in      al, dx
.text:080487C9                 mov     [eax], al
.text:080487C9 ; ---------------------------------------------------------------------------
```

（4）使用IDA把它patch为0x90，转换为代码，汇编代码显示正常



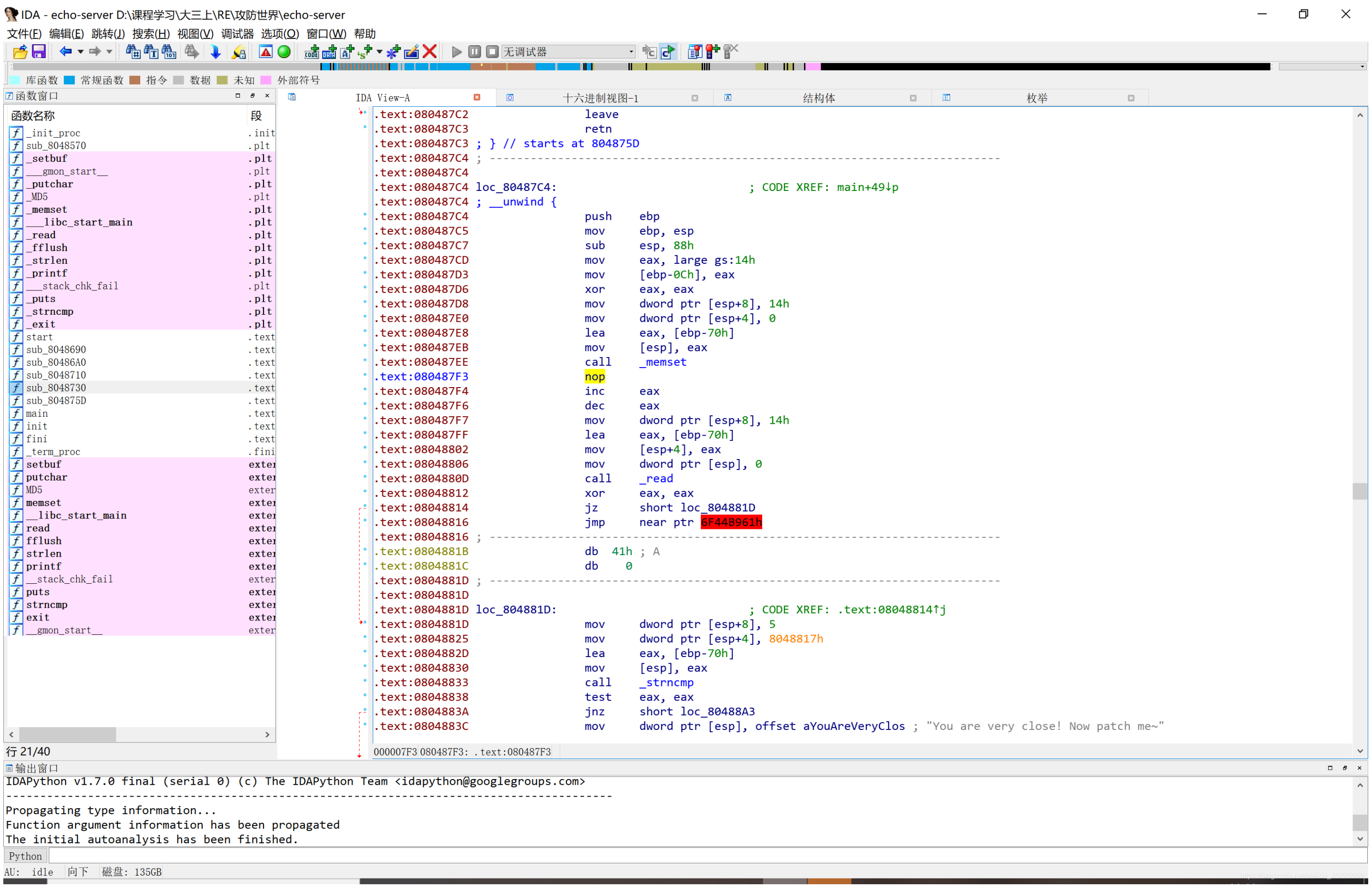
```
.text:080487BF ; ---------------------------------------------------------------------------
.text:080487C1                 db 0x90
.text:080487C2 ; ---------------------------------------------------------------------------
.text:080487C2
.text:080487C2 locret_80487C2:                         ; CODE XREF: sub_804875D+60↑j
.text:080487C2                                         ; sub_804875D+62↑j
.text:080487C2                 leave
.text:080487C3                 retn
.text:080487C3 ; } // starts at 804875D
.text:080487C4 ; ---------------------------------------------------------------------------
.text:080487C4 ; __unwind {
.text:080487C4                 push    ebp             ; CODE XREF: main+49↓p
.text:080487C5                 mov     ebp, esp
.text:080487C7                 sub     esp, 88h
.text:080487CD                 mov     eax, large gs:14h
.text:080487D3                 mov     [ebp-0Ch], eax
.text:080487D6                 xor     eax, eax
.text:080487D8                 mov     dword ptr [esp+8], 14h
.text:080487E0                 mov     dword ptr [esp+4], 0
.text:080487E8                 lea     eax, [ebp-70h]
.text:080487EB                 mov     [esp], eax
