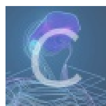
XCTF echo-server

原创

夏了茶糜 于 2020-03-05 23:00:42 发布 889 收藏 1

分类专栏： CTF-REVERSE 文章标签： 安全

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/qin9800/article/details/104684772
版权

 CTF-REVERSE 专栏收录该内容

18 篇文章 0 订阅

订阅专栏

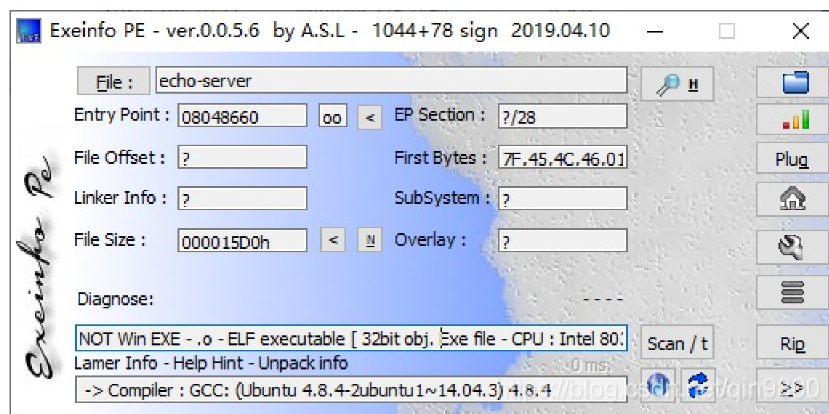
此题是XCTF 3rd-NJCTF-2017的题目

今天做攻防世界的逆向题目的时候遇到了，就在此写下解题过程

在此附上题目下载地址 echo-serverr

1.查壳

使用ExeinfoPe工具查壳，可以发现是32位程序，没有壳

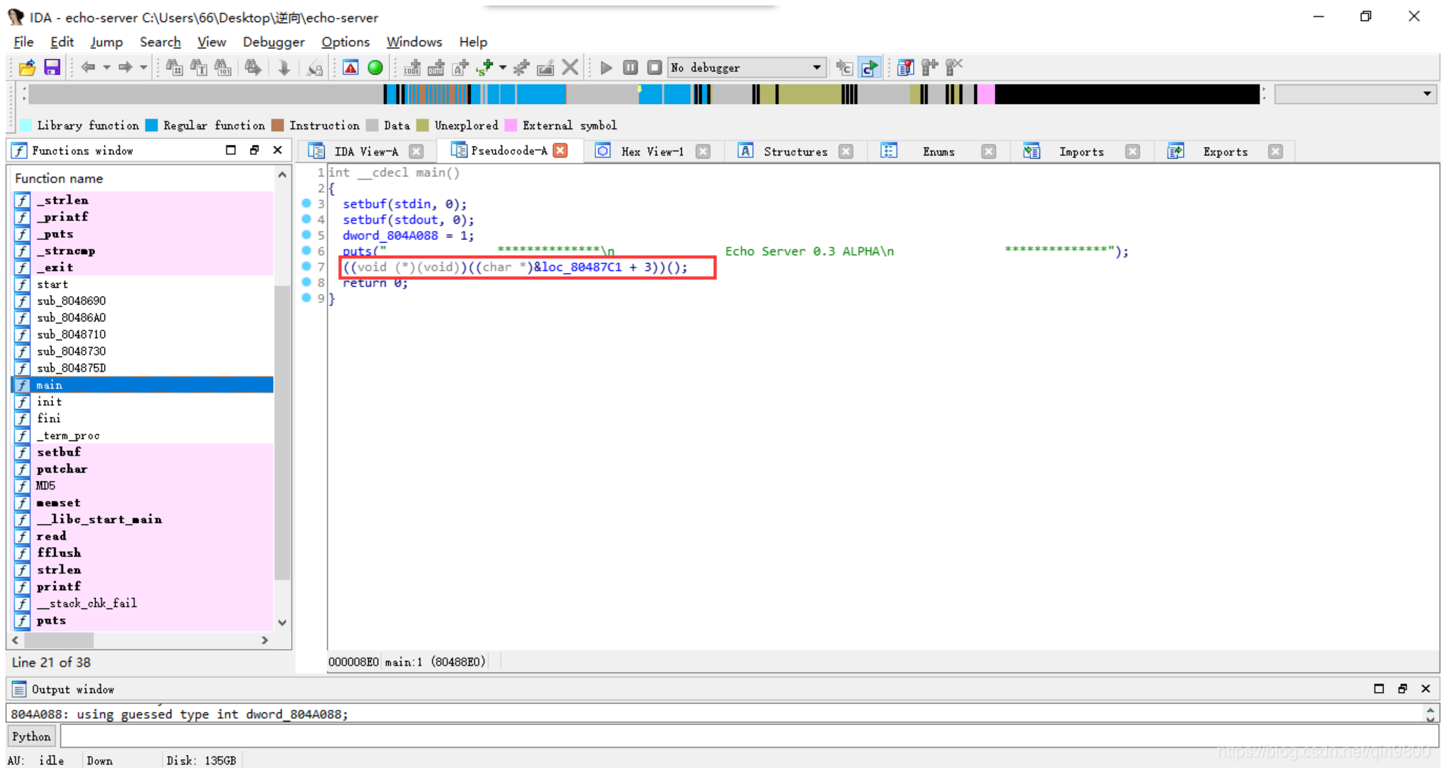


2.使用IDA进行反编译

使用32位的IDA打开程序

3.分析程序

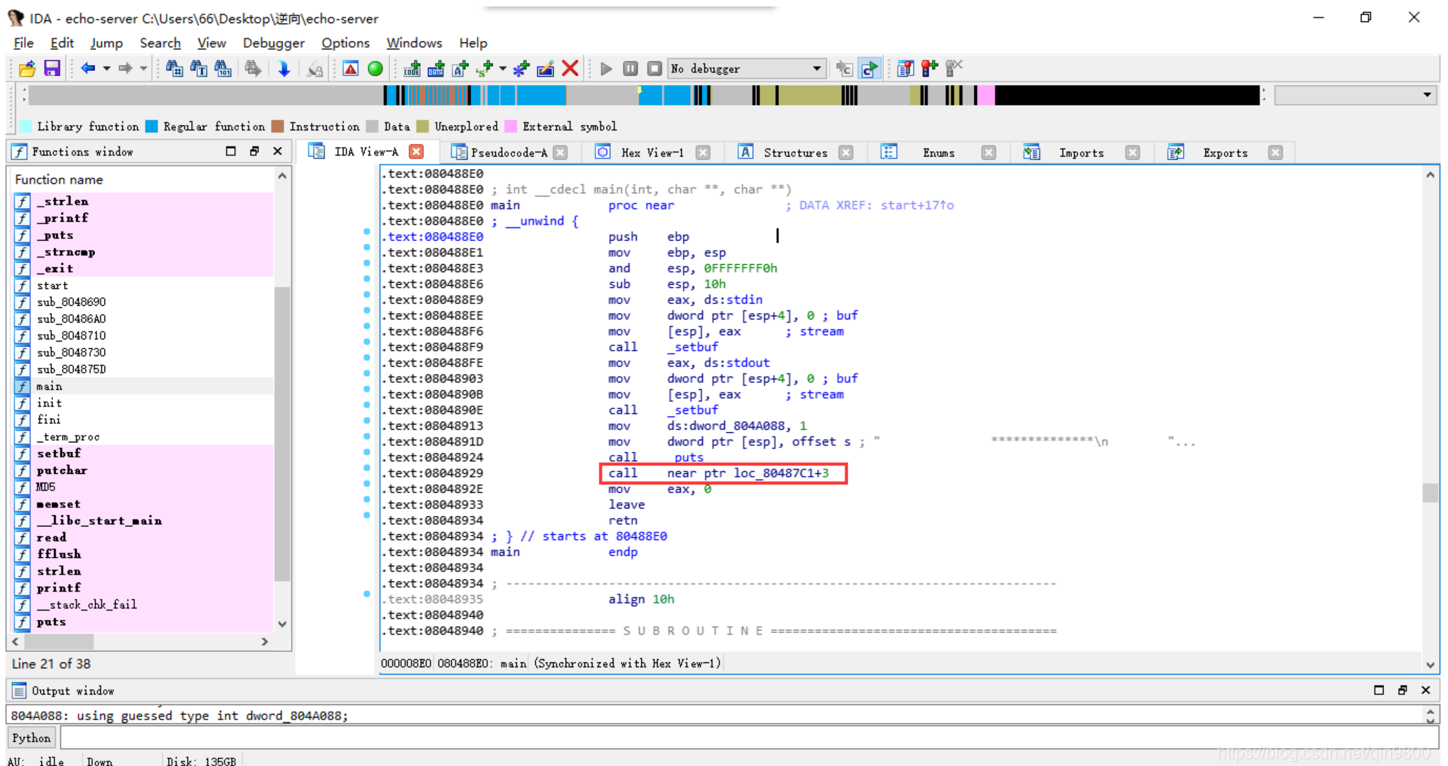
题目提示"输入密钥，得到flag."
定位到main函数，并使用F5大法查看伪代码



