第二题-wp

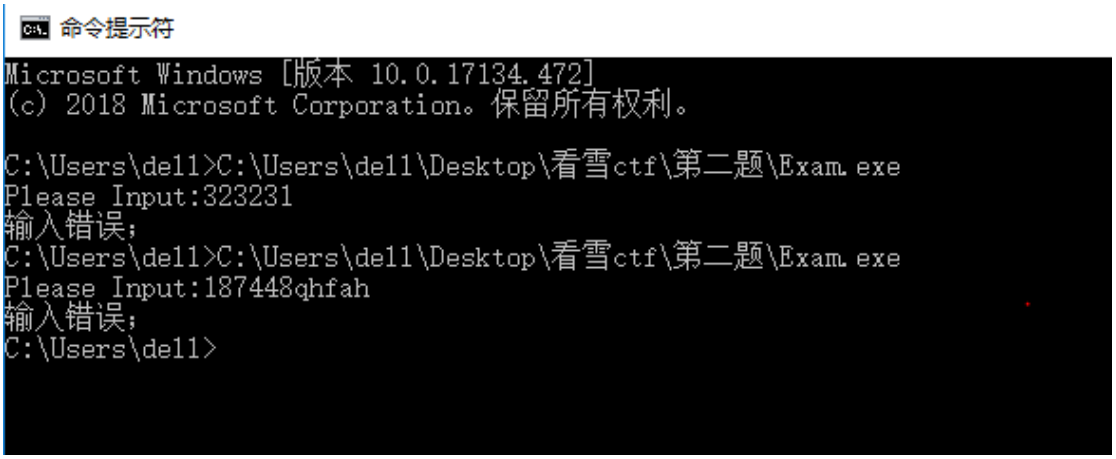
0x00

其实做了几道题才发现，逆向工程确实，跟它名字描述的一样，是一个逆向求解的过程，程序的执行是从上到下的过程，最后输出。而逆向工程是根据输出，找算法。
此篇题解根据看雪论坛诸位大佬的 writeup 所写，自己没做出来。

0x01

首先运行程序，任意输入，发现闪退，emmmm，

百度得到解决方法，使用 cmd，或者 powershell 运行



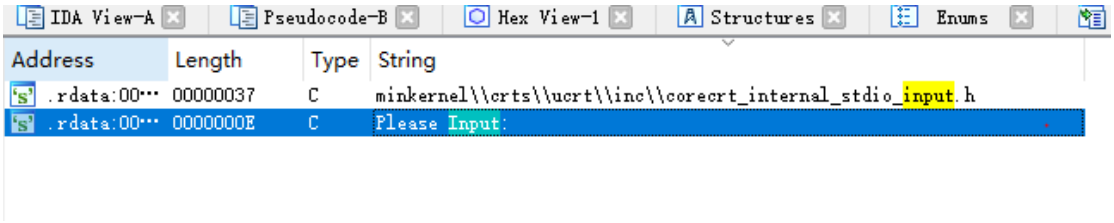
0x02

拖入 IDA，F5 失效，啥也出不来，找不到入口函数。

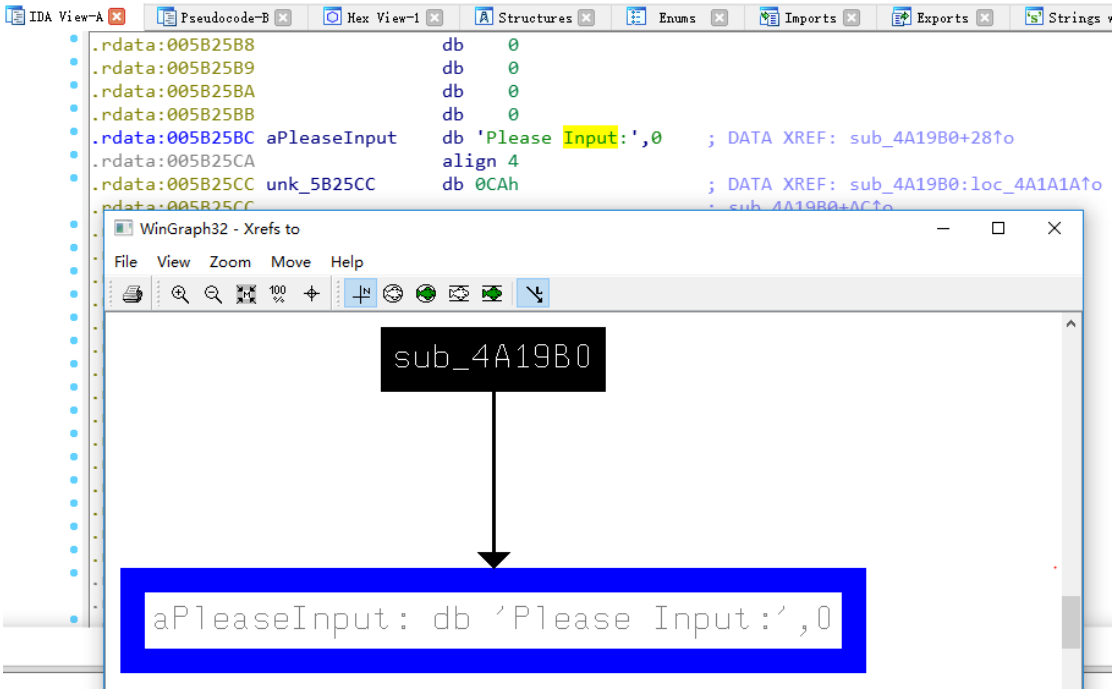
根据大佬 writeup，可以通过输出的“Please input”字符串查找入口函数。

