

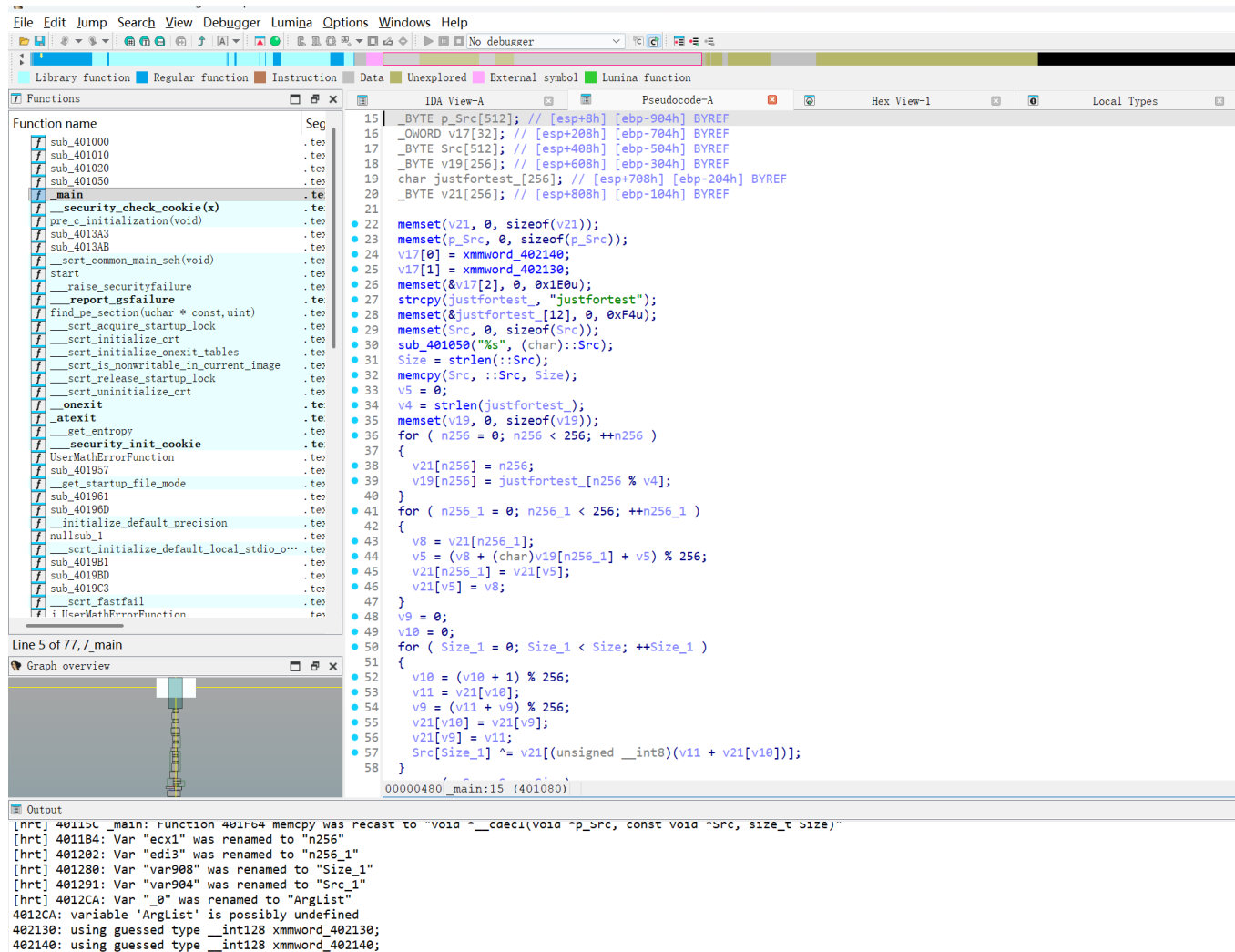
RC4专栏

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,



```
File Edit Jump Search View Debugger Lumina Options Windows Help
Library function Regular function Instruction Data Unexplored External symbol Lumina function

Function name Seg
sub_401000 .text
sub_401010 .text
sub_401020 .text
sub_401050 .text
main .text
security_check_cookie(x) .text
pre_c_initialization(void) .text
sub_4013A3 .text
sub_4013AB .text
srt_common_main_seh(void) .text
start .text
raise_securityfailure .text
report_gsfailure .text
find_pe_section(uchar * const,uint) .text
srt_acquire_startup_lock .text
srt_initialize_crt .text
srt_initialize_onexit_tables .text
srt_is_nonwritable_in_current_image .text
srt_release_startup_lock .text
srt_uninitialize_crt .text
onexit .text
atexit .text
get_entropy .text
security_init_cookie .text
UserMathErrorFunction .text
sub_401957 .text
get_startup_file_mode .text
sub_401961 .text
sub_40196D .text
initialize_default_precision .text
nullsub_1 .text
srt_initialize_default_local_stdio_o... .text
sub_4019B1 .text
sub_4019BD .text
sub_4019C3 .text
srt_fastfail .text
i UserMathErrorFunction .text

Line 5 of 77, /_main
Graph overview
Output
[hr] 40115C: _main: function 401964 memcpy was recast to "void *__cdecl(void *p_src, const void *src, size_t size)"
[hr] 401184: Var "ecx1" was renamed to "n256"
[hr] 401202: Var "edi3" was renamed to "n256_1"
[hr] 401280: Var "var908" was renamed to "Size_1"
[hr] 401291: Var "var904" was renamed to "Src_1"
[hr] 4012CA: Var "_0" was renamed to "ArgList"
4012CA: variable 'ArgList' is possibly undefined
402130: using guessed type __int128 xmmword_402130;
402140: using guessed type __int128 xmmword_402140;
```