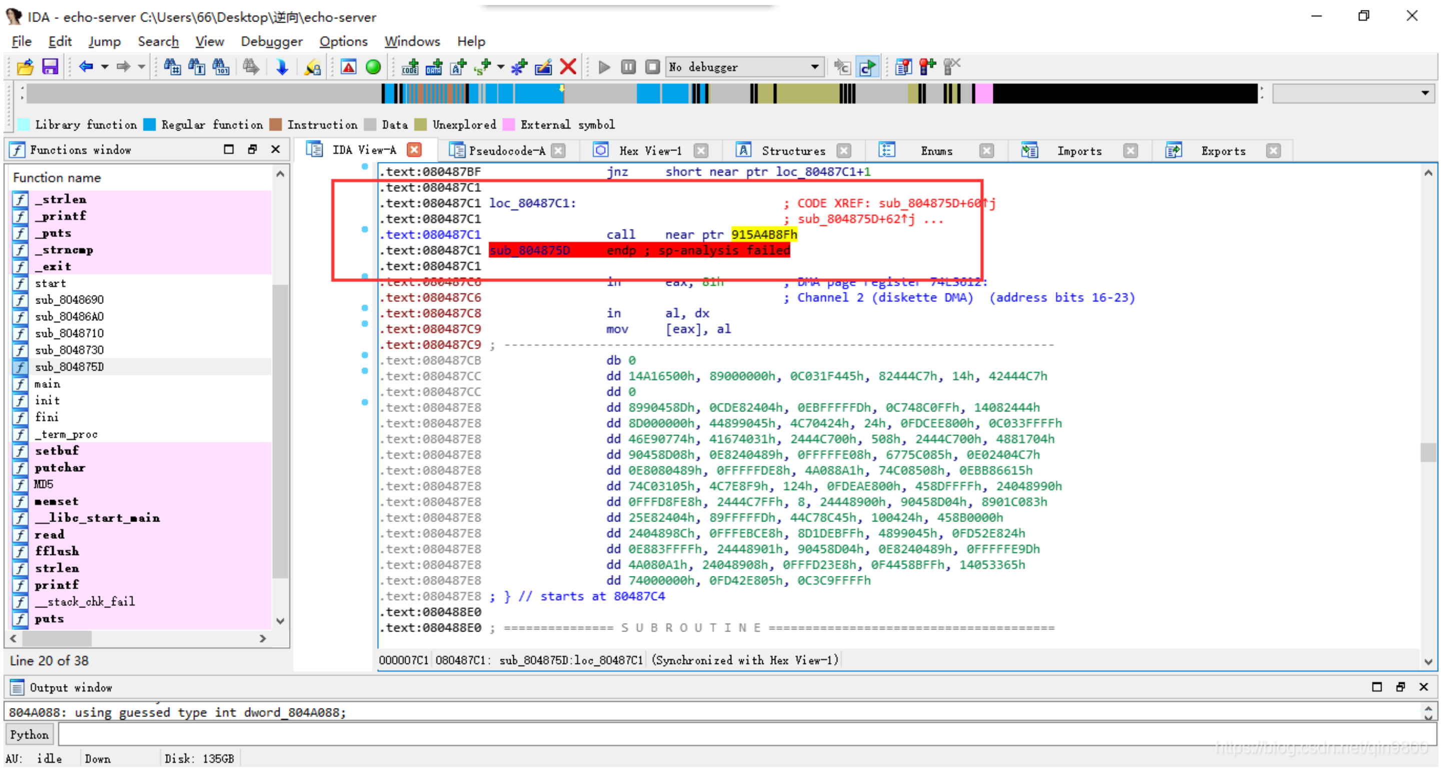
这个程序的main函数伪代码比较清晰，根据伪代码可知程序main函数中只有一个关键函数，但是这个函数调用比较奇怪。猜测这里可能有问题。

```
((void (*)(void))((char *)&loc_80487C1 + 3))();
```

