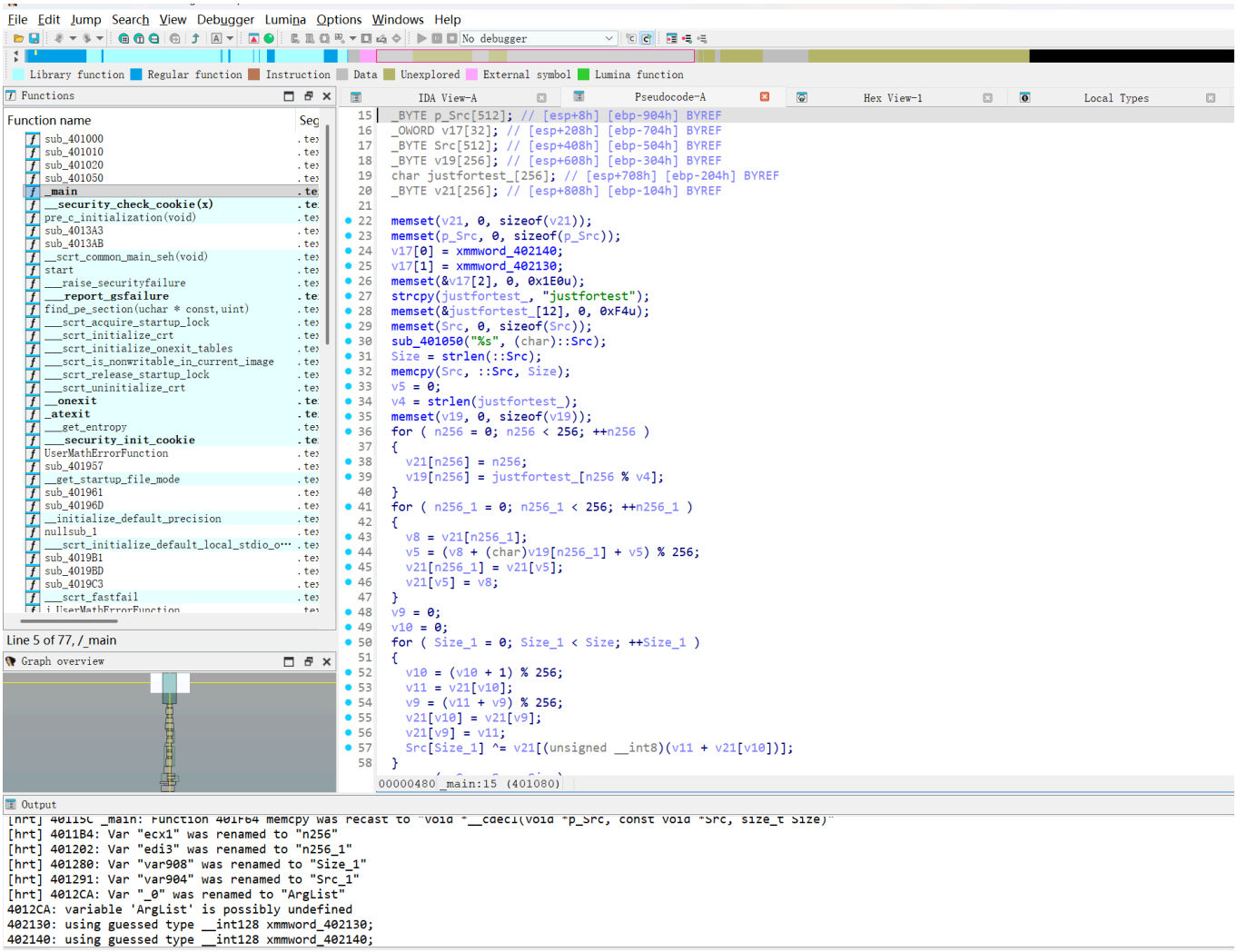
RC4专栏

逆向题目wp

张程思

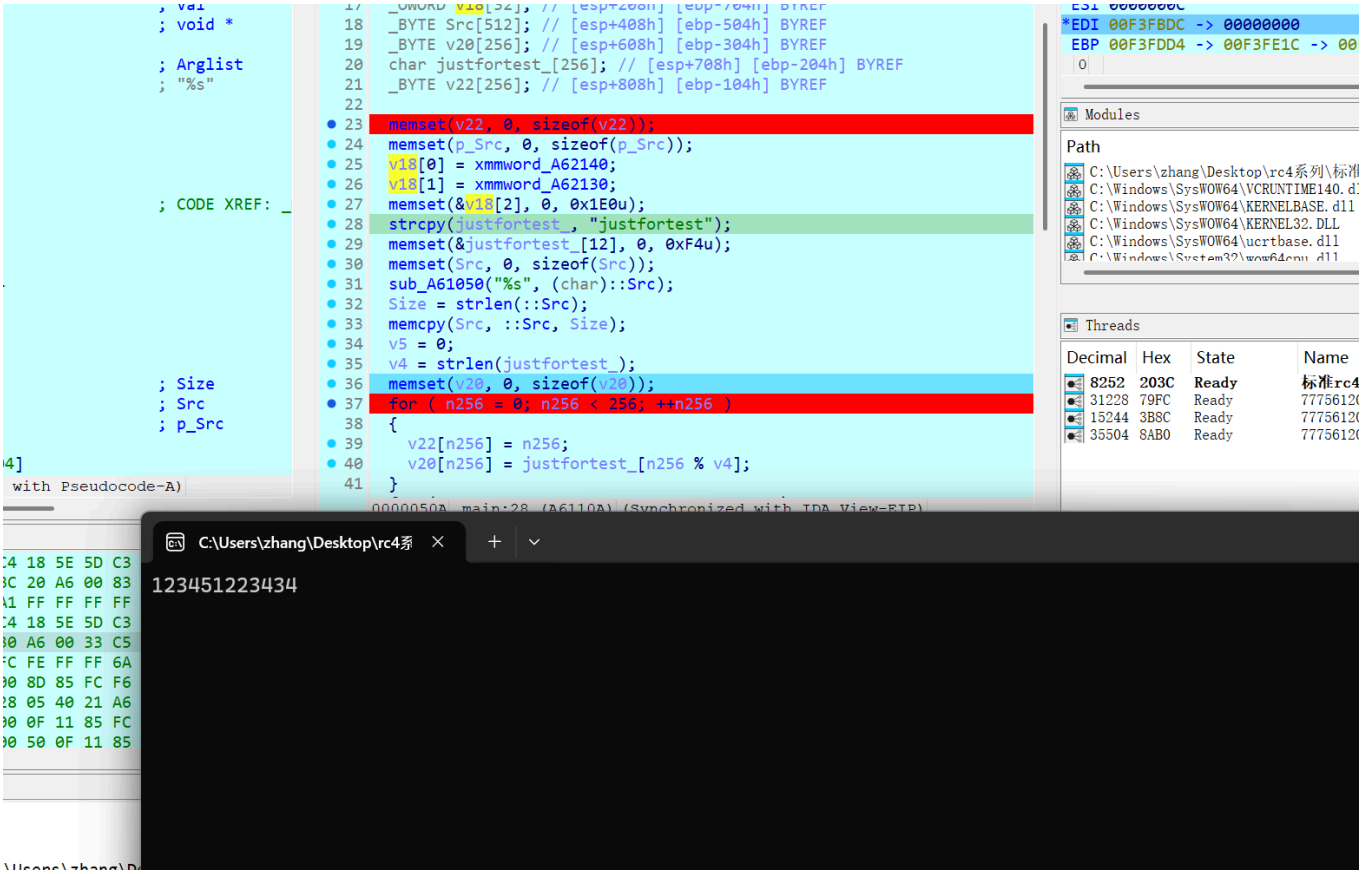
一,标准rc4.exe

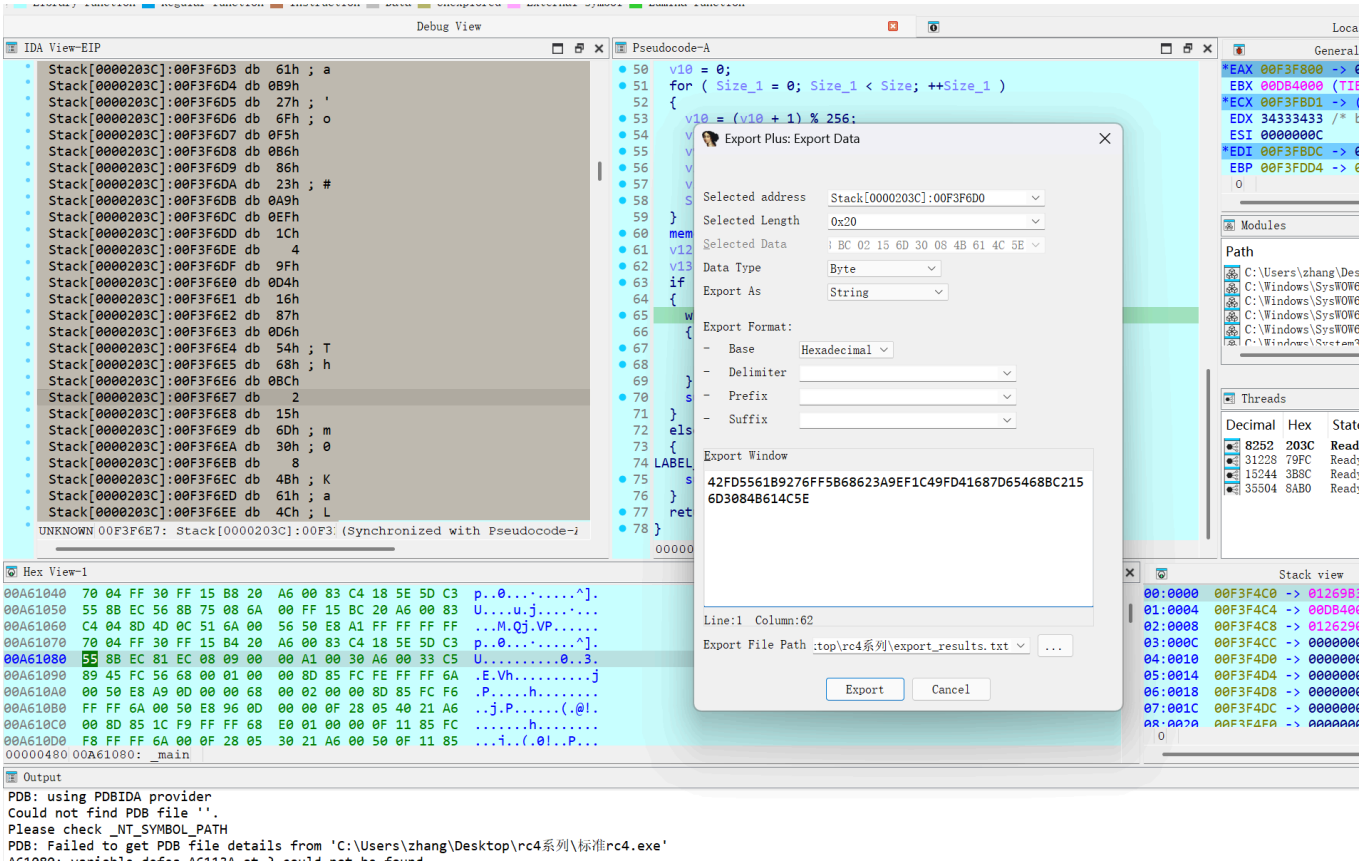
1,无壳,ida查看main函数,很明显的for循环256等等,标准的rc4解密,

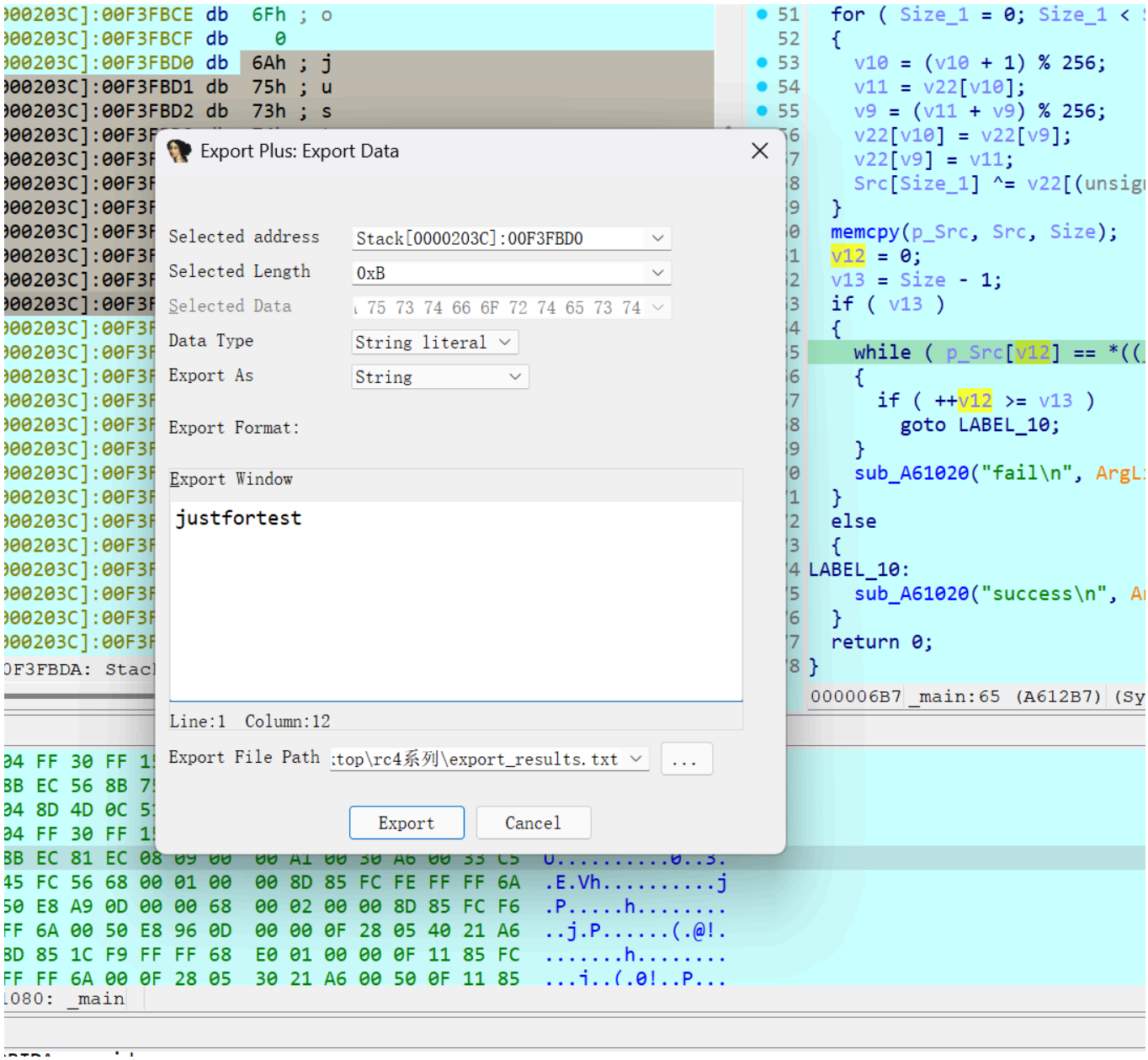



```
15  _BYTE p_Src[512]; // [esp+8h] [ebp-904h] BYREF
16  _OWORD v17[32]; // [esp+208h] [ebp-704h] BYREF
17  _BYTE Src[512]; // [esp+408h] [ebp-504h] BYREF
18  _BYTE v19[256]; // [esp+608h] [ebp-304h] BYREF
19  char justfortest_[256]; // [esp+708h] [ebp-204h] BYREF
20  _BYTE v21[256]; // [esp+808h] [ebp-104h] BYREF
21
22  memset(v21, 0, sizeof(v21));
23  memset(p_Src, 0, sizeof(p_Src));
24  v17[0] = xmmword_402140;
25  v17[1] = xmmword_402130;
26  memset(&v17[2], 0, 0x1E0u);
27  strcpy(justfortest_, "justfortest");
28  memset(&justfortest_[12], 0, 0xF4u);
29  memset(Src, 0, sizeof(Src));