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

```
void *
; void *

; Arglist
; "%s"

; CODE XREF: _
; Size
; Src
; p_Src
```

```
17 _DWORD v20[32]; // [esp+208h] [ebp+70h] BYREF
18 _BYTE Src[512]; // [esp+408h] [ebp+504h] BYREF
19 _BYTE v20[256]; // [esp+608h] [ebp+304h] BYREF
20 char justfortest_256; // [esp+708h] [ebp+204h] BYREF
21 _BYTE v22[256]; // [esp+808h] [ebp+104h] BYREF
22
23 memset(v22, 0, sizeof(v22));
24 memset(p_Src, 0, sizeof(p_Src));
25 v10[0] = xmmword_A62140;
26 v10[1] = xmmword_A62130;
27 memset(&v10[2], 0, 0x1E0u);
28 strcpy(justfortest_, "justfortest");
29 memset(&justfortest_[12], 0, 0xF4u);
30 memset(Src, 0, sizeof(Src));
31 sub_A61050("%s", (char)::Src);
32 Size = strlen(::Src);
33 memcpy(Src, ::Src, Size);
34 v5 = 0;
35 v4 = strlen(justfortest_);
36 memset(v20, 0, sizeof(v20));
37 for ( n256 = 0; n256 < 256; ++n256 )
38 {
39     v22[n256] = n256;
40     v20[n256] = justfortest_[n256 % v4];
41 }
```

```
EDI 00F3FBDC -> 00000000
EBP 00F3FDD4 -> 00F3FE1C -> 00
0
```

Modules

Path

C:\Users\zhang\Desktop\rc4系列\标准
C:\Windows\SysWOW64\VCRUNTIME140.d
C:\Windows\SysWOW64\KERNELBASE.dll
C:\Windows\SysWOW64\KERNEL32.DLL
C:\Windows\SysWOW64\userbase.dll
C:\Windows\System32\userbase.dll

Threads

Decimal	Hex	State	Name
8252	203C	Ready	标准rc4
31228	79FC	Ready	77756120
15244	3B8C	Ready	77756120
35504	8AB0	Ready	77756120

C:\Users\zhang\Desktop\rc4系列

123451223434

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]:00F3F6E7: (Synchronized with Pseudocode-)
```

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 88 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 ..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 00 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 .j.P.....h.....
00A610D0 F8 FF FF 6A 00 0F 28 05 30 21 A6 00 50 0F 11 85 .i..(.0!..P...
00000480 00A61080: _main
```