.text:080487EE                 call    _memset
.text:080487F3
.text:080487F3 loc_80487F3:                            ; CODE XREF: .text:loc_80487F3↑j
.text:080487F3                 jmp     short near ptr loc_80487F3+1
.text:080487F3 ; ---------------------------------------------------------------------------
```

（5）在上图中可以看到，0x080487F3处汇编代码也存在花指令，转换成数据后出现0xEB，也常被用于混淆



```
.text:080487E8                 lea     eax, [ebp-70h]
.text:080487EB                 mov     [esp], eax
.text:080487EE                 call    _memset
.text:080487EE ; ---------------------------------------
.text:080487F3                 db 0EBh
.text:080487F4                 db 0FFh
.text:080487F5                 db 0C0h
.text:080487F6                 db  48h ; H
.text:080487F7                 db 0C7h
.text:080487F8                 db  44h ; D
.text:080487F9                 db  24h ; $
.text:080487FA                 db    8
.text:080487FB                 db  14h
.text:080487FC                 db    0
```

（6）将0xEBpatch为0x90，汇编代码恢复为nop，但下方又出现一出标红。但是标红代码的上两行代码是无条件跳转指令，所以没有影响。

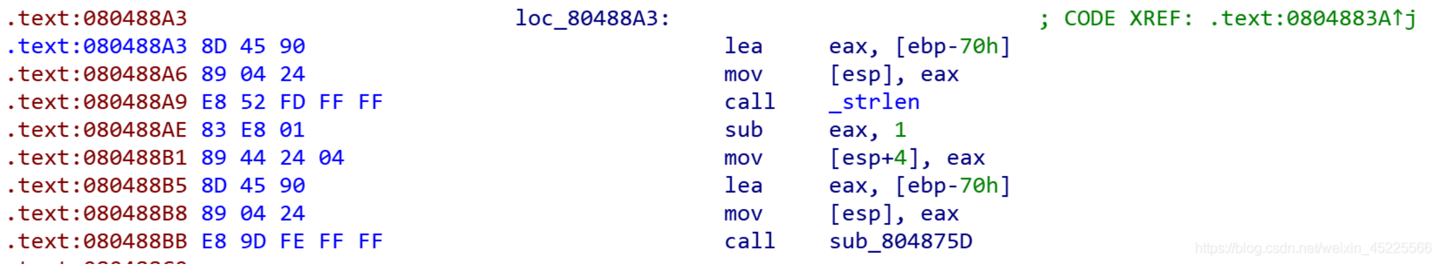

IDA - echo-server D:\课程学习\大三上\RE\攻防世界\echo-server
文件(F) 编辑(E) 跳转(J) 搜索(H) 视图(V) 调试器 选项(O) 窗口(W) 帮助
无调试器
库函数 常规函数 指令 数据 未知 外部符号
函数窗口
函数名称 段
IDA View-A
十六进制视图-1
结构体
枚举
行 21/40
输出窗口
Python

```
.text:080487C2                 leave
.text:080487C3                 retn
.text:080487C3 ; } // starts at 804875D