30  sub_401050("%s", (char)::Src);
31  Size = strlen(::Src);
32  memcpy(Src, ::Src, Size);
33  v5 = 0;
34  v4 = strlen(justfortest_);
35  memset(v19, 0, sizeof(v19));
36  for ( n256 = 0; n256 < 256; ++n256 )
37  {
38    v21[n256] = n256;
39    v19[n256] = justfortest_[n256 % v4];
40  }
41  for ( n256_1 = 0; n256_1 < 256; ++n256_1 )
42  {
43    v8 = v21[n256_1];
44    v5 = (v8 + (char)v19[n256_1] + v5) % 256;
45    v21[n256_1] = v21[v5];
46    v21[v5] = v8;
47  }
48  v9 = 0;
49  v10 = 0;
50  for ( Size_1 = 0; Size_1 < Size; ++Size_1 )
51  {
52    v10 = (v10 + 1) % 256;
53    v11 = v21[v10];
54    v9 = (v11 + v9) % 256;
55    v21[v10] = v21[v9];
56    v21[v9] = v11;
57    Src[Size_1] ^= v21[(unsigned __int8)(v11 + v21[v10])];
58  }
```

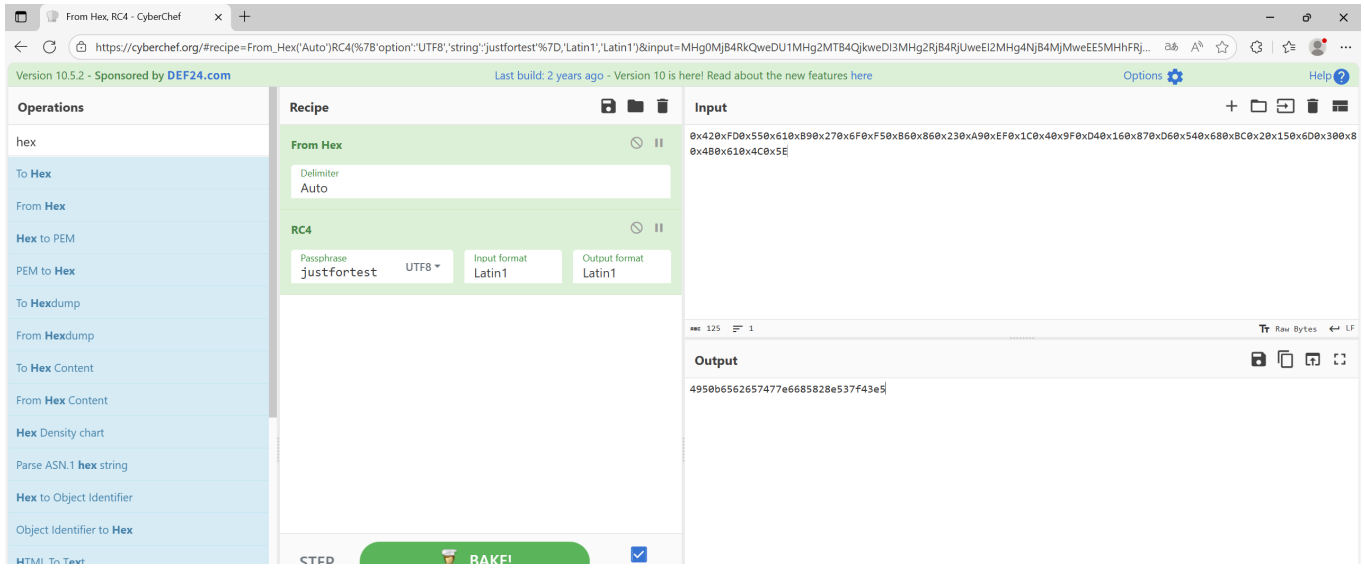
2,直接动态调试,静态分析看看怎么个事,直接设断点,随便输入,提取哈希字符串和密钥