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'
```

Pseudocode-A

```
50 v10 = 0;
51 for ( Size_1 = 0; Size_1 < Size; ++Size_1 )
52 {
53     v10 = (v10 + 1) % 256;
54     v
55     v
56     v
57     v
58     S
59 }
60 mem
61 v12
62 v13
63 if
64 {
65     W
66     p
67     v
68     v
69     }
70     s
71     }
72     els
73     {
74 LABEL
75     s
76     }
77     ret
78 }
```

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

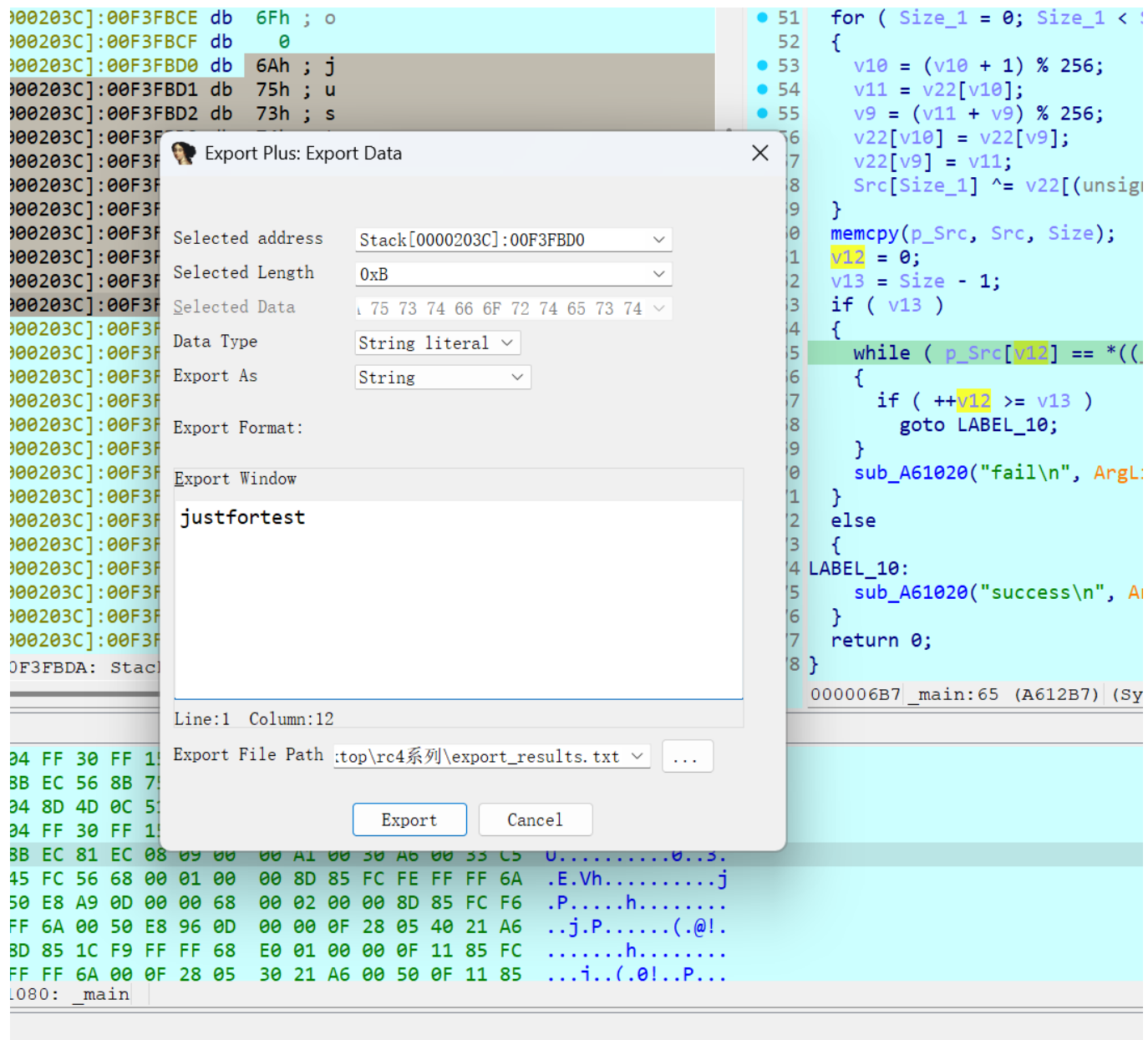
Line:1 Column:62

Export File Path: .top\rc4系列\export_results.txt

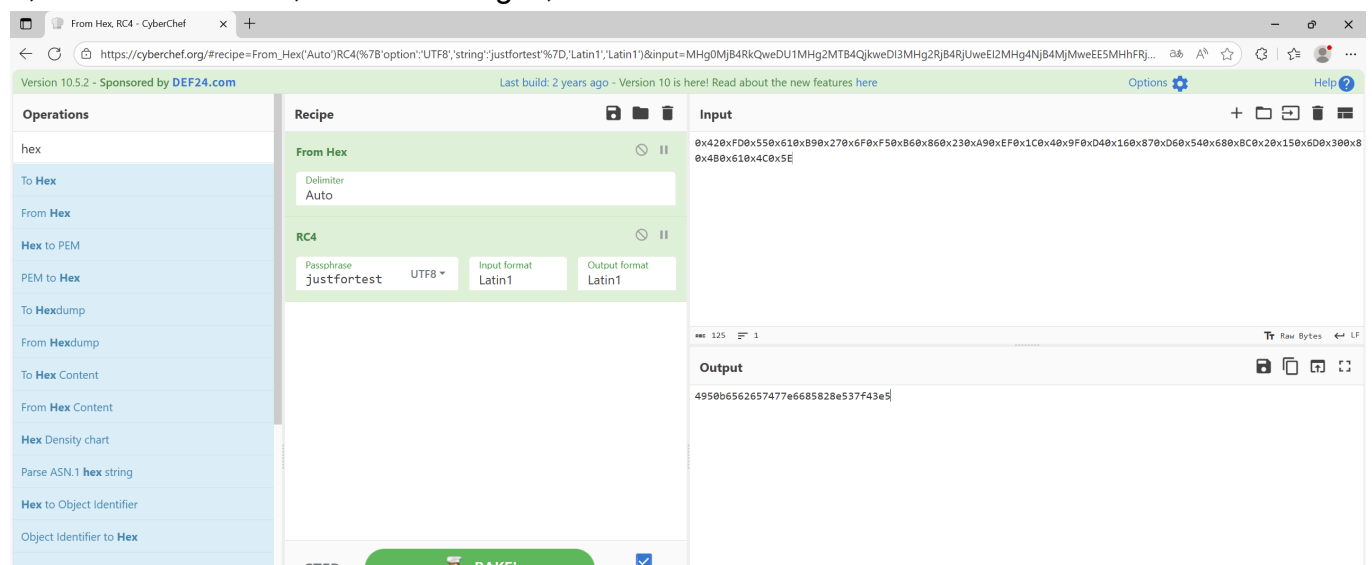
Export Cancel

Stack view

Address	Hex	Dec	Stat
00:0000	00F3F4C0	012698	
01:0004	00F3F4C4	00D840	
02:0008	00F3F4C8	012629	
03:000C	00F3F4CC	000000	
04:0010	00F3F4D0	000000	
05:0014	00F3F4D4	000000	
06:0018	00F3F4D8	000000	
07:001C	00F3F4DC	000000	
08:0020	00F3F4E0	000000	



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

The screenshot displays the Immunity Debugger interface with the following components:

- Function List (Left):** A list of functions in the program, including `sub_401000`, `sub_401010`, `main`, `__security_check_cookie(x)`, `pre_c_initialization(void)`, `sub_401370`, `sub_401378`, `__srt_common_main_seh(void)`, `start`, `__raise_securityfailure`, `__report_gsfailure`, `find_pe_section(uchar * const, uint)`, `__srt_acquire_startup_lock`, `__srt_initialize_crt`, `__srt_initialize_onexit_tables`, `__srt_is_nonwritable_in_current_image`, `__srt_release_startup_lock`, `__srt_uninitialize_crt`, `__onexit`, `__atexit`, `__get_entropy`, `__security_init_cookie`, `UserMathErrorFunction`, `sub_40194F`, `__get_startup_file_mode`, `sub_401959`, `sub_401965`, `__initialize_default_precision`, `__guard_check_icall_nop(x)`, `sub_40198A`, `__srt_initialize_default_local_stdio_o...`, `sub_4019AD`, `sub_4019B9`, `sub_4019BF`, `__srt_fastfail`, `j_UserMathErrorFunction`, and `__srt is managed app`.
- Disassembly Window (Center):** Shows the assembly code for the `main` function. The code includes initialization of `v22`, a `puts` statement, a loop for flag calculation, and a final output statement. The assembly code is as follows:


```

34 n340807546 = 340807546;
35 v22 = -4891;
36 puts("Hello, this is my world.If you want flag, give me something I like.");
37 sub_401010("\n", ArgList);
38 memset(v24, 0, sizeof(v24));
39 gets(v24);
40 v3 = strlen(v24);
41 v4 = strlen(tallmewhy_);
42 memset(v25, 0, sizeof(v25));
43 for ( n256 = 0; n256 < 256; ++n256 )
44 {
45     v27[n256] = n256;
46     v25[n256] = tallmewhy_[n256 % v4];
47 }
48 n256_1 = 0;
49 v7 = 0;
50 do
51 {
52     v8 = v27[n256_1];
53     v7 = (v7 + (char)v25[n256_1] + v8) % 256;
54     v27[n256_1++] = v27[v7];
55     v27[v7] = v8 ^ 0x37;
56 }
57 while ( n256_1 < 256 );
58 sub_401010("\n\n", ArgList_1);
59 v9 = 0;
60 v23 = 0;
61 v10 = 0;
62 if ( v3 )
63 {
64     do
65     {
66         v9 = (v9 + 1) % 256;
67         v11 = v27[v9];
68         v10 = (v11 + v10) % 256;
69         v27[v9] = v27[v10];
70         v27[v10] = v11;
71         v12 = v23;
72         v24[v23] ^= v27[(unsigned __int8)(v11 + v27[v9])];
73         v23 = v12 + 1;
74     }
75     while ( v12 + 1 < v3 );
76     v10 = 0;
77 }

```
- Graph Overview (Bottom):** A visual representation of the program's execution flow, showing a single path through the code.
- Output Window (Bottom):** Displays the output of the program, showing the string "Hello, this is my world.If you want flag, give me something I like." followed by a newline character.

The screenshot displays the Immunity Debugger interface with several windows open:

- IDA View-EIP:** Shows assembly code for the 'main' function. The instruction at address 00401000 is 'push esi', which is highlighted in blue.
- Pseudocode-A:** Shows the corresponding C code. The function 'main' is defined, and it calls 'puts(&v1);' where 'v1' is a string literal 'Hello, this is my world.If you want flag, give me something I'.
- Hex View-1:** Shows the memory dump of the program's memory. The address 00401000 is highlighted, showing the instruction 'push esi'.
- Output:** Shows the program's output, which is 'Hello, this is my world.If you want flag, give me something I'.
- Stack view:** Shows the stack frame for the 'main' function. The return address is '00401000' and the program's entry point is '00401000'.

3.把密钥和密文提取出来,在ida里写个小py脚本

The screenshot shows the IDA Pro interface. On the left, a C code snippet is highlighted:

```
38 memset(v24, 0, sizeof(v24));
39 gets(v24);
40 v3 = strlen(v24);
41 v4 = strlen(tallmewhy_);
42 memset(v25, 0, sizeof(v25));
43 for ( n256 = 0; n256 < 256; ++n256 )
```

In the center, the 'Execute script' dialog box is open. It contains a 'Snippet list' on the left with 'Default snippet' selected. The 'Please enter script body' text area contains the following Python code:

```
1 add=0x4FF7E4
2 enc=[0xF5,0x8C,0x8D,0xE4,0x9F,
3     0xA5,0x28,0x65,0x30,0xF4,0xEB,
4     0xD3,0x24,0xA9,0x91,0x1A,0x6F,0xD4,0x6A,0xD7,0xB,
5     0x8D,0xE8,0xB8,0x83,0x4A,0x5A,0x6E,0xBE,0xCB,
6     0xF4,0x4B,0x99,0xD6,0xE6,0x54,0x7A,0x4F,
7     0x50,0x14,0xE5,0xEC]
8 for i in range(len(enc)):
9     patch_byte(add+i,enc[i])
```

At the bottom of the dialog, the 'Scripting language' is set to 'IDC' and 'Tab size' is set to '4'. There are 'Run', 'Export', and 'Import' buttons.

On the right side of the IDA Pro window, the 'Stack view' is visible, showing memory addresses and their corresponding values in hexadecimal:

Address	Value
004FF778	004
004FF77C	004
004FF780	000
004FF784	000
004FF788	008
004FF78C	008
004FF790	004
004FF794	000
004FF798	000

0 0x01 0x1A 0x65 0xD4 0x6A 0xD7 0xB 0x8D 0xE8 0xB8 0x83 0x4A 0x5A 0x6E 0xBE 0xCB 0x54 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

Stack[00008650]:00F5F5ABdb 0

Stack[00008650]:00F5F5ACdb 66h ; f

Stack[00008650]:00F5F5ADdb 6Ch ; 1

Stack[00008650]:00F5F5AEdb 61h ; a

Stack[00008650]:00F5F5AFdb 67h ; g

Stack[00008650]:00F5F5B0db 7Bh ; {

Stack[00008650]:00F5F5B1db 63h ; c

Stack[00008650]:00F5F5B2db 35h ; 5

Stack[00008650]:00F5F5B3db 65h ; e