我们切回汇编代码，直接看汇编代码

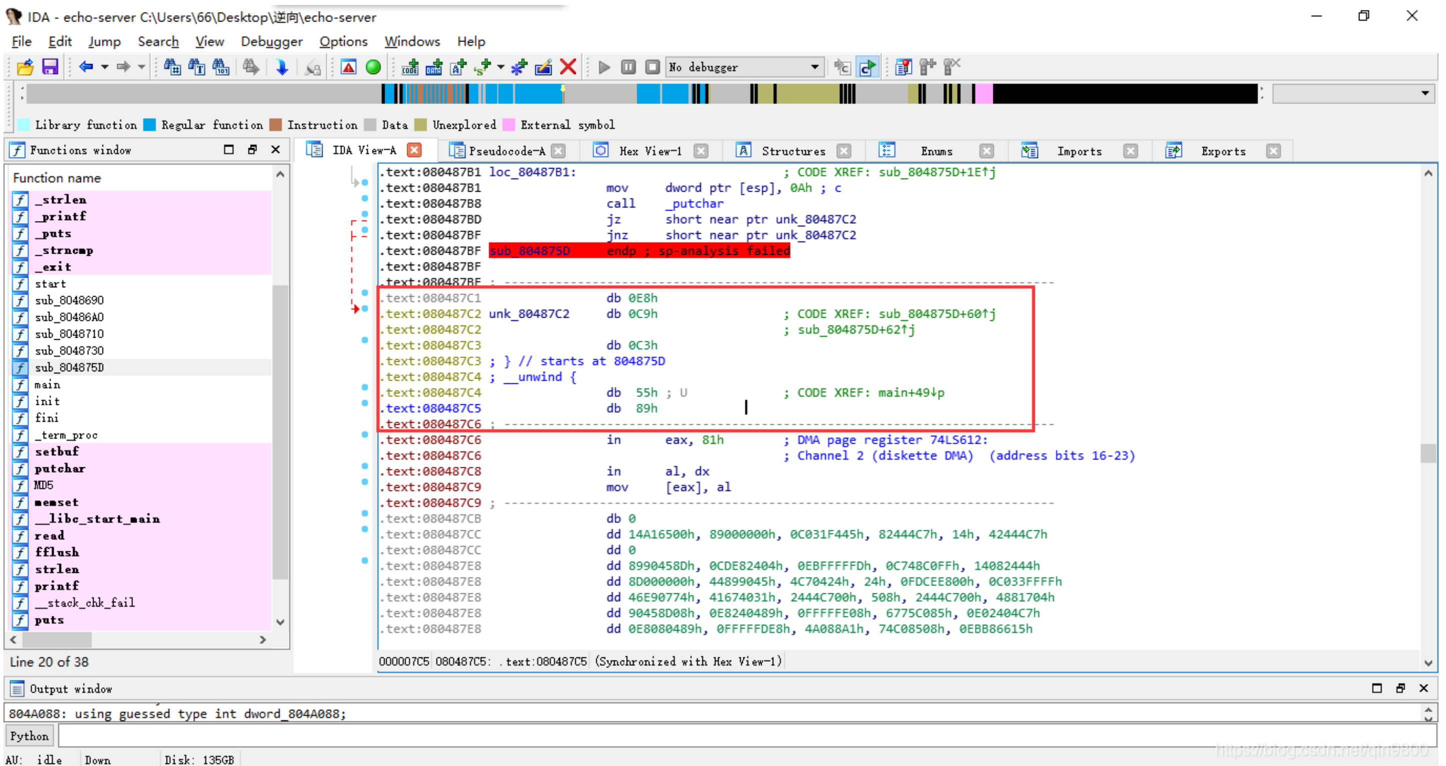


我们双击这个函数，跟去这个函数看下

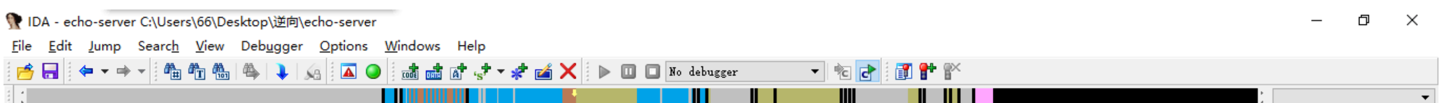


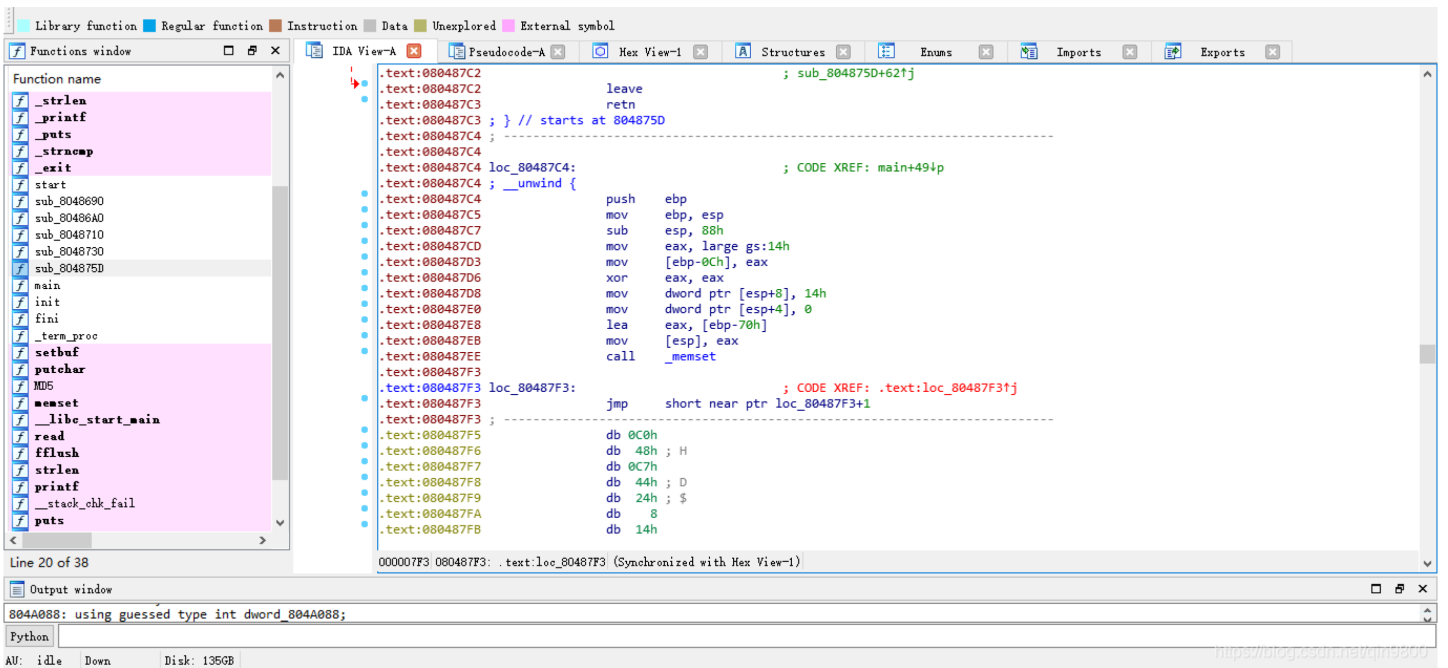
双击后程序跳转到这里了，根据main函数中的那个call，我们可以知道，程序应该跳转到`loc_80487C1+3`处，但是跳转过来后，发现汇编代码却乱七八糟的，得知程序被混淆了；我们需要想办法去除花指令。