.text:080487C4 ; ---------------------------------------------------------------------------
.text:080487C4
.text:080487C4 loc_80487C4:                            ; CODE XREF: main+49↓p
.text:080487C4 ; __unwind {
.text:080487C4                 push    ebp
.text:080487C5                 mov     ebp, esp
.text:080487C7                 sub     esp, 88h
.text:080487CD                 mov     eax, large gs:14h
.text:080487D3                 mov     [ebp-0Ch], eax
.text:080487D6                 xor     eax, eax
.text:080487D8                 mov     dword ptr [esp+8], 14h
.text:080487E0                 mov     dword ptr [esp+4], 0
.text:080487E8                 lea     eax, [ebp-70h]
.text:080487EB                 mov     [esp], eax
.text:080487EE                 call    _memset
.text:080487F3                 nop
.text:080487F4                 inc     eax
.text:080487F6                 dec     eax
.text:080487F7                 mov     dword ptr [esp+8], 14h
.text:080487FF                 lea     eax, [ebp-70h]
.text:08048802                 mov     [esp+4], eax
.text:08048806                 mov     dword ptr [esp], 0
.text:0804880D                 call    _read
.text:08048812                 xor     eax, eax
.text:08048814                 jz      short loc_804881D
.text:08048816                 jmp     near ptr 6F44B961h
.text:08048816 ; ---------------------------------------------------------------------------
.text:0804881B                 db  41h ; A