Stack[00008650]:00F5F5B4db 30h ; 0

Stack[00008650]:00F5F5B5db 66h ; f

Stack[00008650]:00F5F5B6db 35h ; 5

Stack[00008650]:00F5F5B7db 66h ; f

Stack[00008650]:00F5F5B8db 36h ; 6

Stack[00008650]:00F5F5B9db 2Dh ; -

Stack[00008650]:00F5F5BAdb 66h ; f

Stack[00008650]:00F5F5Bbdb 37h ; 7

Stack[00008650]:00F5F5BCdb 39h ; 9

Stack[00008650]:00F5F5BDdb 65h ; e

Stack[00008650]:00F5F5BEDb 2Dh ; -

Stack[00008650]:00F5F5BFdb 35h ; 5

Stack[00008650]:00F5F5C0db 62h ; b

Stack[00008650]:00F5F5C1db 39h ; 9

Stack[00008650]:00F5F5C2db 62h ; b

Stack[00008650]:00F5F5C3db 30h ; 0

UNKNOWN 00F5F5AC: Stack[00008650]:00F5F5AC (Synchronized with Pseudocode-A)

61v10

62if (

63{

64do

65{

66

67

68

69

70

71

72

73

74}

75wh

76v1

77}

78for

79v1

80v14

81if (

82v1

83sub

84ret

85}

00000

ex View-1

L0B011 85 CC FA FF FF 68 18 21 7D 00 0F 28 05 C0 21

L0C07D 00 0F 11 85 DC FA FF FF C7 85 EC FA FF FF 99

L0D0D6 E6 54 C7 85 F0 FA FF FF 7A 4F 50 14 66 C7 85

L0E0F4 FA FF FF E5 EC FF 15 AC 20 7D 00 68 5C 21 7D

.告...h.!}..(...

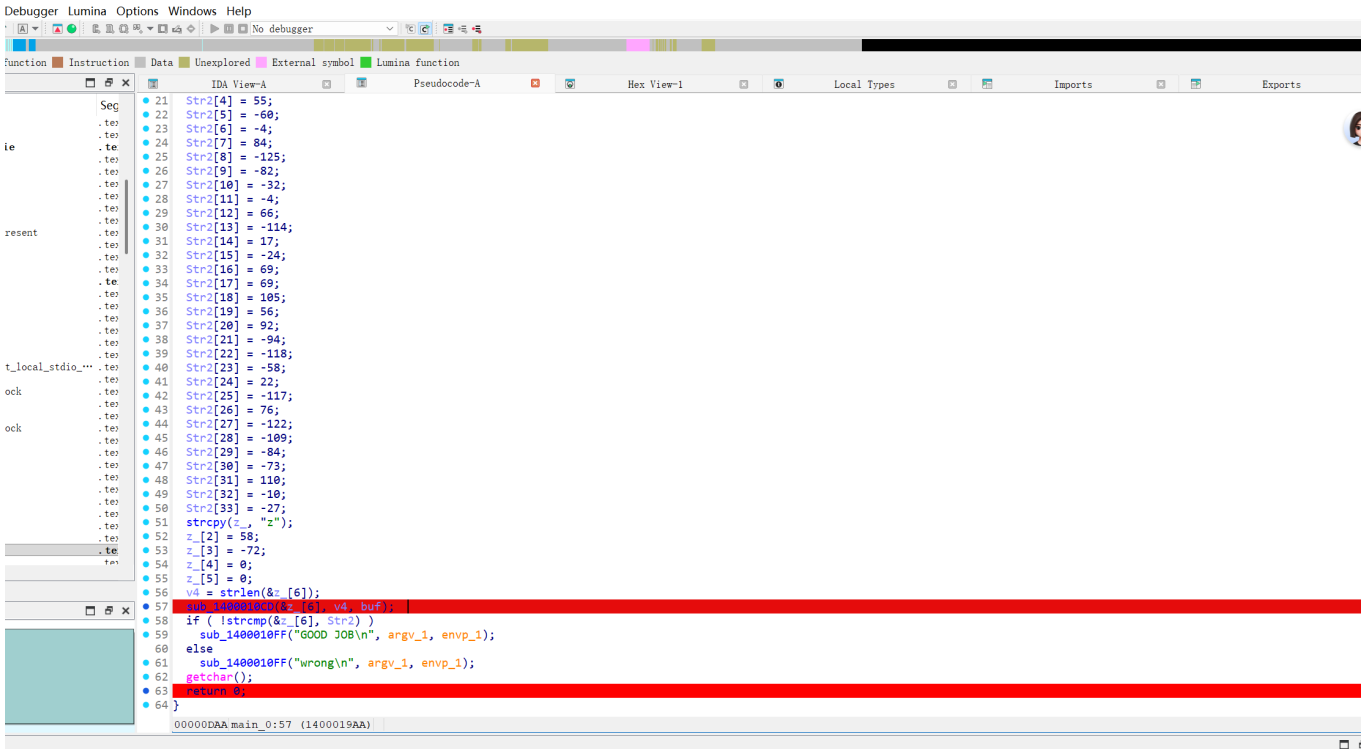
}...哮喘...药·糕...

宙·T药·痰...zOP.f药·

酏...展.....}.h\!}

三,标准rc4 001

1,ida查看分析一下,算了,直接断点调试

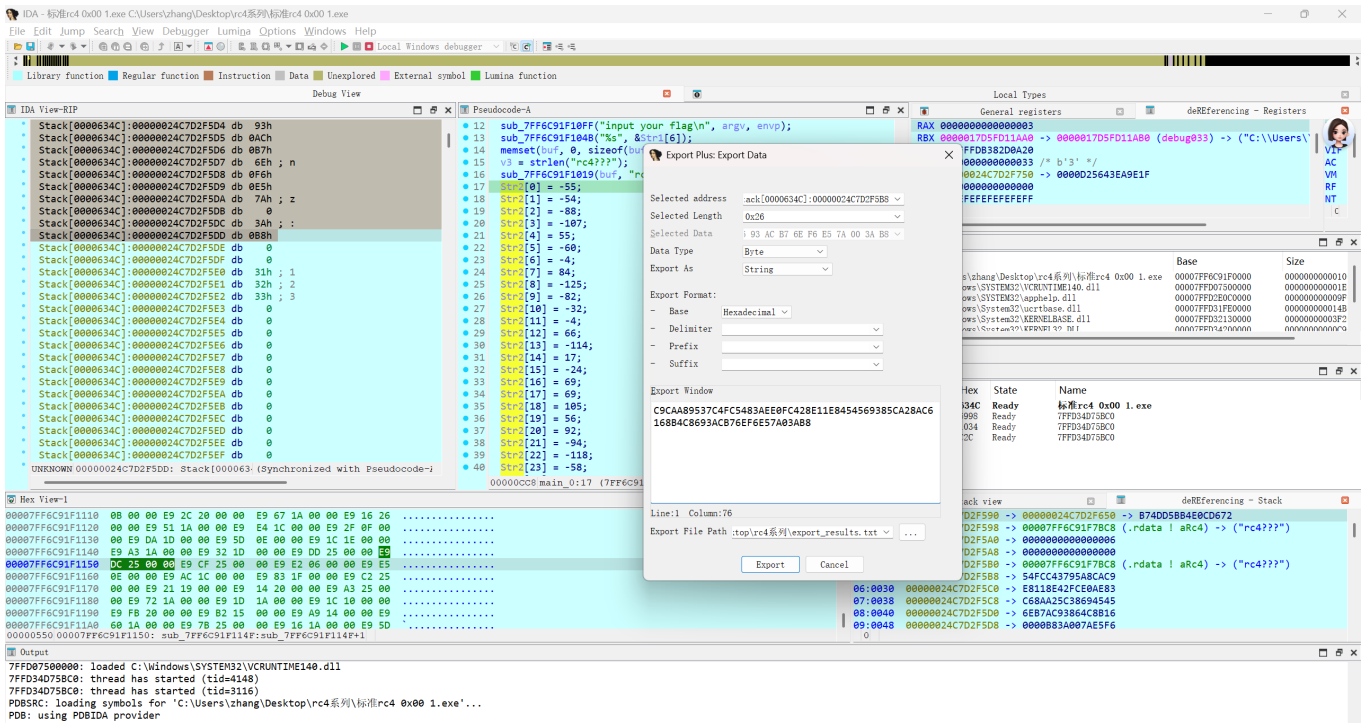