它的原理根据我的理解就是，输出函数 printf 之类的这种函数执行之前，肯定会有函数对他进行调用。

于是通过 shift+12 查找字符串，“Please input”



点进去，查看哪个函数调用了它，使用右键——然后 xref



是 sub_4A19B0 函数，然后查看这个函数内容，双击函数，然后 F5

```
int __usercall sub_4A19B0@<eax>(int a1@<xmm0>)
{
  int v1; // edx
  int v2; // ecx
  int v4; // [esp+D0h] [ebp-8h]

  sub_48D7B4((int)&unk_5F6007);
  printf((int)&unk_5F31E0, (int)"Please Input:");
  scanf("%s", &input, 30);
  v4 = len((int)&input);
  if ( v4 > 30 || v4 < 10 )                    // 判断输入长度是不是>30,或者<10，如果是，直接退出
  {
    sub_48A6DB(a1, (int)&unk_5B25CC);
    exit(0);
  }
  sub_48E5BF(input1, 30, &input);              // 这是把输入，拷贝到dword中
  if ( *(_BYTE *)(input1 + 7) != 'A' )         // 第8个字符==ascii（A）
  {
    sub_48A6DB(a1, (int)&unk_5B25CC);          // 输入不正确
    exit(0);
  }
  sub_48D3A4(a1, input1);                      // 又是一次检验，下面看一下检验过程
  return sub_48D935(v2, v1, 1, 0, a1);
}
```

函数具体逻辑：v5 是输入字符串的长度，输入长度要在 10-30 之间然后将输入做拷贝至另一空间，我命名为 input1，然后使用 input1 进行计算检查，首先检查，第 8 位是不是 A，不是 A 也直接结束程序运行。随后的 sub_48D3A4 也是一个检验，跟进去。需要跟两次

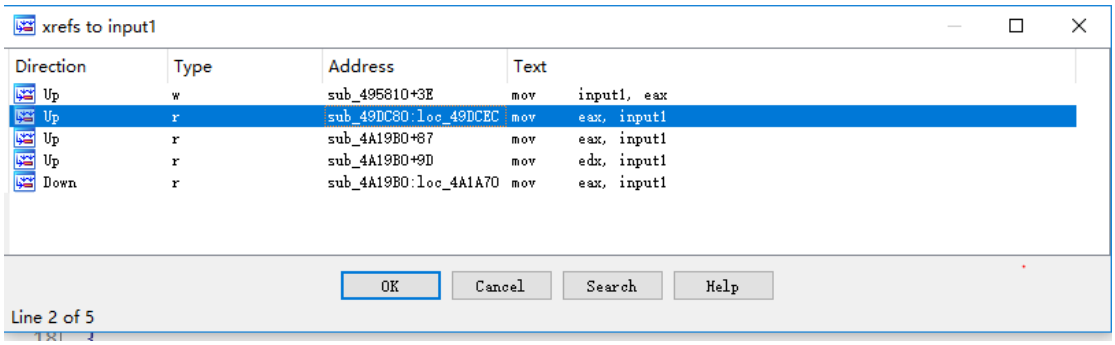
```
int __usercall sub_49DBD0@<eax>(int xmm0_4_0@<xmm0>, int a1)
{
  int v2; // edx
  int v3; // ecx
  unsigned int i; // [esp+D0h] [ebp-8h]

  sub_48D7B4((int)&unk_5F6007);
  *(_BYTE *)(a1 + 7) = 35;
  for ( i = 0; i < len(a1); ++i )
    *(_BYTE *)(i + a1) ^= 0x1Fu;
  return sub_48D935(v3, v2, 1, a1, xmm0_4_0);
}
```