Debug View
IDA View-EIP
Stack[0000203C]:00F3F6D3 db 61h ; a
Stack[0000203C]:00F3F6D4 db 0B9h
Stack[0000203C]:00F3F6D5 db 27h ; '
Stack[0000203C]:00F3F6D6 db 6Fh ; o
Stack[0000203C]:00F3F6D7 db 0F5h
Stack[0000203C]:00F3F6D8 db 0B6h
Stack[0000203C]:00F3F6D9 db 86h
Stack[0000203C]:00F3F6DA db 23h ; #
Stack[0000203C]:00F3F6DB db 0A9h
Stack[0000203C]:00F3F6DC db 0EFh
Stack[0000203C]:00F3F6DD db 1Ch
Stack[0000203C]:00F3F6DE db 4
Stack[0000203C]:00F3F6DF db 9Fh
Stack[0000203C]:00F3F6E0 db 0D4h
Stack[0000203C]:00F3F6E1 db 16h
Stack[0000203C]:00F3F6E2 db 87h
Stack[0000203C]:00F3F6E3 db 0D6h
Stack[0000203C]:00F3F6E4 db 54h ; T
Stack[0000203C]:00F3F6E5 db 68h ; h
Stack[0000203C]:00F3F6E6 db 0BCh
Stack[0000203C]:00F3F6E7 db 2
Stack[0000203C]:00F3F6E8 db 15h
Stack[0000203C]:00F3F6E9 db 6Dh ; m
Stack[0000203C]:00F3F6EA db 30h ; 0
Stack[0000203C]:00F3F6EB db 8
Stack[0000203C]:00F3F6EC db 4Bh ; K
Stack[0000203C]:00F3F6ED db 61h ; a
Stack[0000203C]:00F3F6EE db 4Ch ; L
UNKNOWN 00F3F6E7: Stack[0000203C]:00F3
Pseudocode-A
v10 = 0;
for (Size_1 = 0; Size_1 < Size; ++Size_1)
{
v10 = (v10 + 1) % 256;
Export Plus: Export Data
Selected address
Stack[0000203C]:00F3F6D0
Selected Length
0x20
Selected Data
BC 02 15 6D 30 08 4B 61 4C 5E
Data Type
Byte
Export As
String
Export Format:
Base
Hexadecimal
Delimiter
Prefix
Suffix
Export Window
42FD5561B9276FF5B68623A9EF1C49FD41687D65468BC215
6D3084B614C5E
Line:1 Column:62
Export File Path
:top\rc4系列\export_results.txt
...
Export
Cancel
General
*EAX 00F3F800 -> 6
EBX 00DB4000 (TIB
*ECX 00F3FBD1 -> (
EDX 34333433 /* b
ESI 0000000C
*EDI 00F3FBDC -> 6
EBP 00F3FDD4 -> 6
Modules
Path
C:\Users\zhang\Des
C:\Windows\SysWOW6
C:\Windows\SysWOW6
C:\Windows\SysWOW6
C:\Windows\SysWOW6
Threads
Decimal Hex Stat
8252 203C Read
31228 79FC Ready
15244 3B8C Ready
35504 8AB0 Ready
Hex View-1
00A61040 70 04 FF 30 FF 15 B8 20 A6 00 83 C4 18 5E 5D C3 p..0... .....^].
00A61050 55 8B EC 56 8B 75 08 6A 00 FF 15 BC 20 A6 00 83 U....u.j.... ...
00A61060 C4 04 8D 4D 0C 51 6A 00 56 50 E8 A1 FF FF FF FF ...M.Qj.VP......
00A61070 70 04 FF 30 FF 15 B4 20 A6 00 83 C4 18 5E 5D C3 p..0... .....^].
00A61080 55 8B EC 81 EC 08 09 00 00 A1 00 30 A6 00 33 C5 U.........0..3.
00A61090 89 45 FC 56 68 00 01 00 00 8D 85 FC FE FF FF 6A .E.Vh.........j
00A610A0 00 50 E8 A9 0D 00 00 68 00 02 00 00 8D 85 FC F6 .P.....h........
00A610B0 FF FF 6A 00 50 E8 96 0D 00 00 0F 28 05 40 21 A6 ..j.P......(.@!.
00A610C0 00 8D 85 1C F9 FF FF 68 E0 01 00 00 0F 11 85 FC ....h........
00A610D0 F8 FF FF 6A 00 0F 28 05 30 21 A6 00 50 0F 11 85 ...j..(.0!..P...
00000480 00A61080: _main
Stack view
00:0000 00F3F4C0 -> 012698
01:0004 00F3F4C4 -> 00DB40
02:0008 00F3F4C8 -> 012629
03:000C 00F3F4CC -> 000000
04:0010 00F3F4D0 -> 000000
05:0014 00F3F4D4 -> 000000
06:0018 00F3F4D8 -> 000000
07:001C 00F3F4DC -> 000000
Output
PDB: using PDBIDA provider
Could not find PDB file ''.
Please check _NT_SYMBOL_PATH
PDB: Failed to get PDB file details from 'C:\Users\zhang\Desktop\rc4系列\标准rc4.exe'