在`0x080487C1`处按下D键把此处的汇编代码转换为数据，我在此处按了两次D键才转换完成。

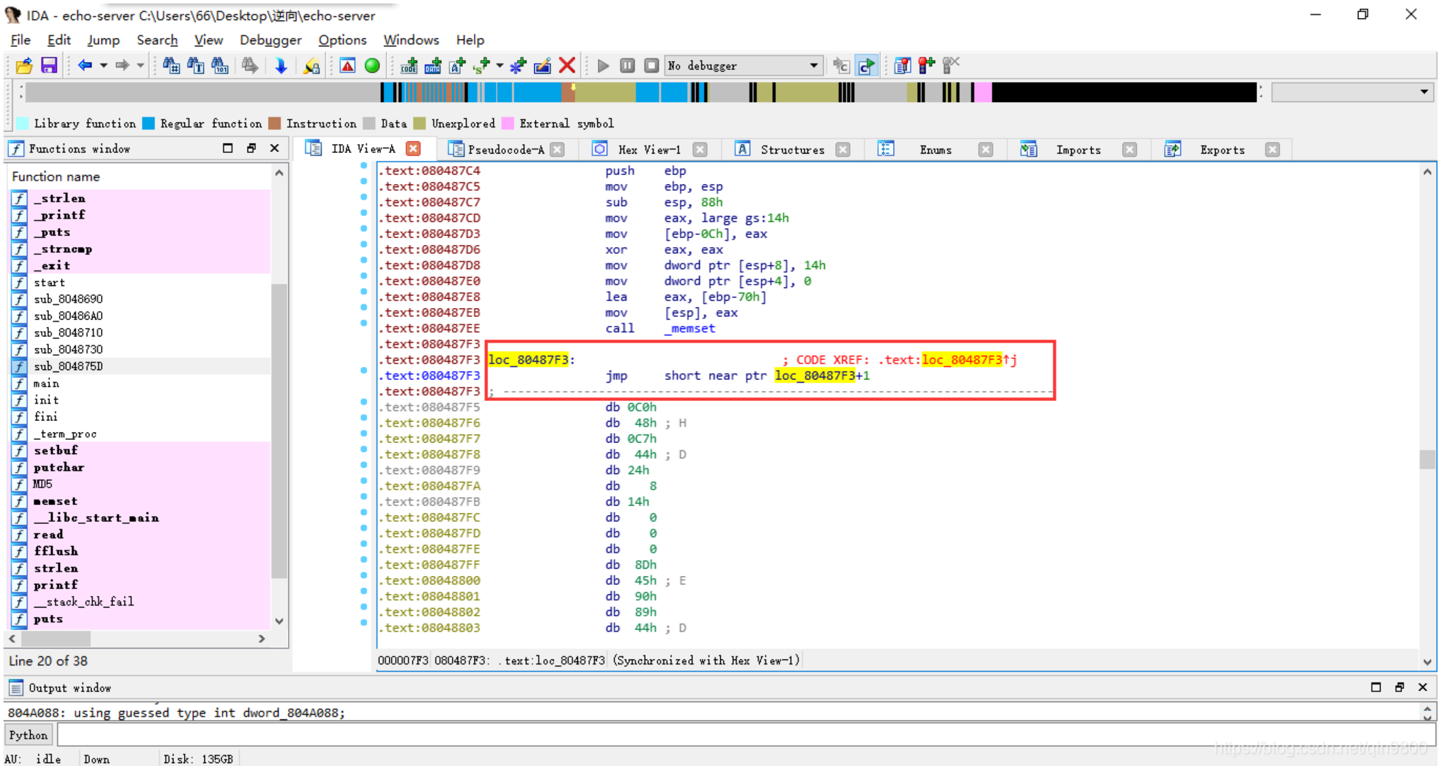


`0x80487C1`处应该变成这样，我们发现`0x80487C1`处的字节是0xE8，这个0xE8经常用于混淆，使用IDA把它patch为0x90，在`0x80487C2`处按下C键，我们会看到IDA分析`0x80487C2、0x80487C3`的汇编代码显示正常，我们返回main函数看一眼那个call，发现那个call变成了`call near ptr unk_80487C4`，我们回到`unk_80487C4`处，按下C键。会发现IDA分析的汇编代码和正常的函数开通差不多了。