这个函数，跟 input 暂时无关，拉出来独立讨论，a1 是参数，把 a1 的第 8 个字符变成 ascii（35）=‘#’，然后将 a1 的每一位都与 0x1F 异或。
至此，主函数部分追踪完毕。

0x03

根据大佬 wp，经验+直觉，对全局变量 input1，查找引用，



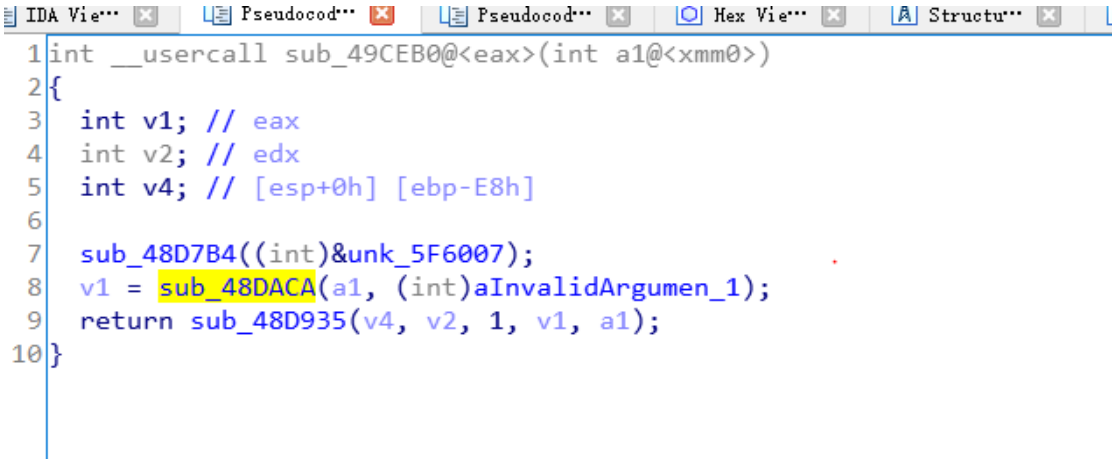
找到了下面的函数

```
1 int __userpurge sub_49DC80@<eax>(int a1@<xmm0>, int a2)
2 {
3   int v2; // edx
4   int v3; // ecx
5   unsigned int i; // [esp+E8h] [ebp-14h]
6
7   sub_48D7B4((int)&unk_5F6007);
8   if ( a2 )
9   {
.0    for ( i = 0; i < len(a2); ++i )
.1      *(_BYTE *)(i + a2) ^= 0x1Cu;
.2    if ( !cmp(a2, input1) )
.3    {
.4      sub_48B4AA(&unk_5F31E0, 'o');
.5      sub_48B4AA(&unk_5F31E0, 'k');
.6    }
.7  }
.8  return sub_48D935(v3, v2, 1, 0, a1);
.9 }
```

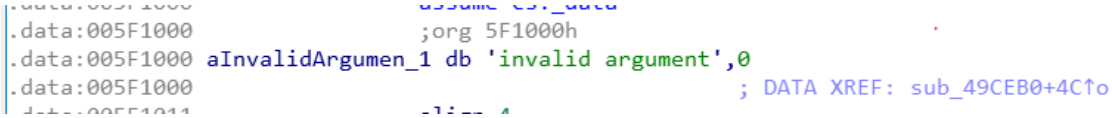
这又是一个校验函数，将 a2 的每一位，与 0x1C 异或，比较 a2 与 input 两者是否相等。这时，a2 是什么，就很关键了。查找对于该函数的上层引用。

```
int __userpurge sub_48DACA@<eax>(int a1@<xmm0>, int a2)
{
  return sub_49DC80(a1, a2);
}
```

没东西，再来一次



可以看到，变量出现了，aInvalidArgumen_1，跟进去，查看值



aInvalidArgumen_1==invalid argument

0x04

再理一遍逻辑
整个校验算法如下：
key[7]='A'
key[7]='#'
key = key^0x1F

flag = 'invalid argument'
flag = refkey ^0x1C

key==flag

可以写脚本了，运行如下

```python
a = "invalid argument"
input = ''
for c in a:
    input += chr(ord(c) ^ 0x1c ^ 0x1f)
input = input[:7] + 'A' + input[8:]
print input
```

```
D:\python\python.exe D:/python玩玩/123.py
jmubojgAbqdvnfmw
```