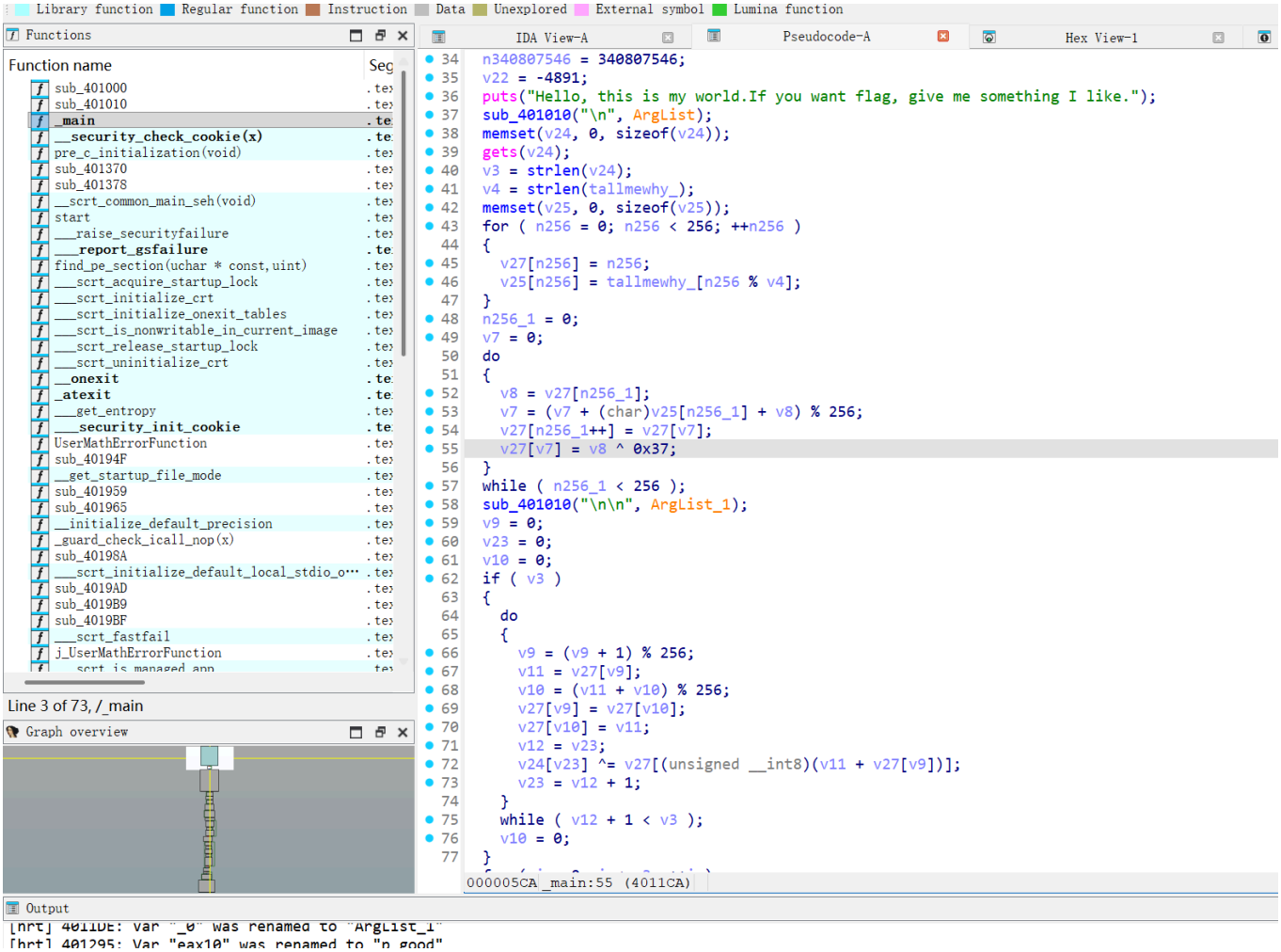


3,赛博厨子直接秒了,只不过没有flag头,自己加上就行

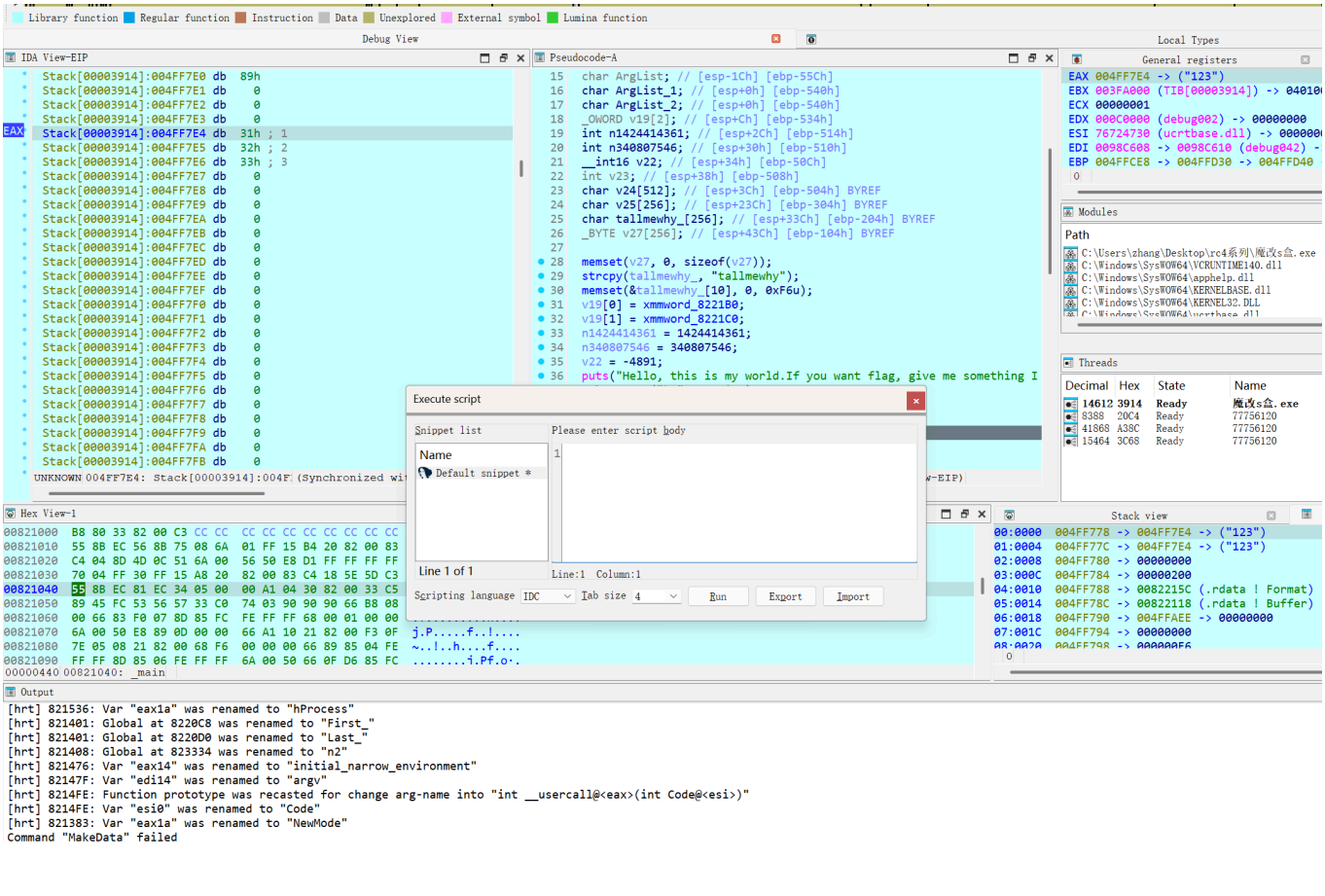


二,魔改rs4盒

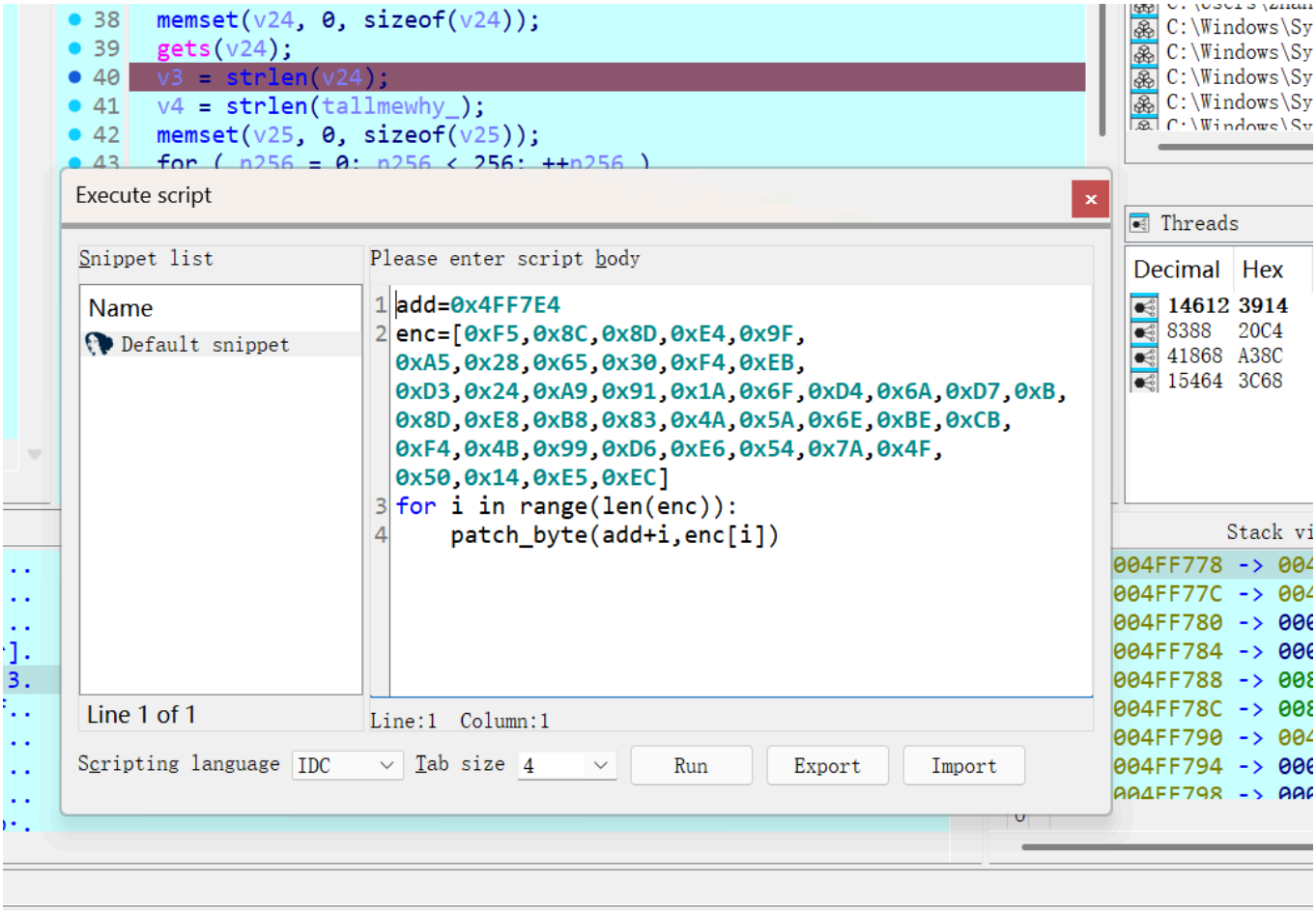
1,ida中查看发现并非标准rc4,而是存在异或



2,断个点直接动态调试静态分析走起,看了一下,patch做这个题更方便