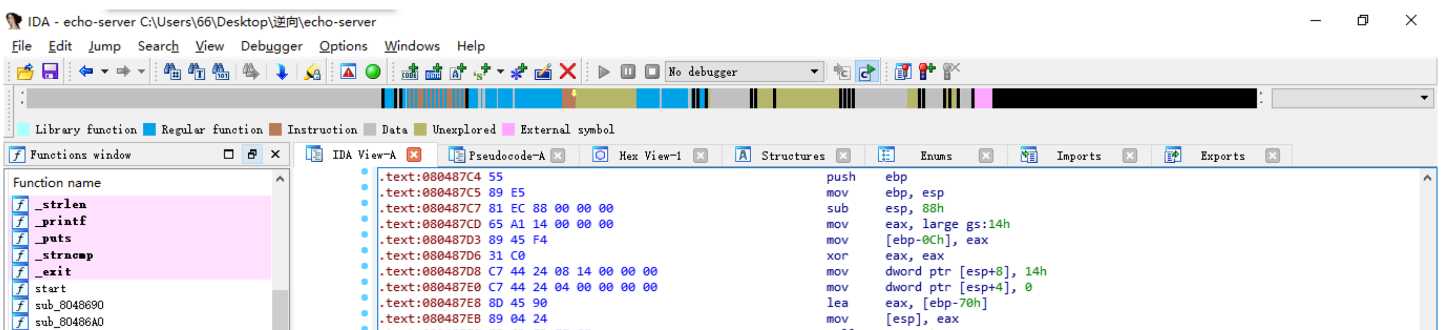


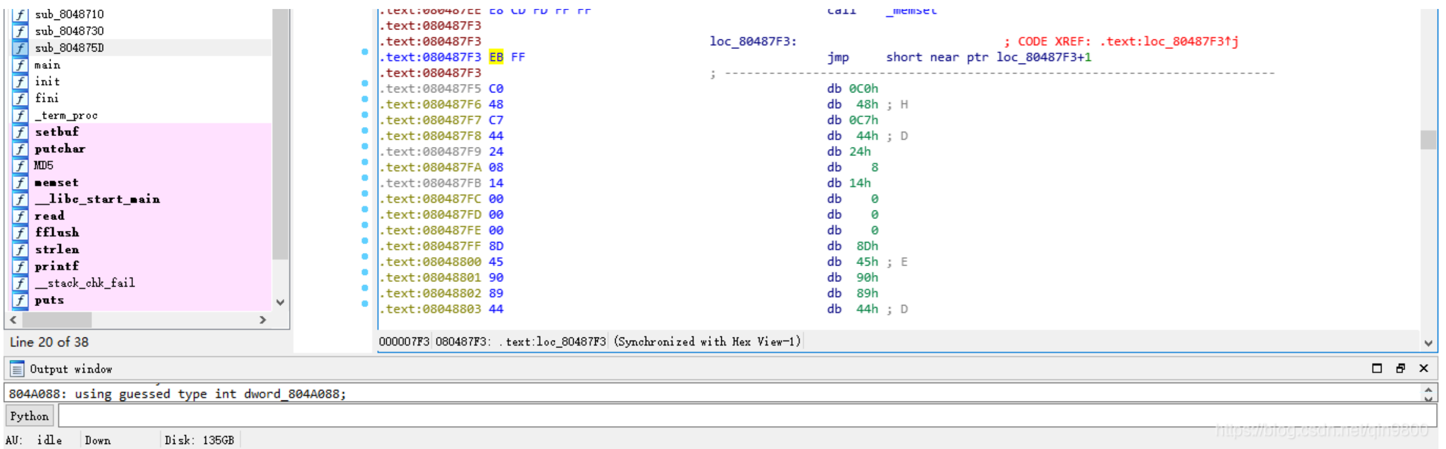


接着往下看

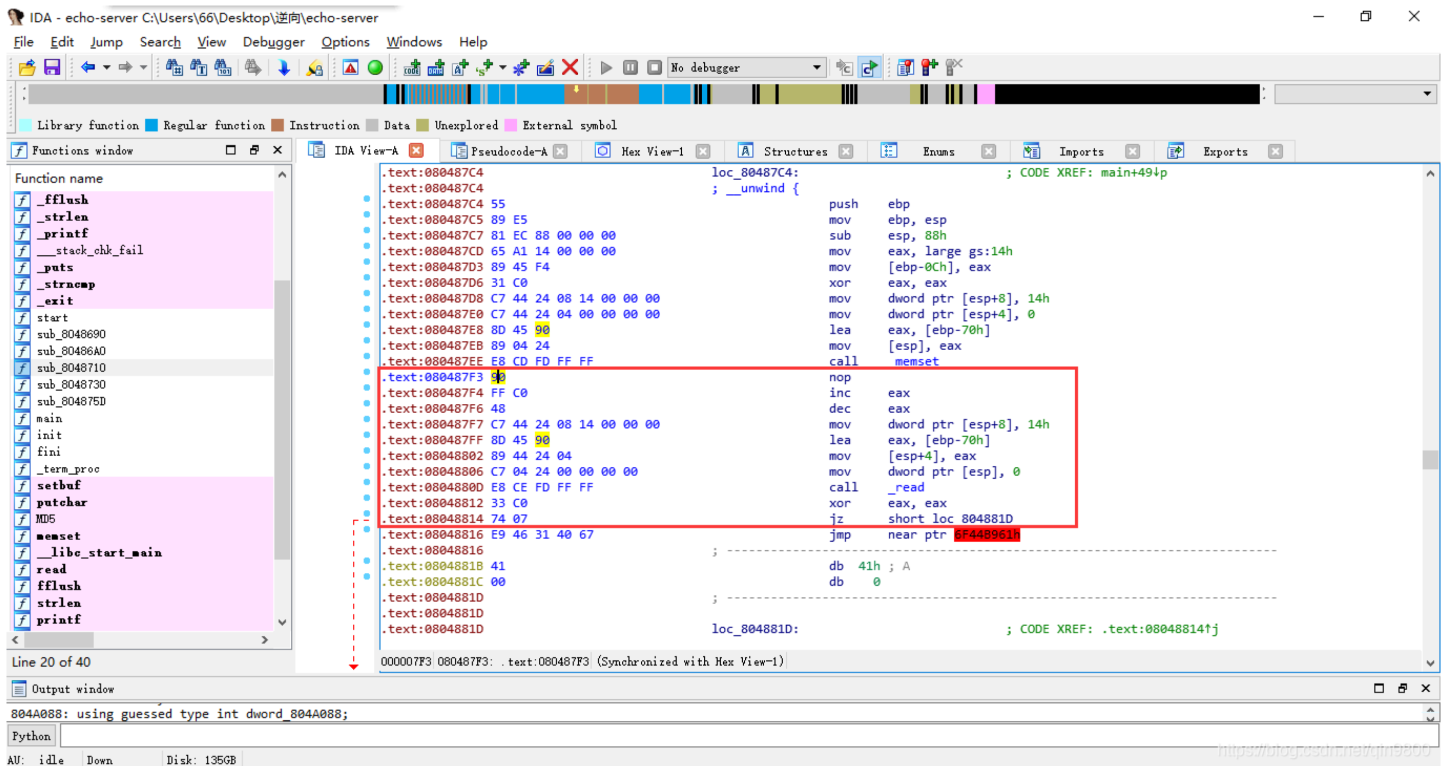


发现这里出现了一个和之前main函数中的函数调用一样诡异的jmp，`jmp short near ptr loc_80487F3+1` 我们可以仔细观察一些，可以发现这条指令占了两个字节，这条指令本身的意思就是跳到自己的第二个字节处，因此我们可以理解为这题指令就是相当于把第一个字节nop掉。



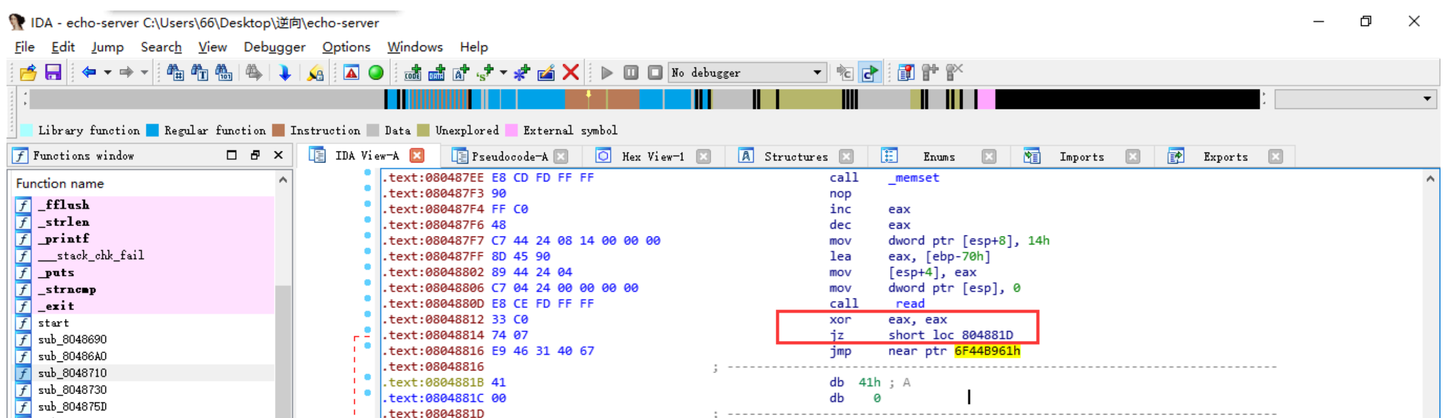


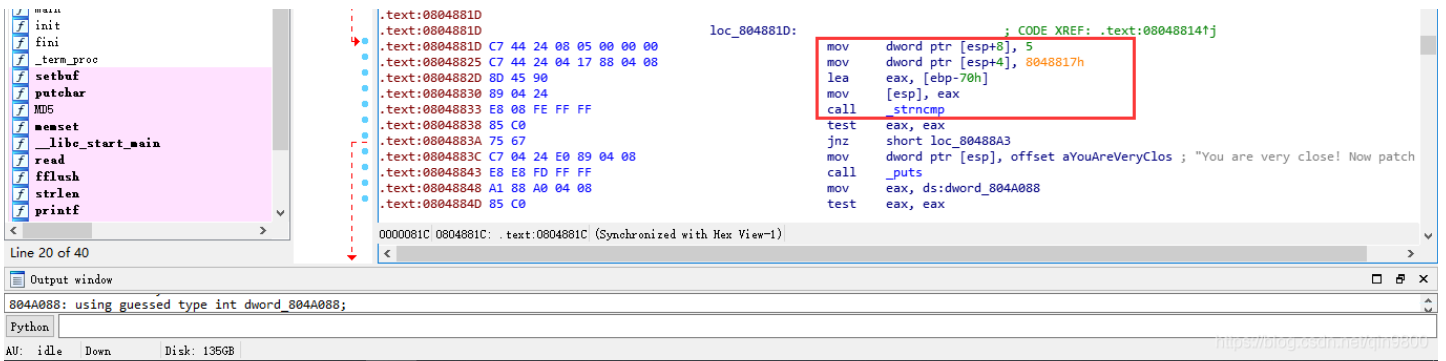
我门看下机器码，发现`0x80487F3`处的字节0xE8，0xE8也经常被用于混淆，再结合我们上一步的分析，可以得知这里就是用0xE8来干扰IDA的静态分析，我们直接把0xE8patch为0x90。



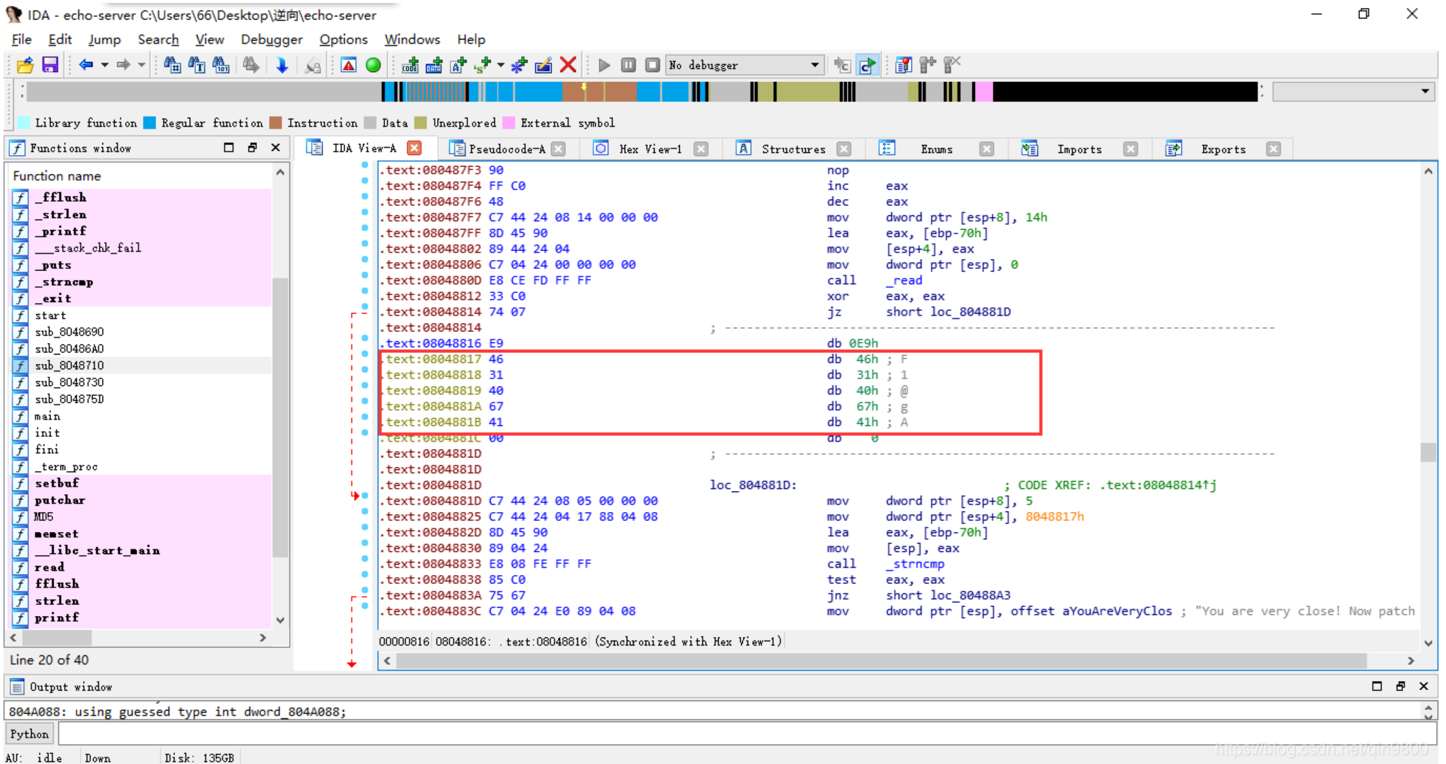
然后可以观察到，又有一块汇编代码恢复正常。然后我发现`0x08048816`处有一个地址被IDA标红。我们观察下发现`0x08048816`前面的两行汇编代码相当于无条件跳转

```
xor     eax, eax
jz      short loc_804881D
```