.text:0804881C                 db    0
.text:0804881D ; ---------------------------------------------------------------------------
.text:0804881D
.text:0804881D loc_804881D:                            ; CODE XREF: .text:08048814↑j
.text:0804881D                 mov     dword ptr [esp+8], 5
.text:08048825                 mov     dword ptr [esp+4], 8048817h
.text:0804882D                 lea     eax, [ebp-70h]
.text:08048830                 mov     [esp], eax
.text:08048833                 call    _strncmp
.text:08048838                 test    eax, eax
.text:0804883A                 jnz     short loc_80488A3
.text:0804883C                 mov     dword ptr [esp], offset aYouAreVeryClos ; "You are very close! Now patch me~"
000007F3 080487F3: .text:080487F3
```

```
IDAPython v1.7.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
-------------------------------------------------------------------------------------
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
```

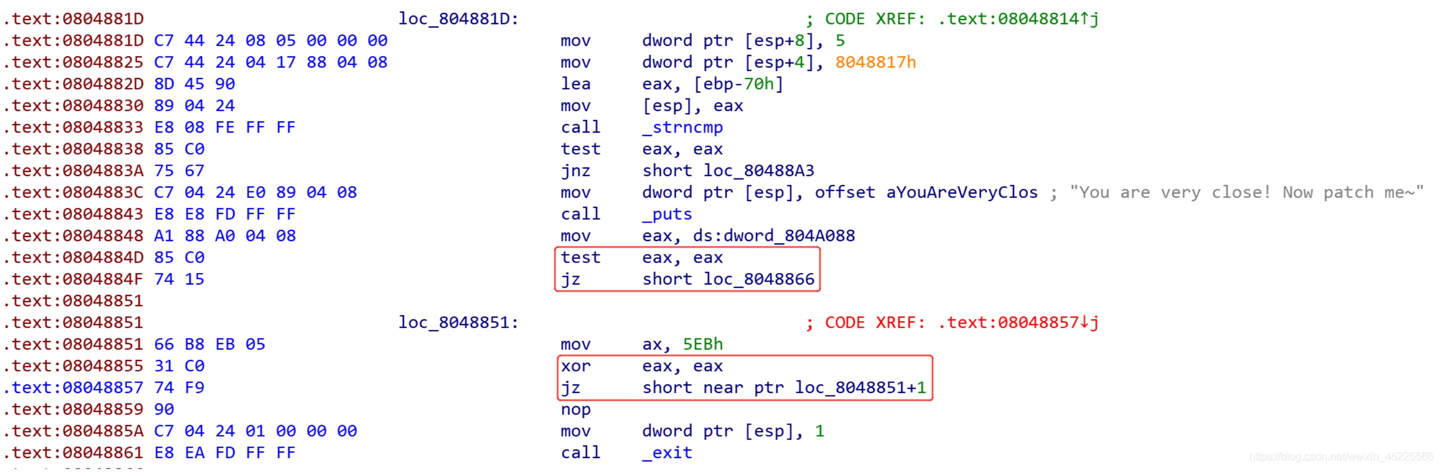
```
AU:  idle   向下   磁盘: 135GB
```

（7）继续向下运行，eax被赋值为5，然后跳进loc_80488A3，进去后发现每次eax会减一，因此这是个循环，读输入的前5个字符



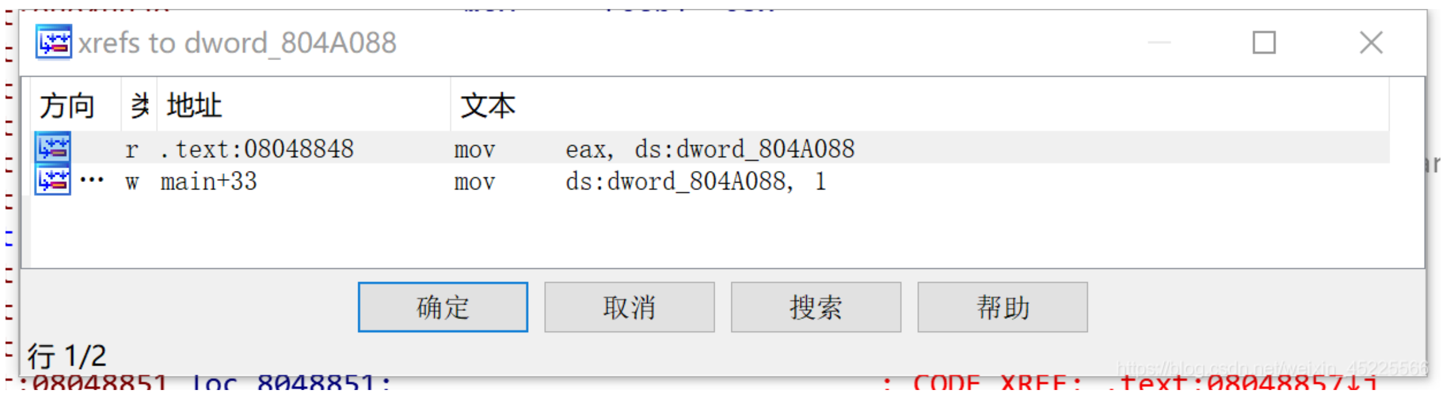
```
.text:080488A3                          loc_80488A3:                          ; CODE XREF: .text:0804883A↑j
.text:080488A3 8D 45 90                                 lea     eax, [ebp-70h]
.text:080488A6 89 04 24                                 mov     [esp], eax