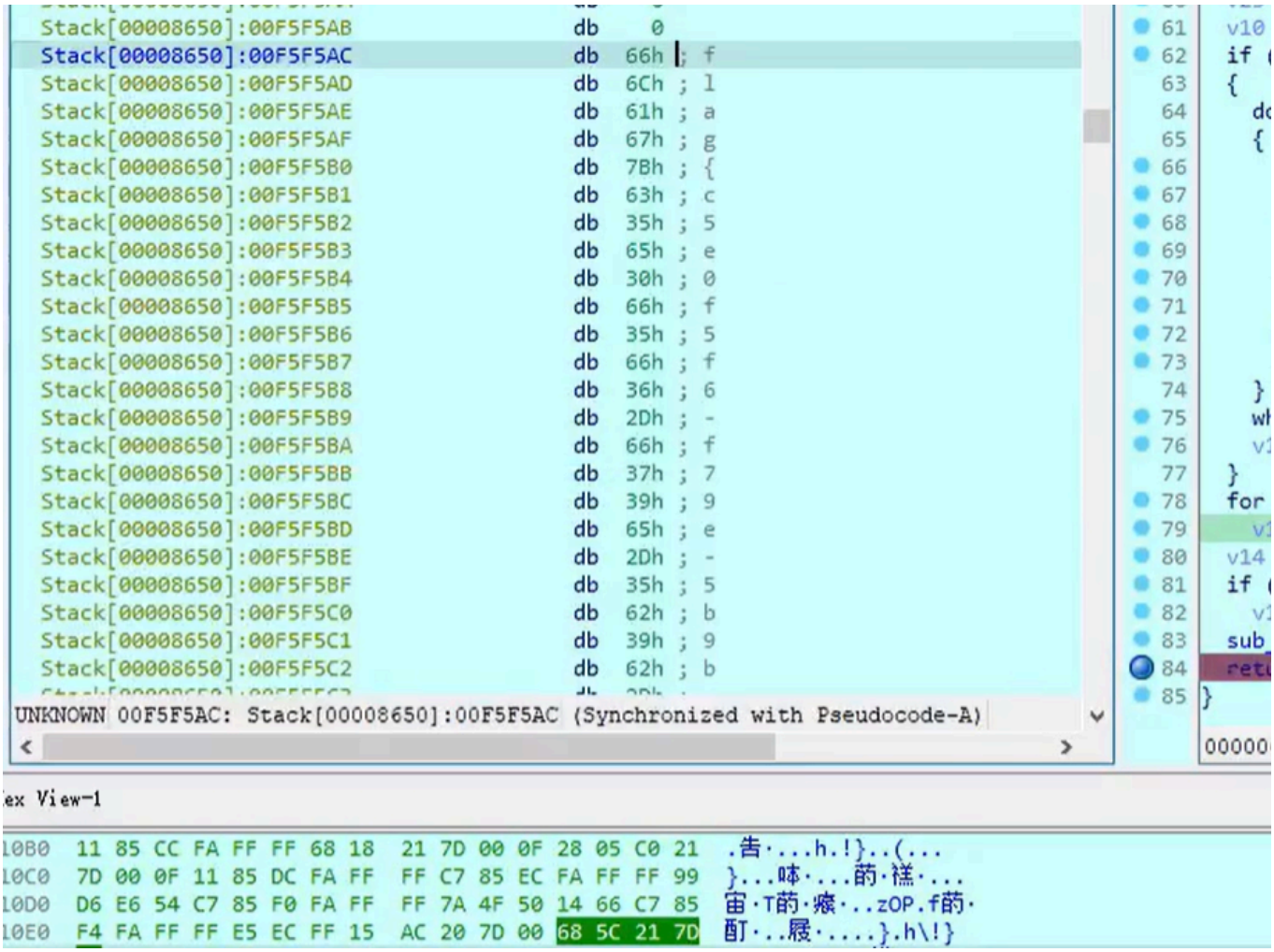


3,把密钥和密文提取出来,在ida里写个小py脚本



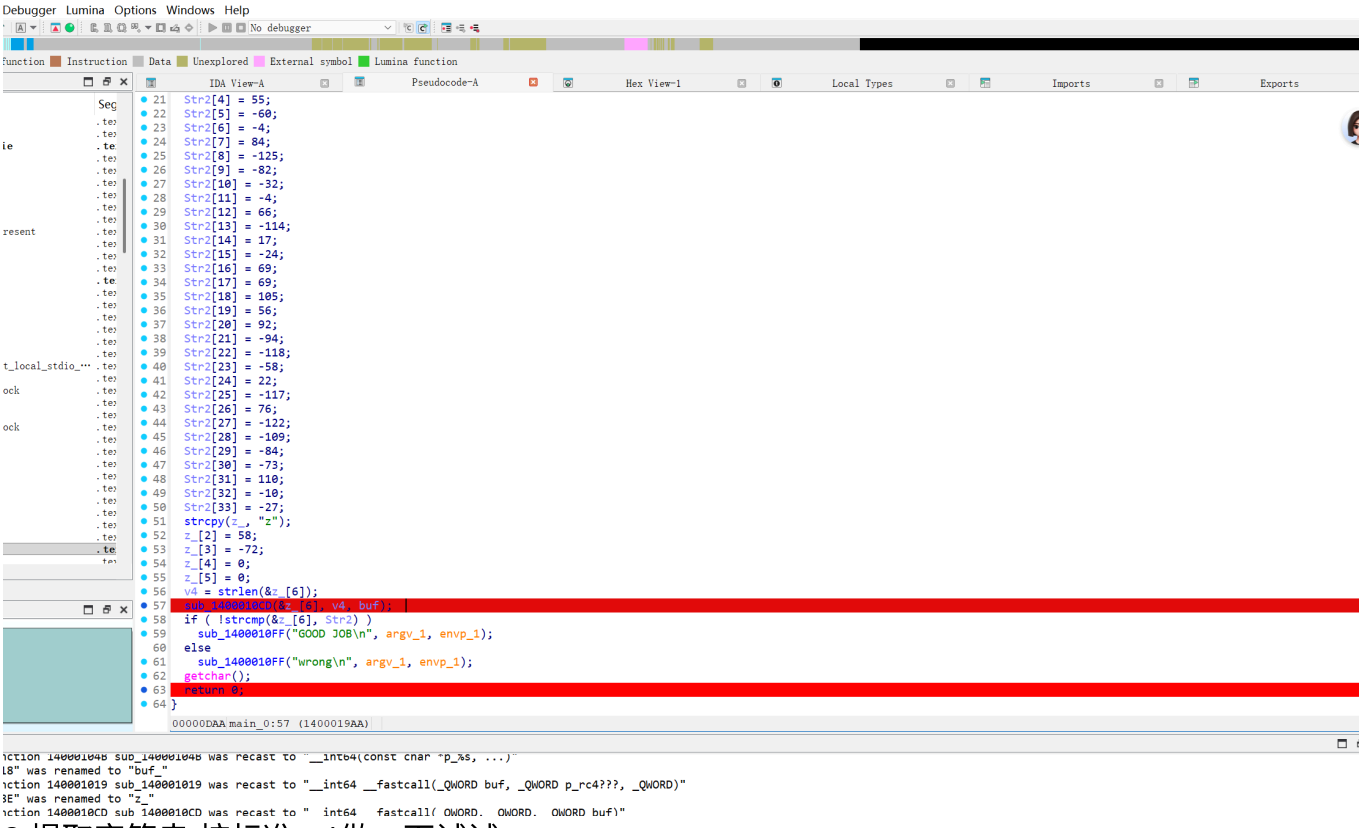
0x91,0x1A,0x6F,0xD4,0x6A,0xD7,0xB,0x8D,0xE8,0xB8,0x83,0x4A,0x5A,0x6E,0xBE,0xCB,0xF4,0x4B,0x99