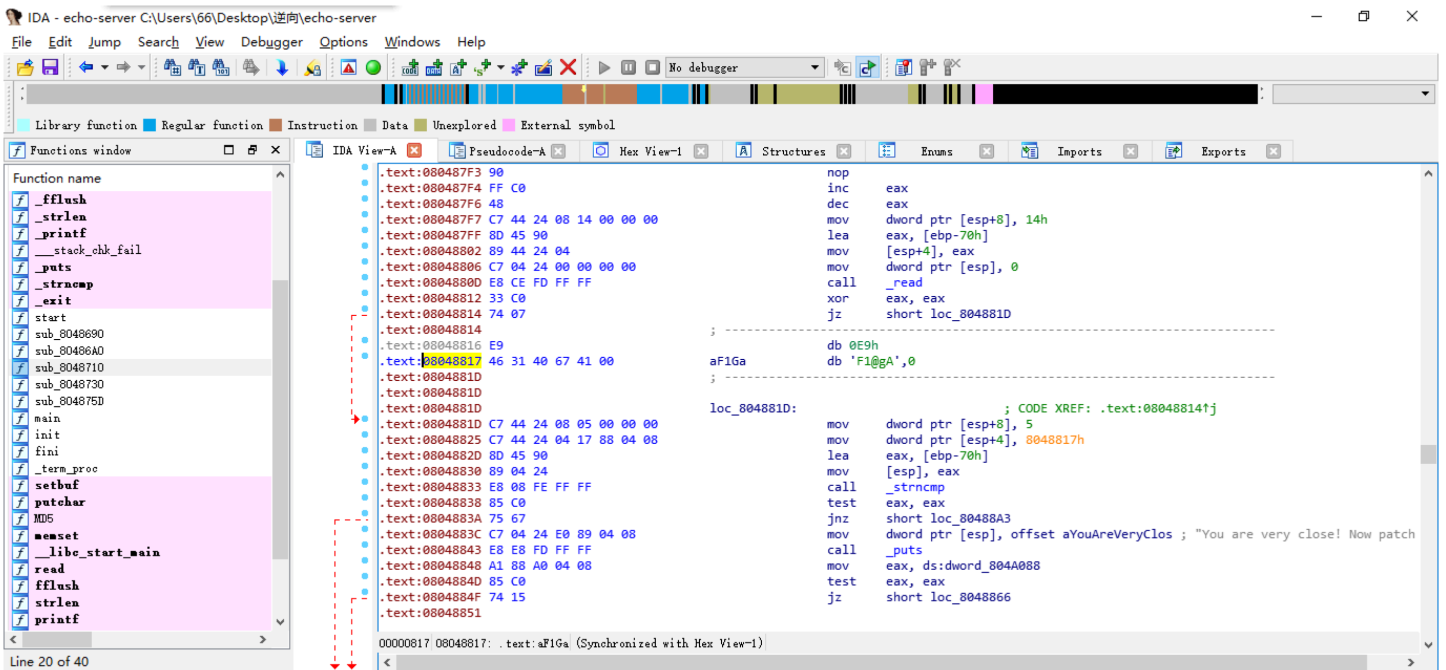


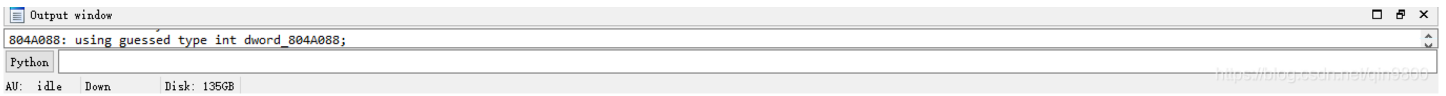


然后在`0x08048816`后面发现有一处比较，其中一个参数是地址是`0x8048817h`，因此我猜测此处的`0x08048816`处的几个字节的数据可能不是代码，而是数据，使用D转换为数据看下。