.text:080488A9 E8 52 FD FF FF                           call    _strlen
.text:080488AE 83 E8 01                                 sub     eax, 1
.text:080488B1 89 44 24 04                              mov     [esp+4], eax
.text:080488B5 8D 45 90                                 lea     eax, [ebp-70h]
.text:080488B8 89 04 24                                 mov     [esp], eax
.text:080488BB E8 9D FE FF FF                           call    sub_804875D
```

（8）回去将下面的标红代码改为nop，发现如果0x0804884F行不执行，函数就会陷入死循环。
xor eax, eax的结果一直为0，jz结果为0时跳转



```
.text:0804881D                          loc_804881D:                          ; CODE XREF: .text:08048814↑j
.text:0804881D C7 44 24 08 05 00 00 00                  mov     dword ptr [esp+8], 5
.text:08048825 C7 44 24 04 17 88 04 08                  mov     dword ptr [esp+4], 8048817h
.text:0804882D 8D 45 90                                 lea     eax, [ebp-70h]
.text:08048830 89 04 24                                 mov     [esp], eax
.text:08048833 E8 08 FE FF FF                           call    _strncmp
.text:08048838 85 C0                                    test    eax, eax
.text:0804883A 75 67                                    jnz     short loc_80488A3
.text:0804883C C7 04 24 E0 89 04 08                     mov     dword ptr [esp], offset aYouAreVeryClos ; "You are very close! Now patch me~"
.text:08048843 E8 E8 FD FF FF                           call    _puts