4,在最后下个断点,然后再点击查看我们的输入的值,flag就出来了,提取即可

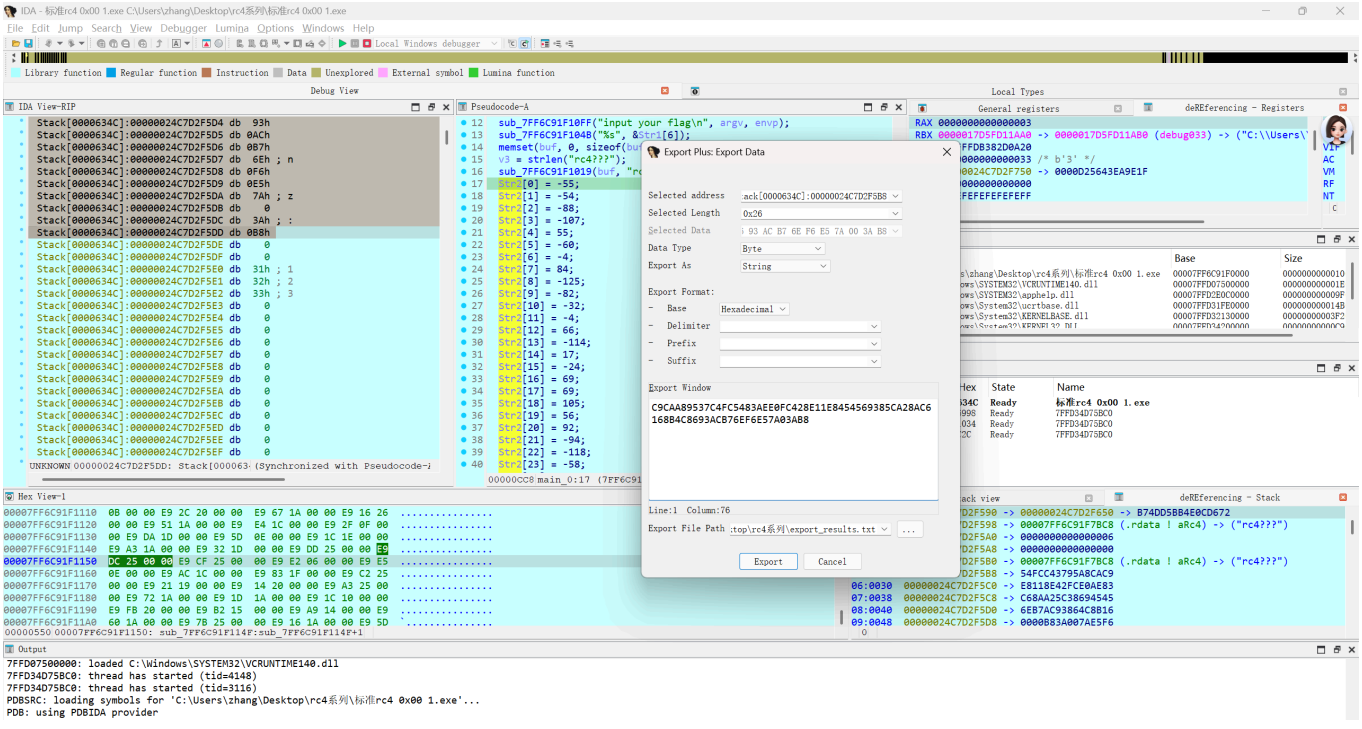


三,标准rc4 001

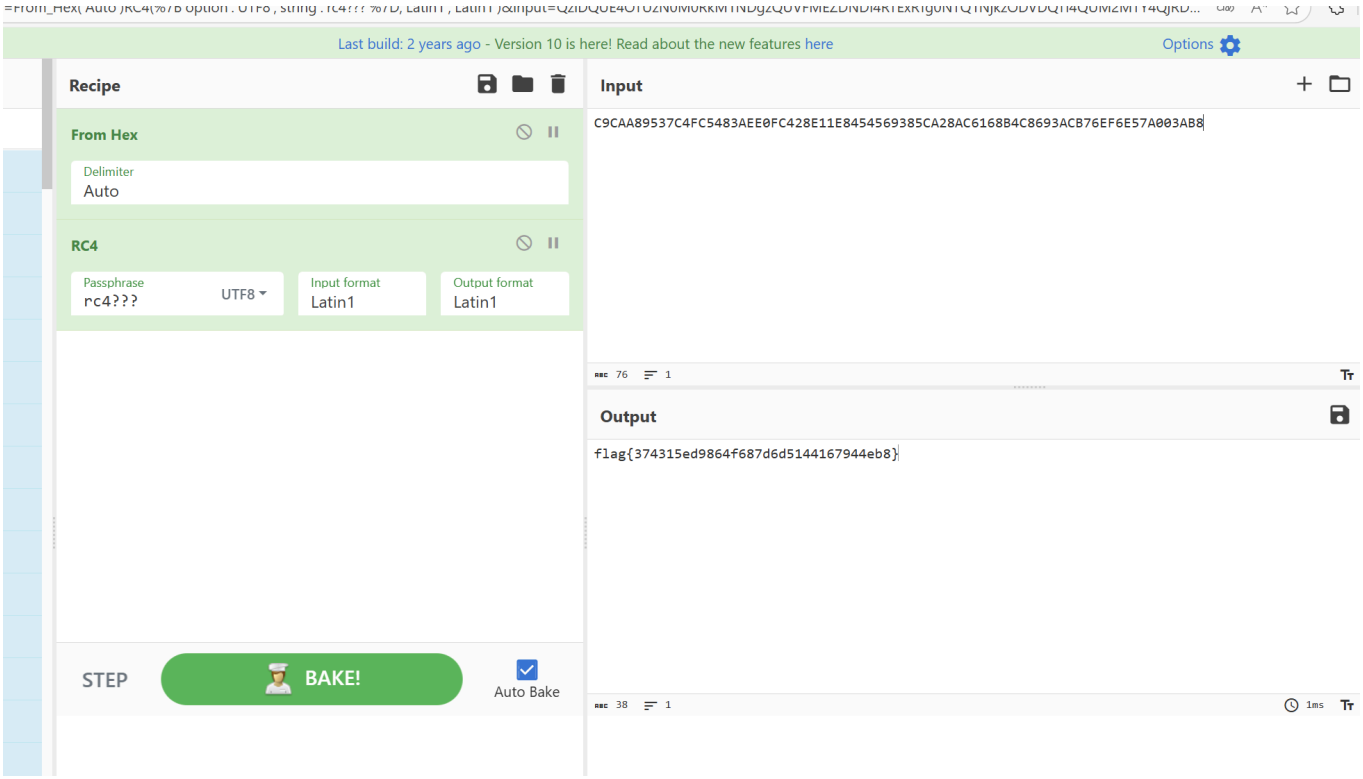
1,ida查看分析一下,算了,直接断点调试



2,提取字符串,按标准rc4做一下试试

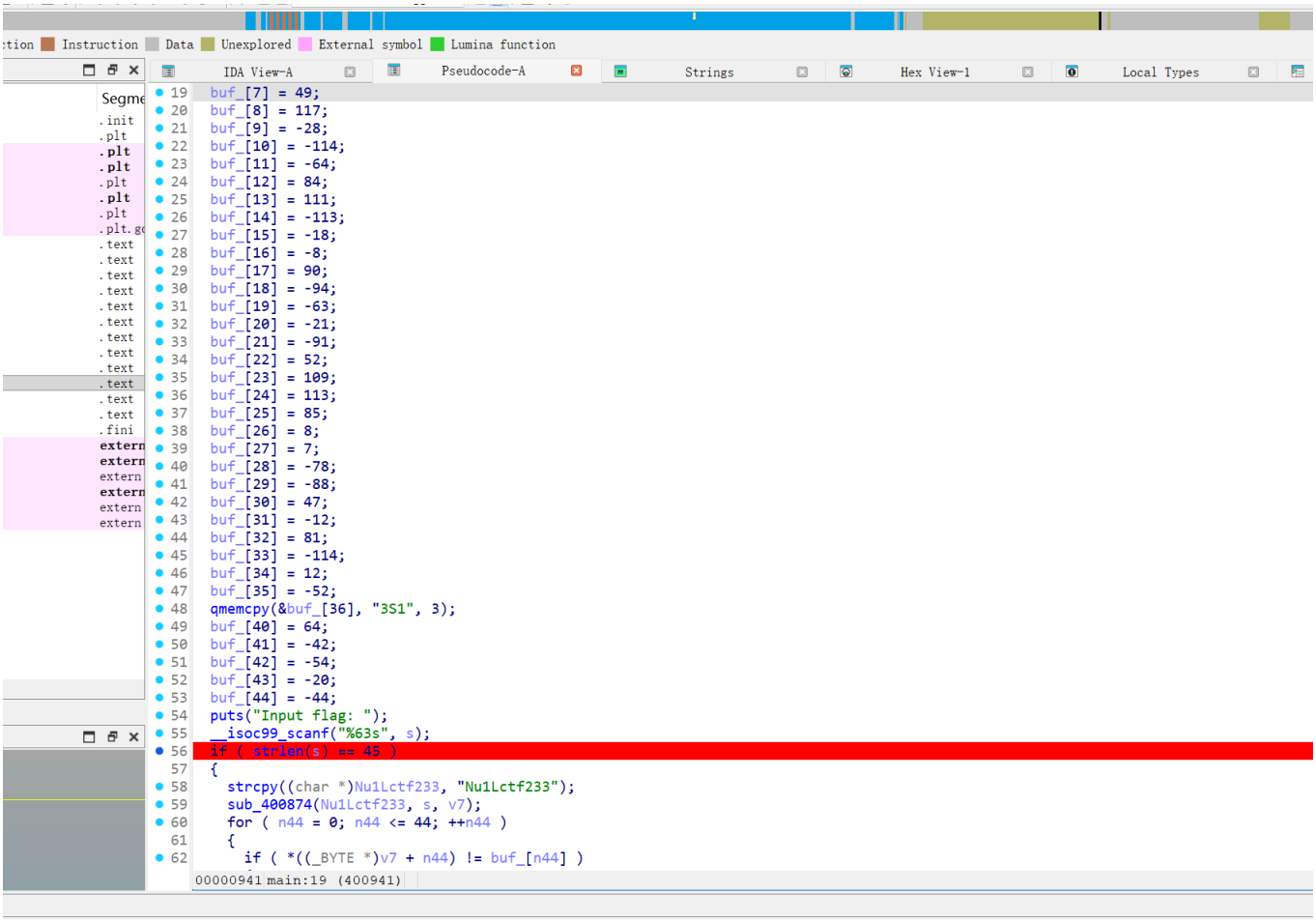


3,赛博厨子秒了



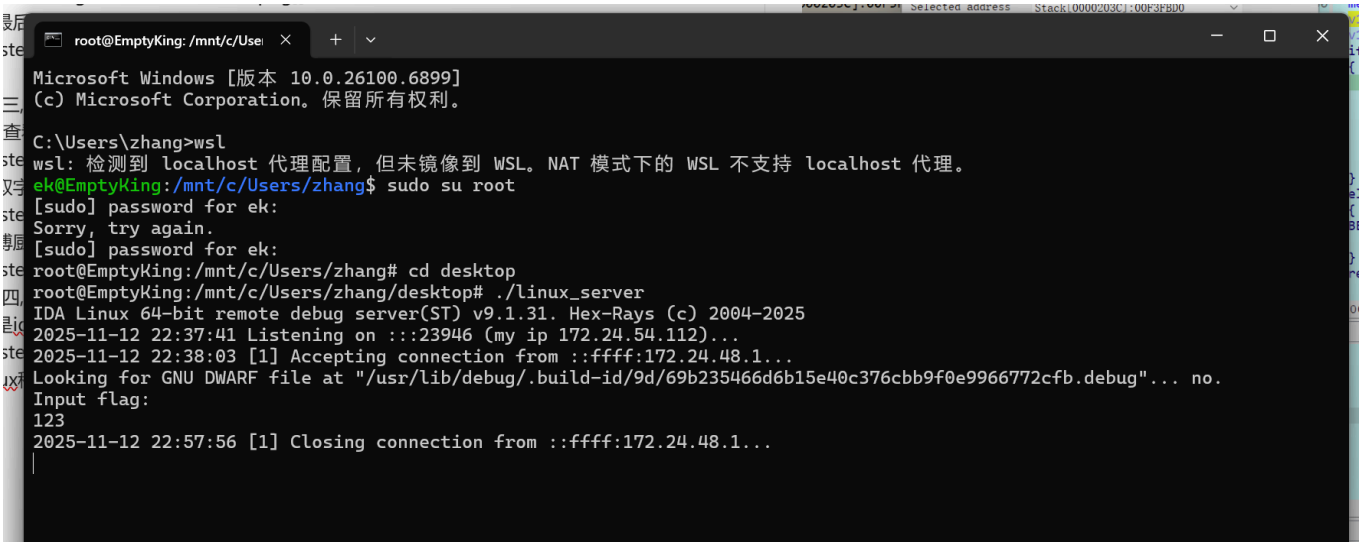
四,标准rc4 002

1,还是ida分析一下,依旧标准rc4

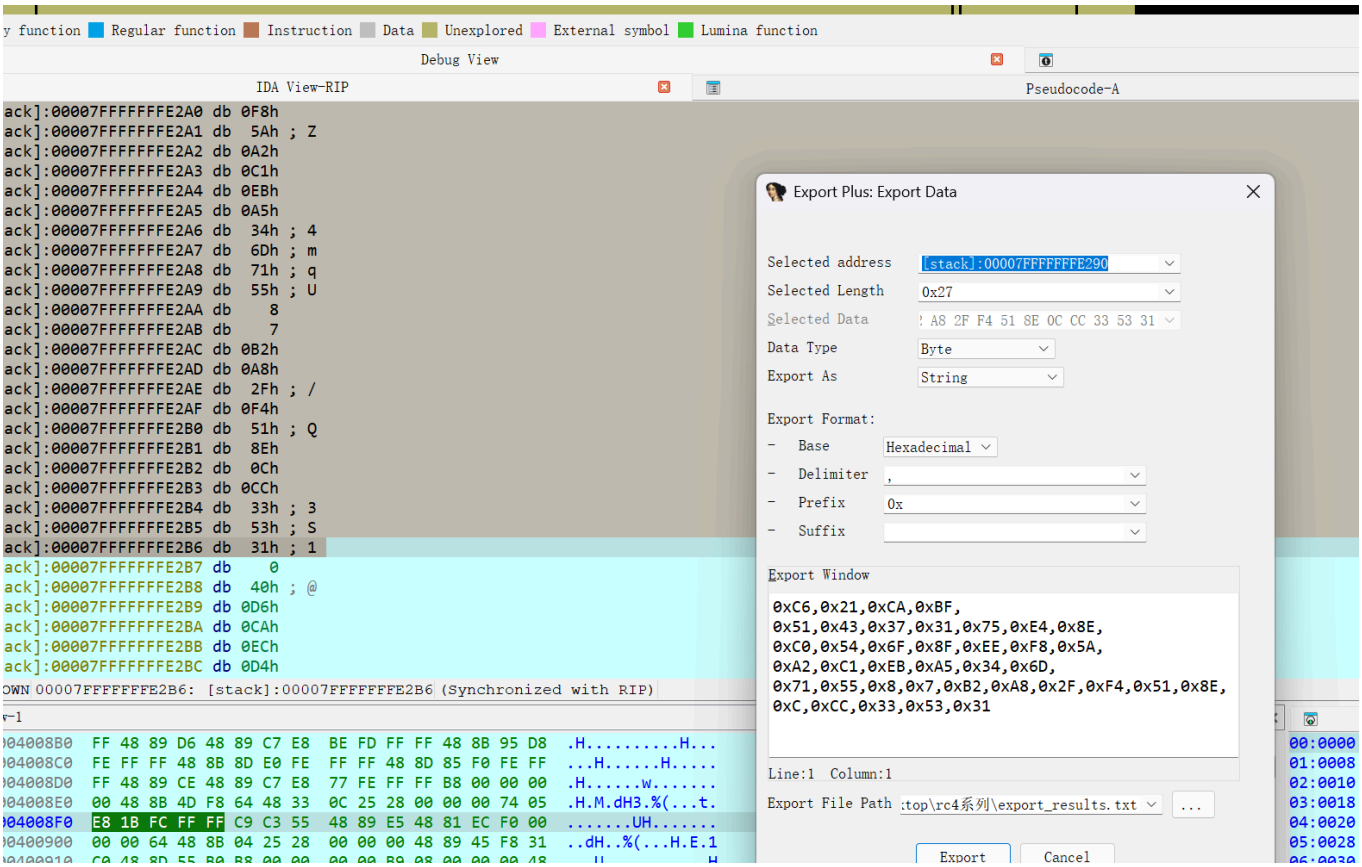


, length)

2,linux程序,所以wsl调试一下



3,提取一下密文

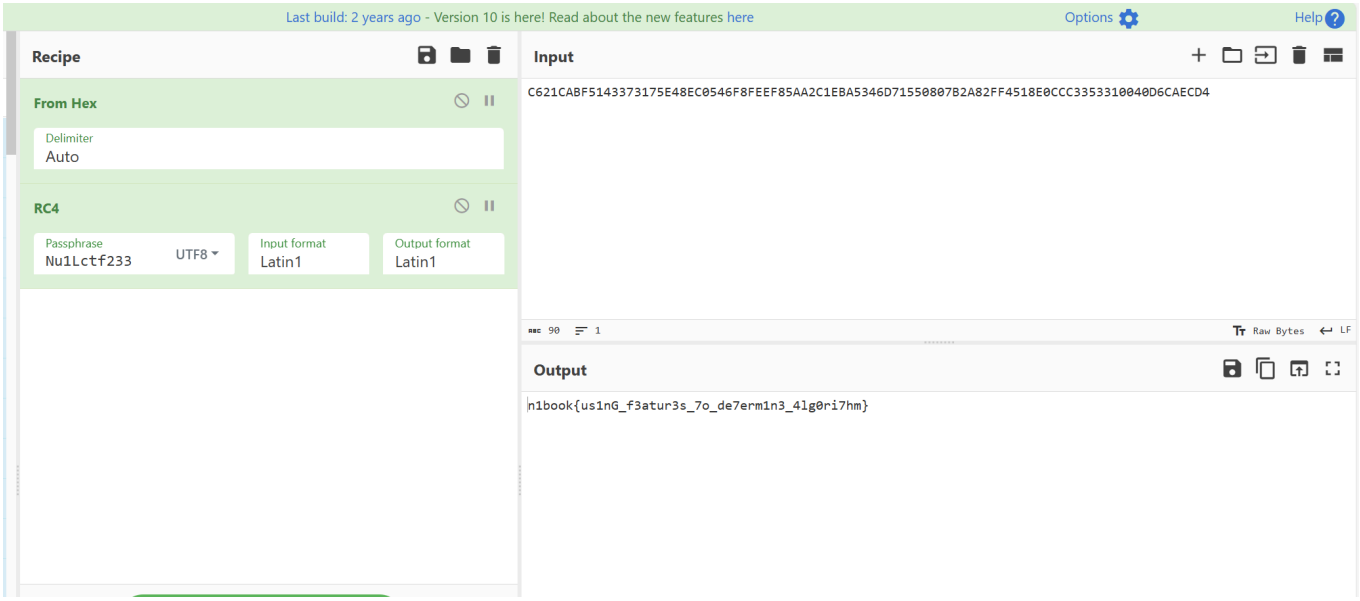


4,我这个傻逼ida不自动补零,上个题我还能手动补一下,这个题没补的太多了,干脆让ai给我写了个自动补零的脚本

```
import re

# 1 输入文件路径（你的导出文件）
input_file = r"C:\Users\zhang\Desktop\rc4系列\export_results.txt"

# 2 输出文件路径（修正后保存）
output_file = r"C:\Users\zhang\Desktop\rc4系列\export_fixed.txt"

# 3 读取文件内容
with open(input_file, "r", encoding="utf-8") as f:
    text = f.read()

# 4 修正十六进制：补零并转大写（0xC → 0x0C）
def fix_hex(match):
    val = match.group(1)
    if len(val) == 1:
        return f"0x0{val.upper()}"
    else:
        return f"0x{val.upper()}"

# 执行补零修正
fixed_text = re.sub(r'0x([0-9A-Fa-f]{1,2})\b', fix_hex, text)
```