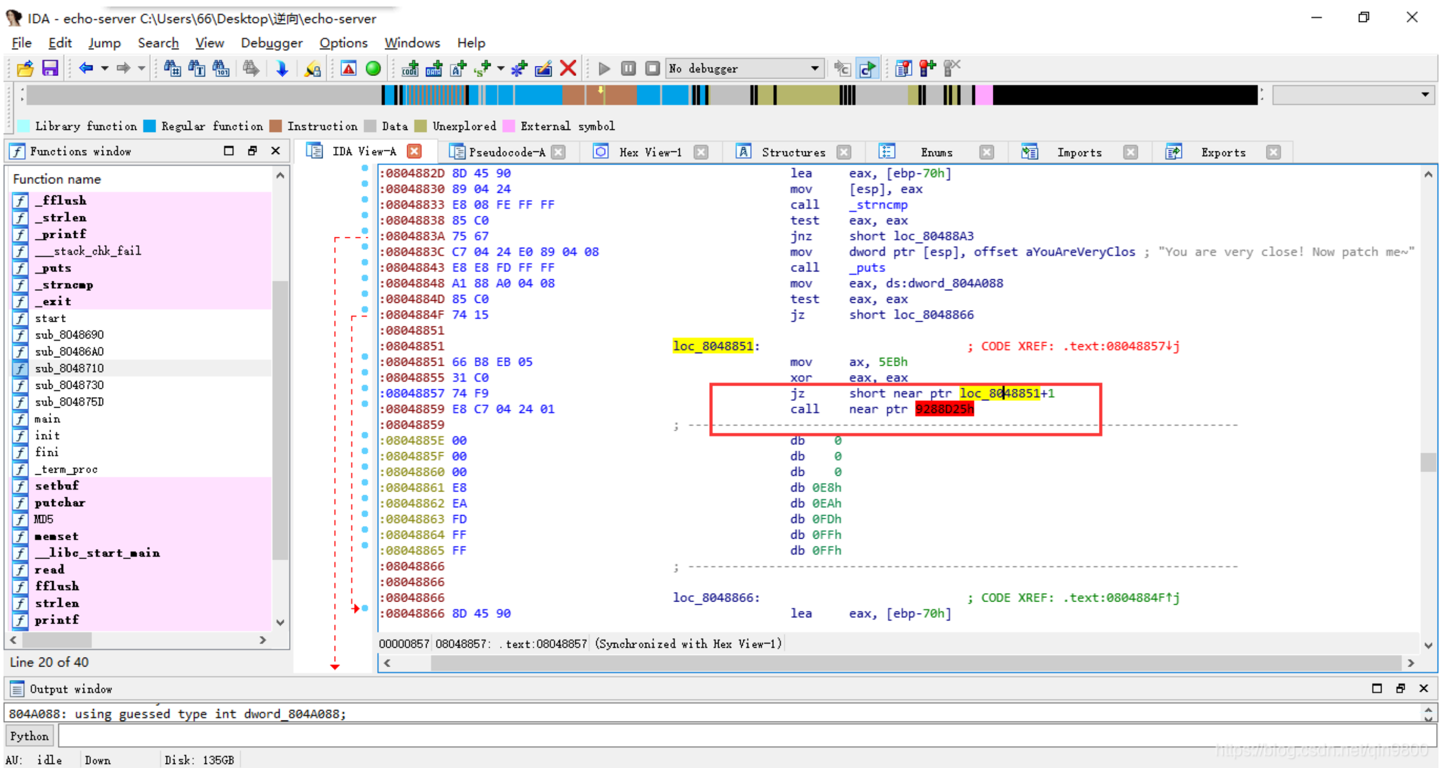
发现果然是数据，我们在`0x08048817`按A，把数据转为字符串



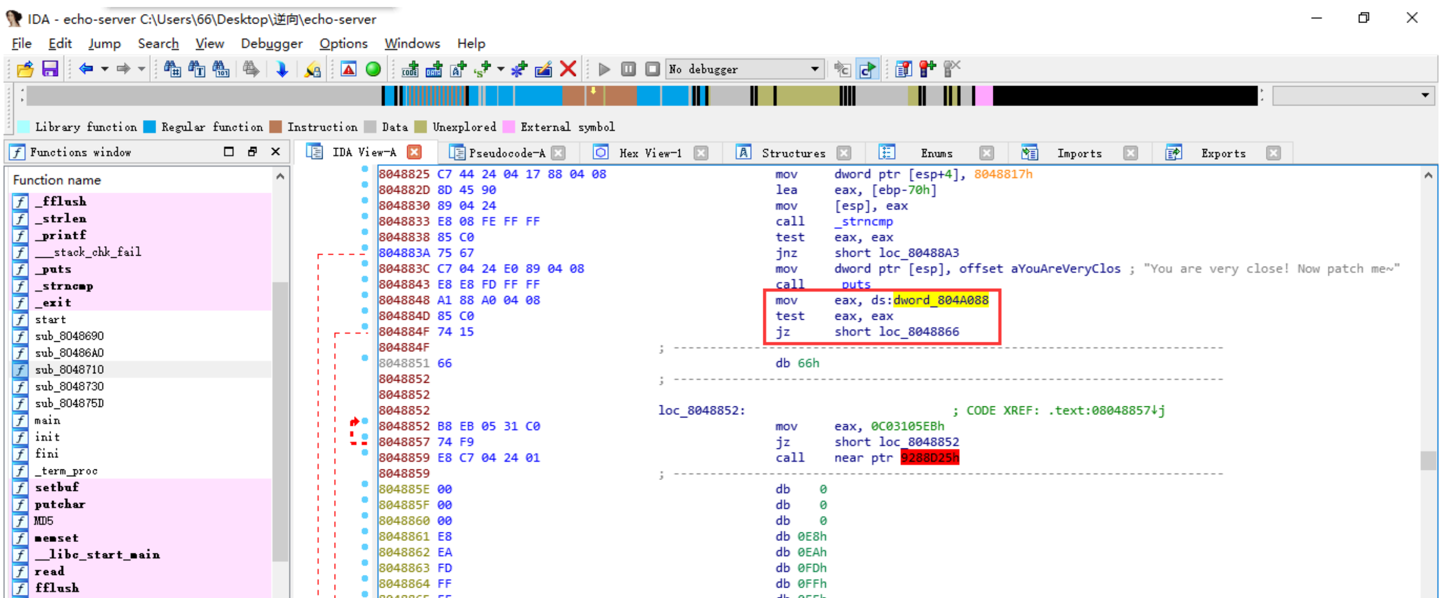


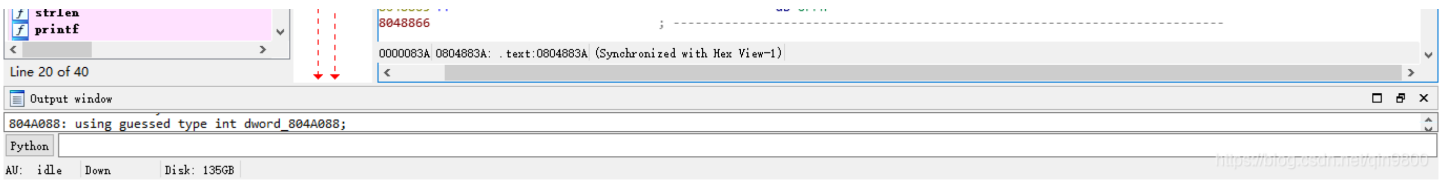
在字符串的下方有一处比较，还有一句"You are very close! Now patch me~"
因此我们得知程序需要用户输入的字符串是"F1@gA"，我们运行程序尝试下

发现确实通过比较了，但是并未输出flag，程序提示需要patch，我们接着往下看，找下程序为什么会卡在这。

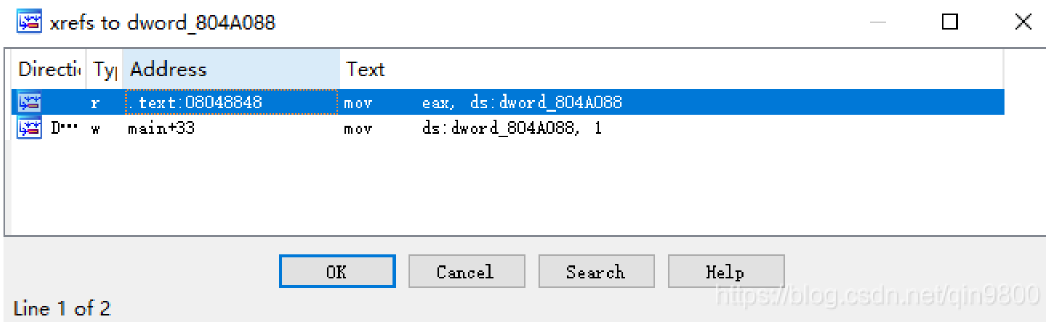


我们发现这里又有奇怪的跳转，我们使用之前的方法恢复下,尝试了下还是比较奇怪，先不管他，我有看到在"You are very close! Now patch me~"下方有个跳转