.text:08048848 A1 88 A0 04 08                           mov     eax, ds:dword_804A088
.text:0804884D 85 C0                                    test    eax, eax
.text:0804884F 74 15                                    jz      short loc_8048866
.text:08048851
.text:08048851                          loc_8048851:                          ; CODE XREF: .text:08048857↓j
.text:08048851 66 B8 EB 05                              mov     ax, 5EBh
.text:08048855 31 C0                                    xor     eax, eax
.text:08048857 74 F9                                    jz      short near ptr loc_8048851+1
.text:08048859 90                                       nop
.text:0804885A C7 04 24 01 00 00 00                     mov     dword ptr [esp], 1
.text:08048861 E8 EA FD FF FF                           call    _exit
```

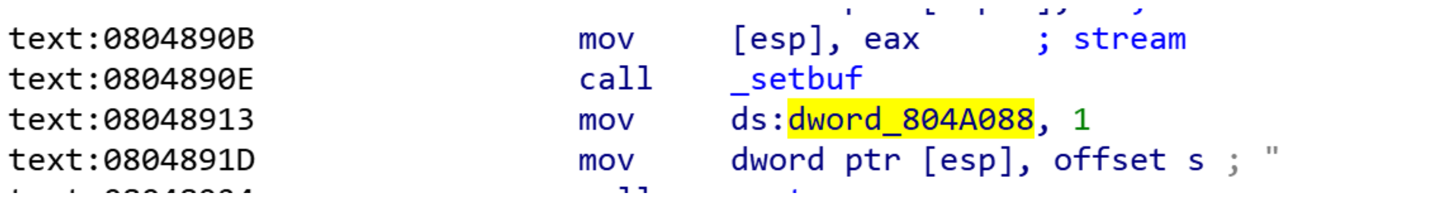
（9）eax的值是根据dword_804A088的值来决定，按x查看交叉引用，看下dword_804A088这个地址的值是哪来的



xrefs to dword_804A088

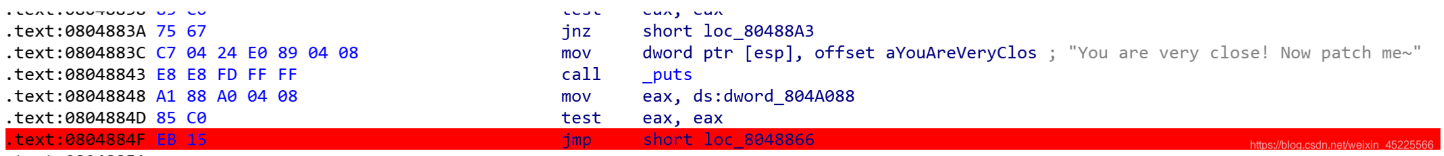
| 方向 | 类 | 地址 | 文本 |
|---|---|---|---|
|  | r | .text:08048848 | mov eax, ds:dword_804A088 |
| … | w | main+33 | mov ds:dword_804A088, 1 |

（10）只有一个地方，跟过去看下，发现dword_804A088这个地址的值是在main函数中硬编码的1



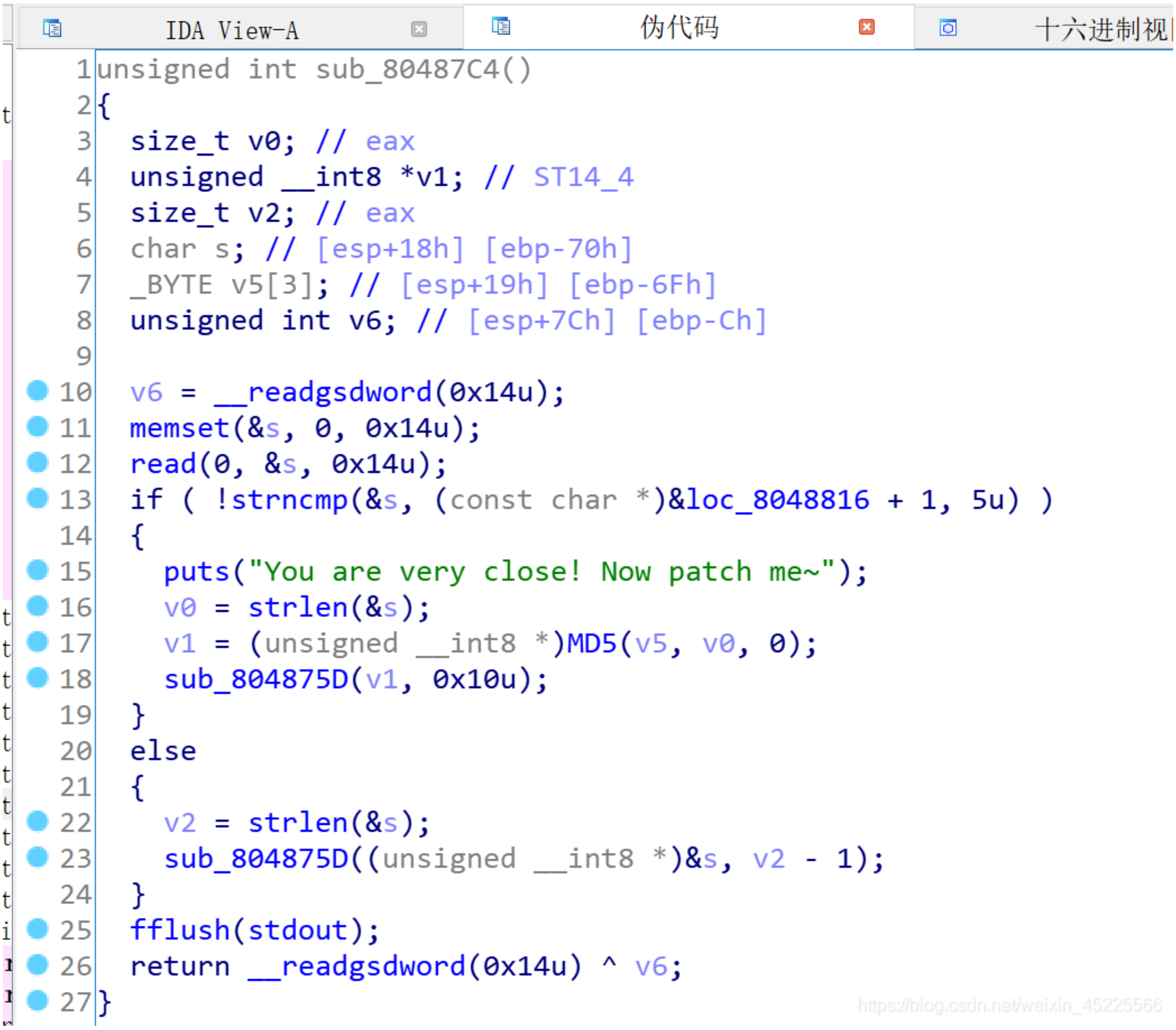
```
text:0804890B          mov     [esp], eax      ; stream
text:0804890E          call    _setbuf
text:08048913          mov     ds:dword_804A088, 1
text:0804891D          mov     dword ptr [esp], offset s ; "
```