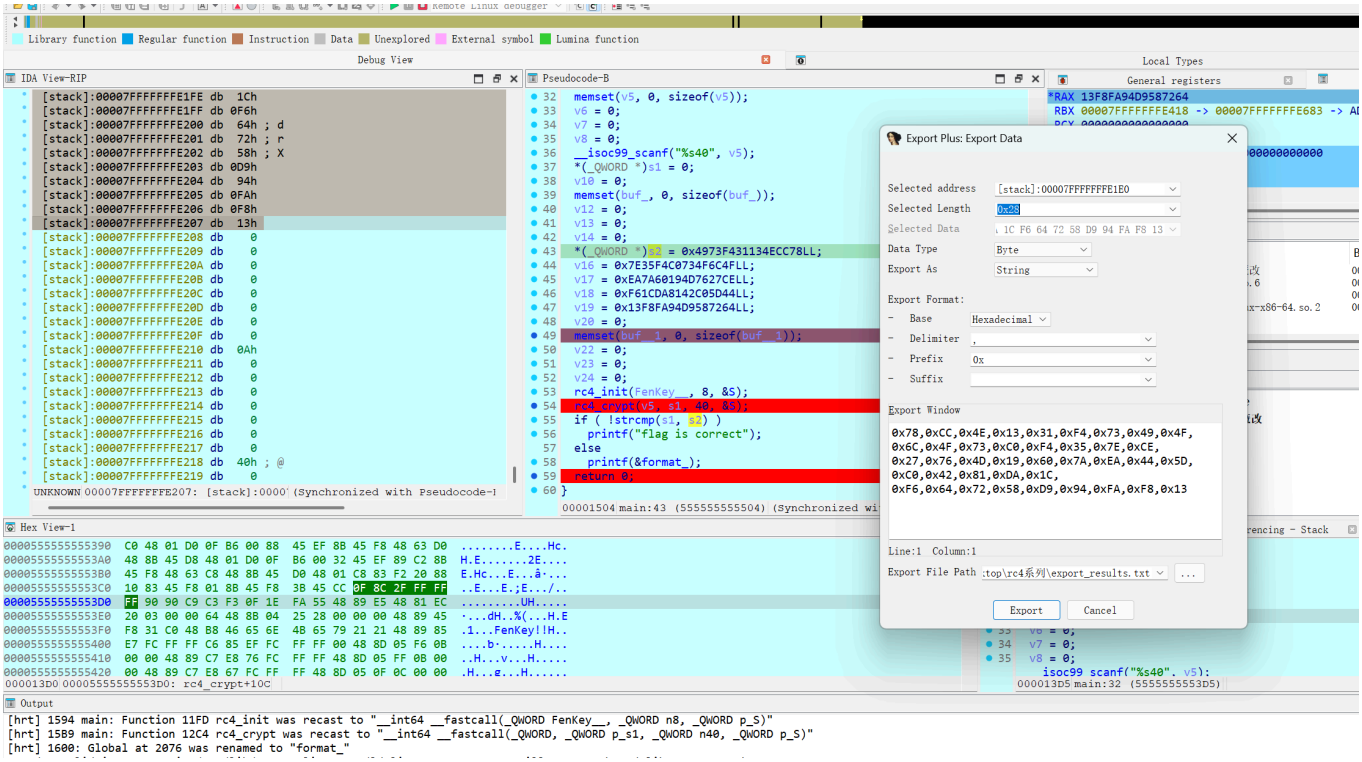
```
# 5️⃣ 删除 0x、逗号、空格等符号，只保留纯16进制字符
cleaned_text = fixed_text.replace("0x", "").replace(",", "").replace(" ", "")
cleaned_text = cleaned_text.strip().upper()

# 6️⃣ 写入输出文件
with open(output_file, "w", encoding="utf-8") as f:
    f.write(cleaned_text)

print("✅ 已修正完成并清理所有符号！")
print("📄 输出文件路径：", output_file)
```

5,把密钥也提取出来,赛博厨子一把梭

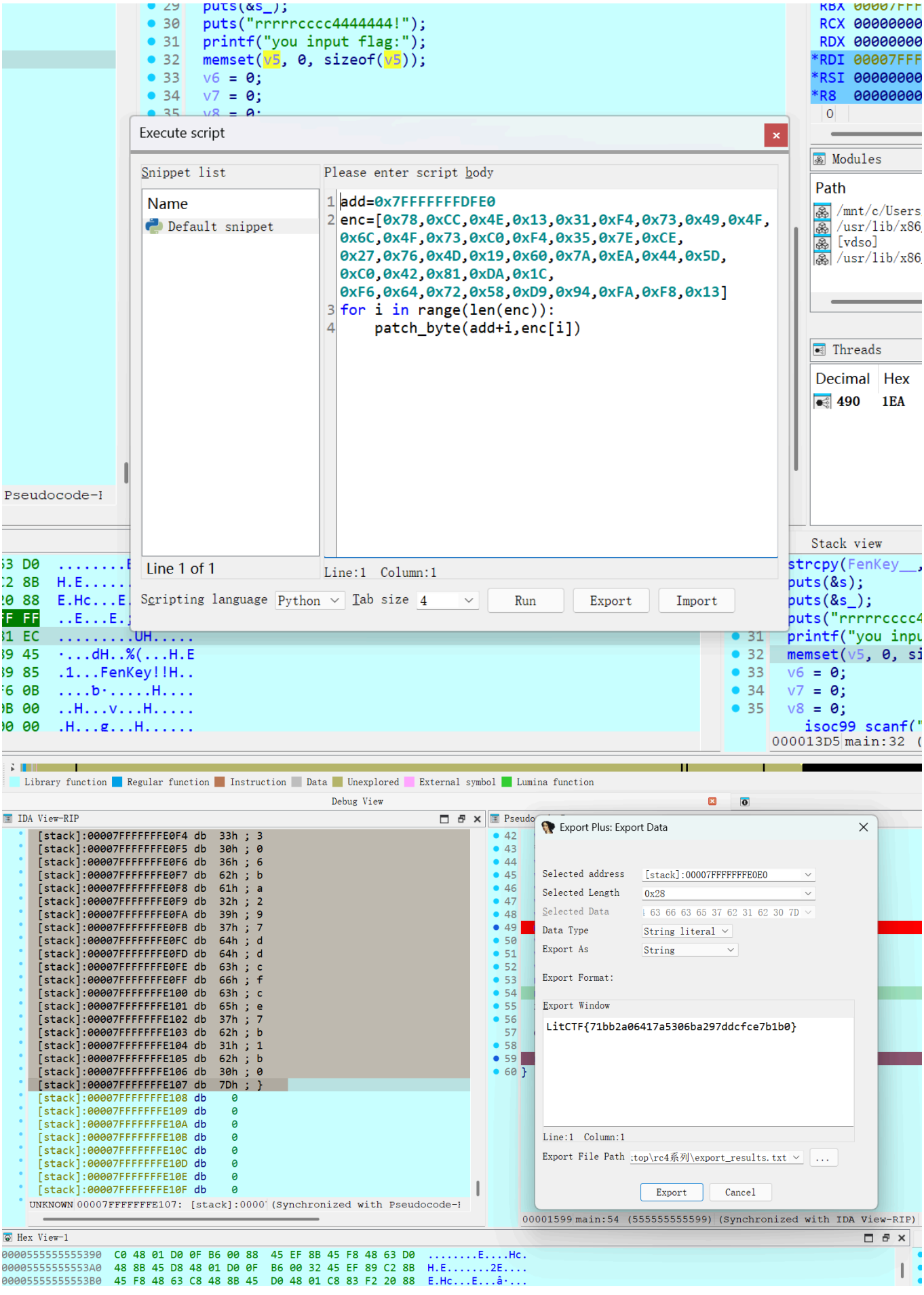


五,异或魔改

1,我先当常规rc4的标准做,这些截图不重复放了(没做出来),那就ptach做吧,先下断点提取



2,ptach之后flag直接出来了


puts("rrrrrcccc4444444!");
printf("you input flag:");
memset(v5, 0, sizeof(v5));
v6 = 0;
v7 = 0;
Execute script
Snippet list
Name
Default snippet
Line 1 of 1
Please enter script body
add=0x7FFFFFFFDFE0
enc=[0x78,0xCC,0x4E,0x13,0x31,0xF4,0x73,0x49,0x4F,
0x6C,0x4F,0x73,0xC0,0xF4,0x35,0x7E,0xCE,
0x27,0x76,0x4D,0x19,0x60,0x7A,0xEA,0x44,0x5D,
0xC0,0x42,0x81,0xDA,0x1C,
0xF6,0x64,0x72,0x58,0xD9,0x94,0xFA,0xF8,0x13]
for i in range(len(enc)):
patch_byte(add+i,enc[i])
Line:1 Column:1
Scripting language Python
Tab size 4
Run
Export
Import
Modules
Path
/mnt/c/Users
/usr/lib/x86
[vdso]
/usr/lib/x86
Threads
Decimal
Hex
490
1EA
Stack view
Library function
Regular function
Instruction
Data
Unexplored
External symbol
Lumina function
Debug View
IDA View-RIP
[stack]:00007FFFFFFFE0F4 db 33h ; 3
[stack]:00007FFFFFFFE0F5 db 30h ; 0
[stack]:00007FFFFFFFE0F6 db 36h ; 6
[stack]:00007FFFFFFFE0F7 db 62h ; b
[stack]:00007FFFFFFFE0F8 db 61h ; a
[stack]:00007FFFFFFFE0F9 db 32h ; 2
[stack]:00007FFFFFFFE0FA db 39h ; 9
[stack]:00007FFFFFFFE0FB db 37h ; 7
[stack]:00007FFFFFFFE0FC db 64h ; d
[stack]:00007FFFFFFFE0FD db 64h ; d
[stack]:00007FFFFFFFE0FE db 63h ; c
[stack]:00007FFFFFFFE0FF db 66h ; f
[stack]:00007FFFFFFFE100 db 63h ; c
[stack]:00007FFFFFFFE101 db 65h ; e
[stack]:00007FFFFFFFE102 db 37h ; 7
[stack]:00007FFFFFFFE103 db 62h ; b
[stack]:00007FFFFFFFE104 db 31h ; 1
[stack]:00007FFFFFFFE105 db 62h ; b
[stack]:00007FFFFFFFE106 db 30h ; 0
[stack]:00007FFFFFFFE107 db 7Dh ; }
[stack]:00007FFFFFFFE108 db 0
[stack]:00007FFFFFFFE109 db 0
[stack]:00007FFFFFFFE10A db 0
[stack]:00007FFFFFFFE10B db 0
[stack]:00007FFFFFFFE10C db 0
[stack]:00007FFFFFFFE10D db 0
[stack]:00007FFFFFFFE10E db 0
[stack]:00007FFFFFFFE10F db 0
Export Plus: Export Data
Selected address [stack]:00007FFFFFFFE0E0
Selected Length 0x28
Selected Data 63 66 63 65 37 62 31 62 30 7D
Data Type String literal
Export As String
Export Format:
Export Window
LitCTF{71bb2a06417a5306ba297ddcfce7b1b0}
Line:1 Column:1
Export File Path :top\rc4系列\export_results.txt
Export
Cancel
Hex View-1