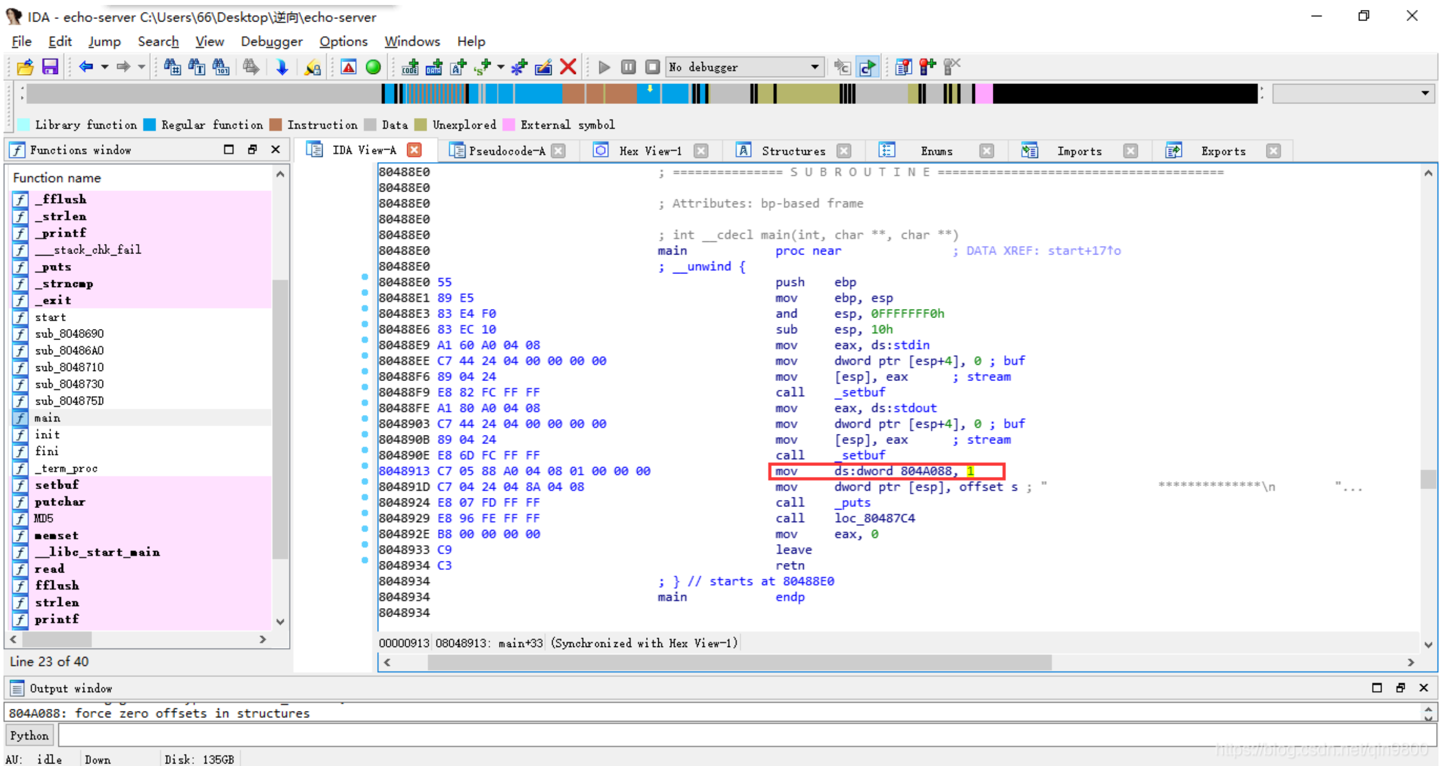




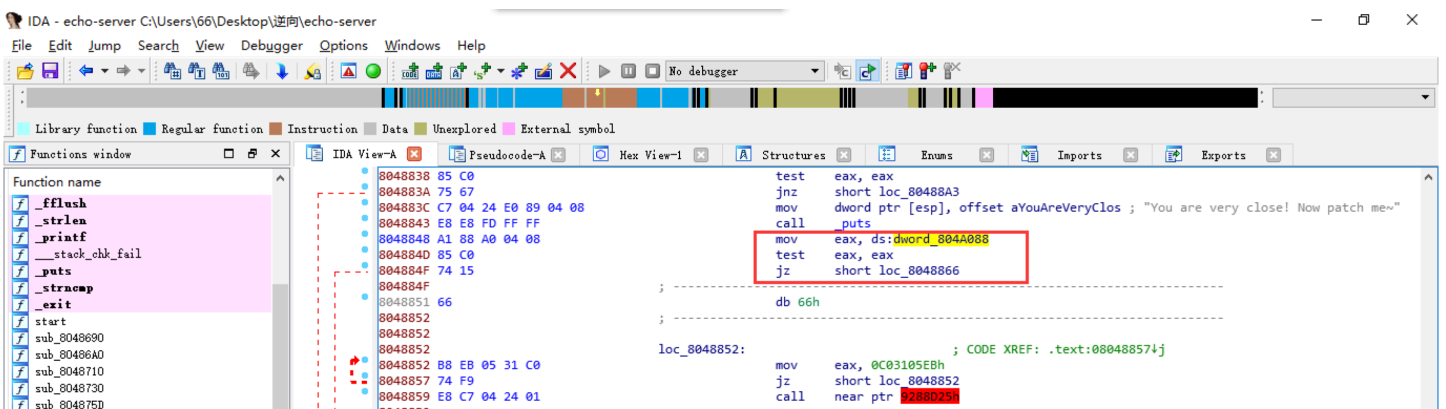
发现这里是根据`dword_804A088`的值来决定是否跳转的，在`dword_804A088`上按下X键，使用交叉引用，看下`dword_804A088`这个地址的值是哪来的，

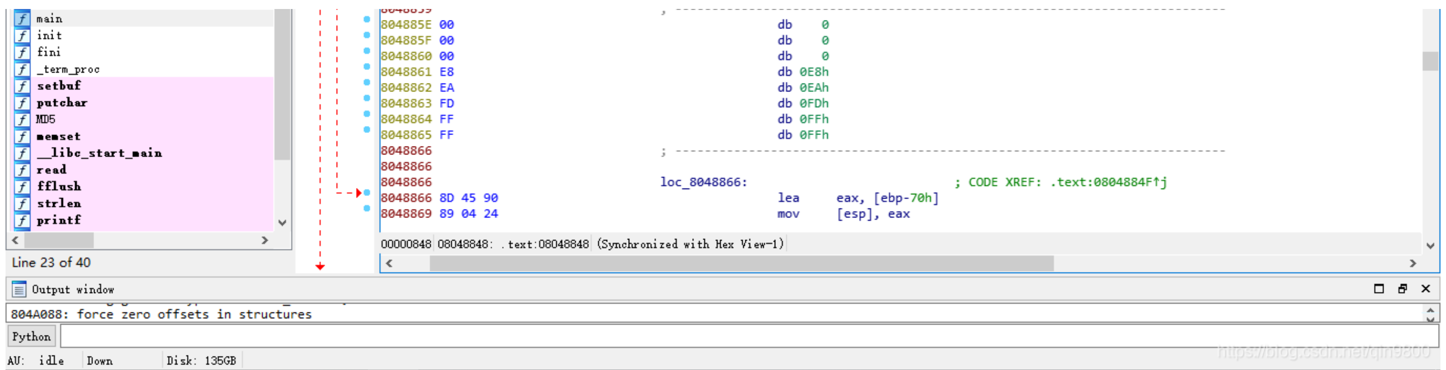


只有一个地方，跟过去看下

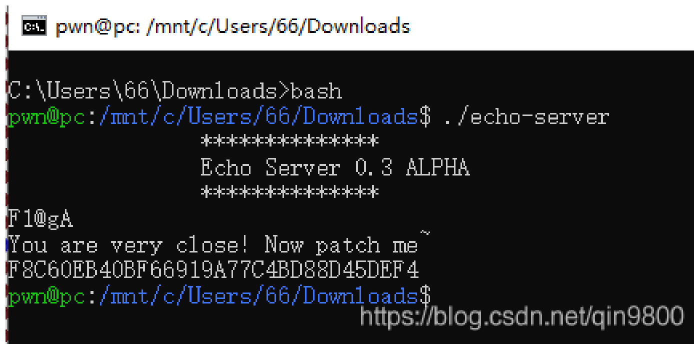


发现`dword_804A088`这个地址的值是在main函数中硬编码的1，





知道EAX的值为1，`test eax,eax` 置z标志位为0，z标志位为0时jz不跳转，所以此处的jz没有跳转
对此处进行patch，改为jmp，然后保存程序。



成功得到flag：F8C60EB40BF66919A77C4BD88D45DEF4

4.汇编知识复习

```
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

cmp eax,eax
cmp 算术减法运算结果为零,就把ZF(零标志)置1.
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1,jz 跳转,jnz 不跳转

test相当于两个参数做逻辑与运算;cmp相当与两个参数做减法运算.
如果结果为0就置z标志位为1;如果结果不为0就置z标志位为0;
当z标志位为1时,jz跳转,jnz不跳转;
当z标志位为0时,jz不跳转,jnz跳转;
jz:结果为0时跳转
jnz:结果不为0时跳转
```