（11）由此可知eax的值为1，需要把jz指令patch为jmp
test逻辑与运算结果为零,就把ZF(零标志)置1;
test eax,eax
当eax=0时,置z标志位为1,jz 跳转,jnz 不跳转
当eax=1时,置z标志位为0,jz 不跳转,jnz 跳转



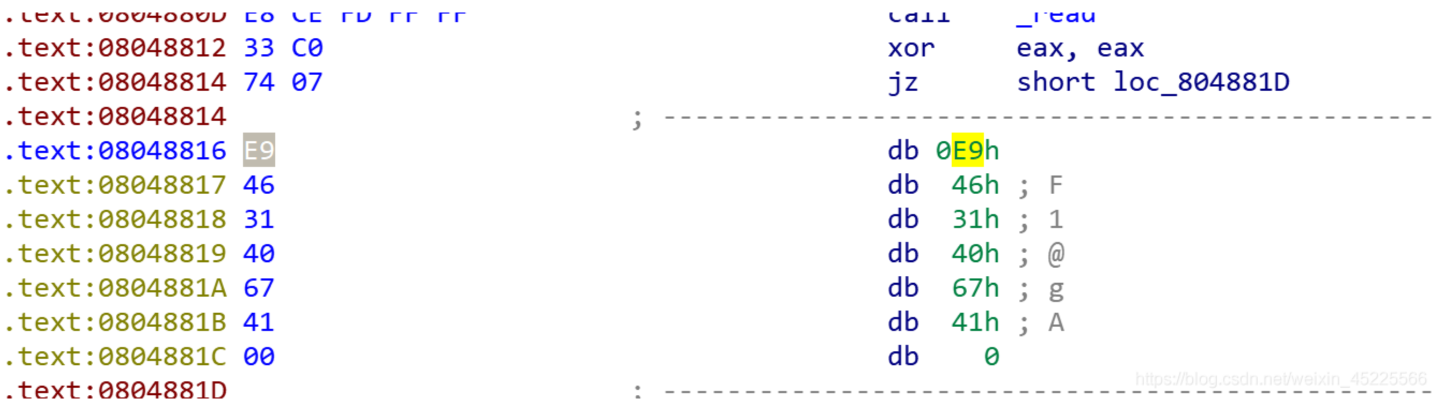
```
.text:0804883A 75 67                                    jnz     short loc_80488A3
.text:0804883C C7 04 24 E0 89 04 08                     mov     dword ptr [esp], offset aYouAreVeryClos ; "You are very close! Now patch me~"
.text:08048843 E8 E8 FD FF FF                           call    _puts
.text:08048848 A1 88 A0 04 08                           mov     eax, ds:dword_804A088
.text:0804884D 85 C0                                    test    eax, eax
.text:0804884F EB 15                                    jmp     short loc_8048866
```

（12）F5生成伪代码，可以看出就是比较输入的字符串的前五个字符与loc_8048816+1处是否相同



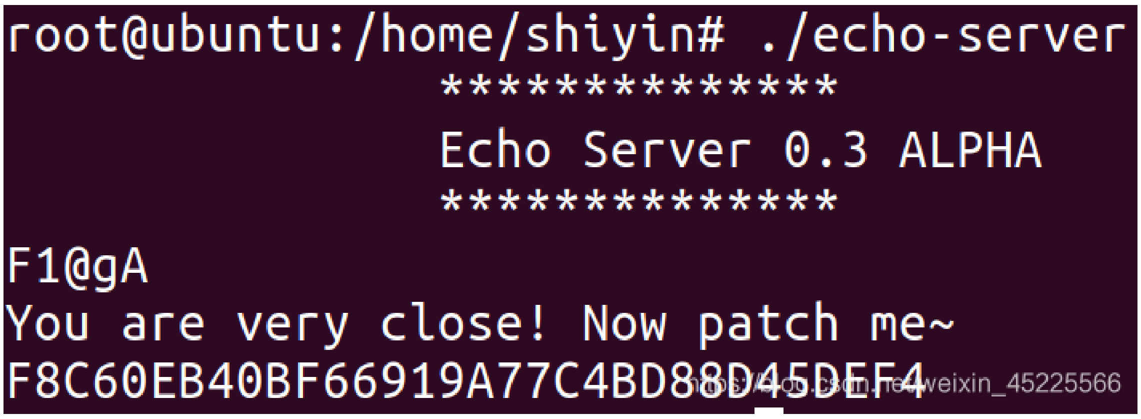
```c
unsigned int sub_80487C4()
{
  size_t v0; // eax
  unsigned __int8 *v1; // ST14_4
  size_t v2; // eax
  char s; // [esp+18h] [ebp-70h]
  _BYTE v5[3]; // [esp+19h] [ebp-6Fh]
  unsigned int v6; // [esp+7Ch] [ebp-Ch]

  v6 = __readgsdword(0x14u);
  memset(&s, 0, 0x14u);
  read(0, &s, 0x14u);
  if ( !strncmp(&s, (const char *)&loc_8048816 + 1, 5u) )
  {
    puts("You are very close! Now patch me~");
    v0 = strlen(&s);
    v1 = (unsigned __int8 *)MD5(v5, v0, 0);
    sub_804875D(v1, 0x10u);
  }
  else
  {
    v2 = strlen(&s);
    sub_804875D((unsigned __int8 *)&s, v2 - 1);
  }
  fflush(stdout);
  return __readgsdword(0x14u) ^ v6;
}
```

（13）回去发现是我们第（6）步没有管的花指令，按D转化成数据，发现是在代码段里存了一段数据。这里不要直接改机器码，改了之后会发现找不到字符串地址。



```
.text:08048812 33 C0                         xor     eax, eax
.text:08048814 74 07                         jz      short loc_804881D
.text:08048814                ; ---------------------------------------
.text:08048816 E9                            db 0E9h
.text:08048817 46                            db  46h ; F
.text:08048818 31                            db  31h ; 1
.text:08048819 40                            db  40h ; @
.text:0804881A 67                            db  67h ; g
.text:0804881B 41                            db  41h ; A
.text:0804881C 00                            db    0
.text:0804881D                ; ---------------------------------------
```

（14）点击编辑->修补文件->修补程序应用到输入文件，一定要保存！！！不然前面白干了。然后运行程序，输入字符串“F1@gA”，得到flag。



```
root@ubuntu:/home/shiyin# ./echo-server
                              ***************
                              Echo Server 0.3 ALPHA
                              ***************
F1@gA
You are very close! Now patch me~
F8C60EB40BF66919A77C4BD88D45DEF4
```

（15）运行报错解决

如果报错 libcrypto.so.1.0.0: cannot open shared object file: No such file or directory 是因为没有 libcrypto.so.1.0.0，我的系统有 64 位 libcrypto.so.1.0.0 但是没有 32 位的，因此需要安装 32 位的共享库。libssl 中带了 libcrypto。

命令行输入下面的指令即可

```
sudo apt-get install libssl1.0.0:i386
```

不过有的虚拟机可以，有的不行。我在kali输完还是不能运行，搞了一个多小时，试了网上各种方法都没用，换了个乌班图虚拟机这样就行了。有懂的童鞋可以告诉一下我。