```

1:action 140001048 sub_140001048 was recast to "__int64(const char *p_as, ...)
18" was renamed to "buf_"
2:action 140001019 sub_140001019 was recast to "__int64 __fastcall(_QWORD buf, _QWORD p_rc4???, _QWORD)"
3E" was renamed to "z_"
3:action 1400010CD sub_1400010CD was recast to "int64 __fastcall(_QWORD, _QWORD, _QWORD buf)"

```

2,提取字符串,按标准rc4做一下试试



3,赛博厨子秒了

From_hex(Auto)RC4(7676 option: UTF8, string: rc4??? %7D, latin1, latin1)joinInput=C9CAA89537C4FC5483AEE0FC428E11E8454569385CA28AC6168B4C8693ACB76EF6E57A003AB8

Recipe

From Hex

Delimiter
Auto

RC4

Passphrase
rc4???

UTF8

Input format
Latin1

Output format
Latin1

STEP

BAKE!

Auto Bake

Input

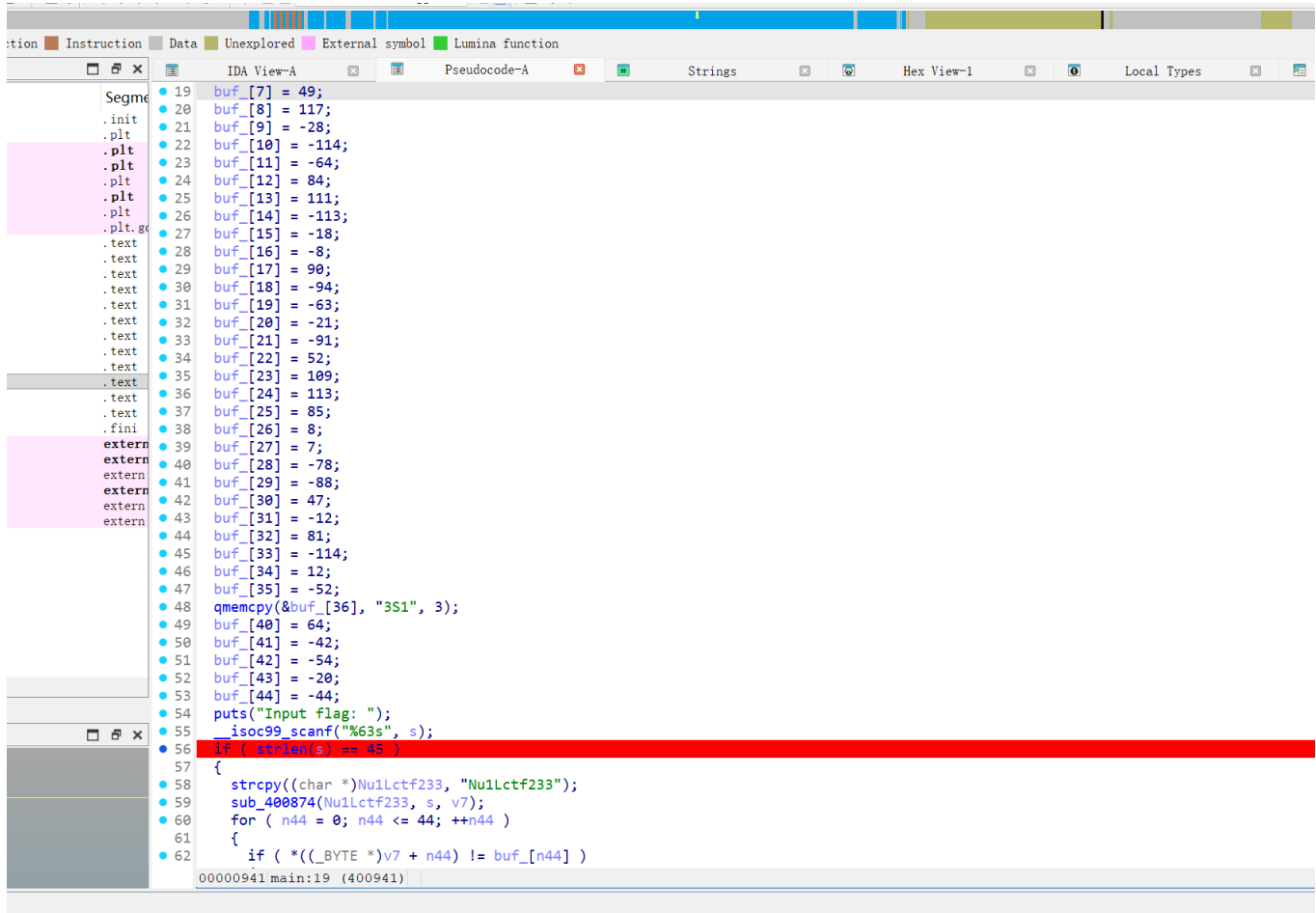
C9CAA89537C4FC5483AEE0FC428E11E8454569385CA28AC6168B4C8693ACB76EF6E57A003AB8

Output

flag{374315ed9864f687d6d5144167944eb8}

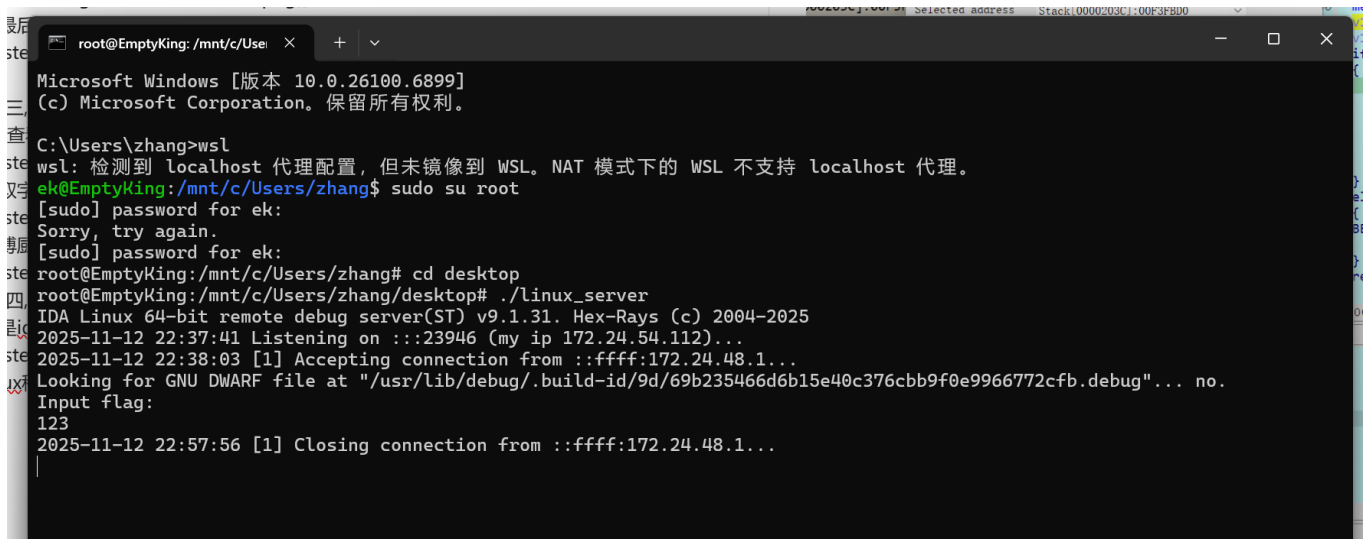
四,标准rc4 002

1,还是ida分析一下,依旧标准rc4

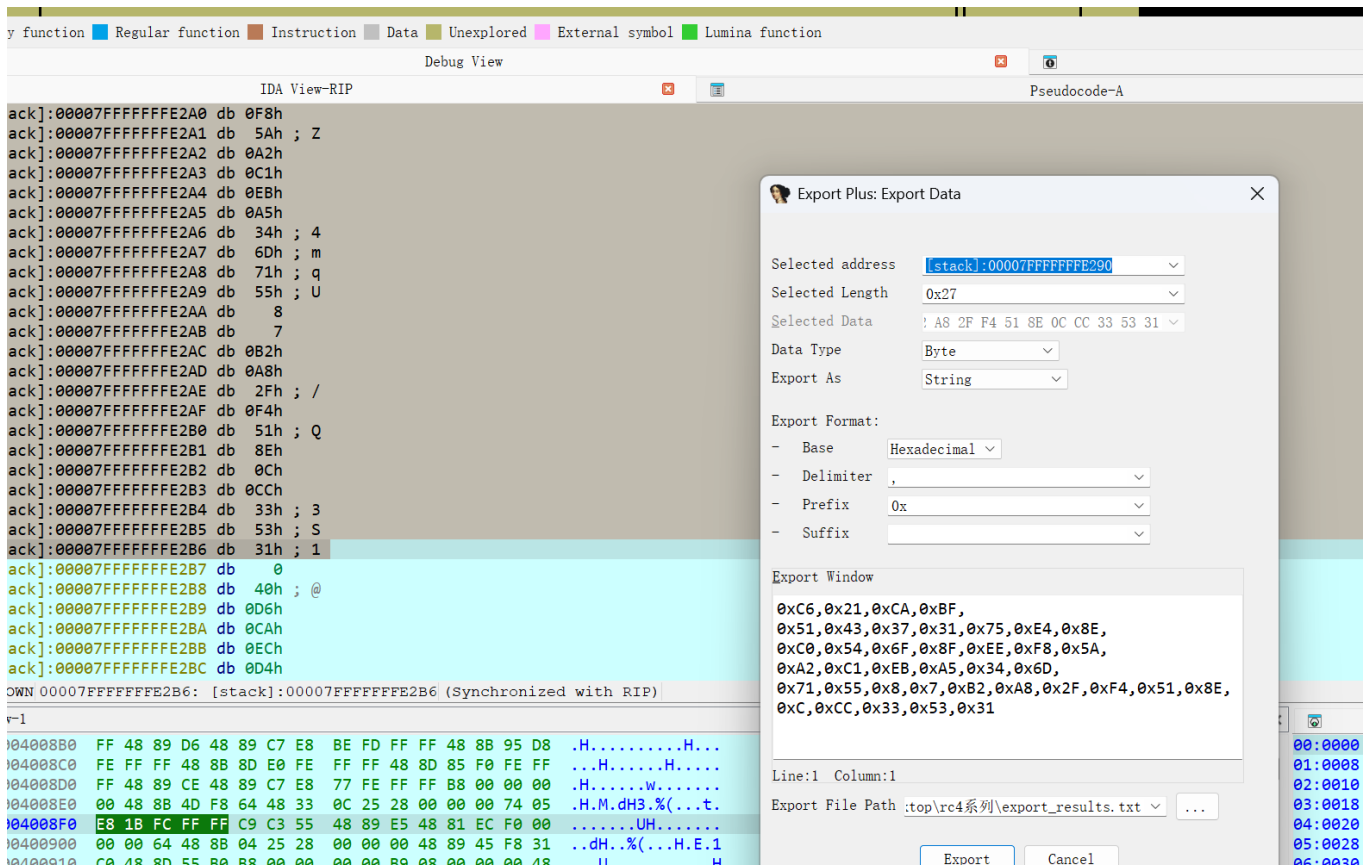


, length)

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
```

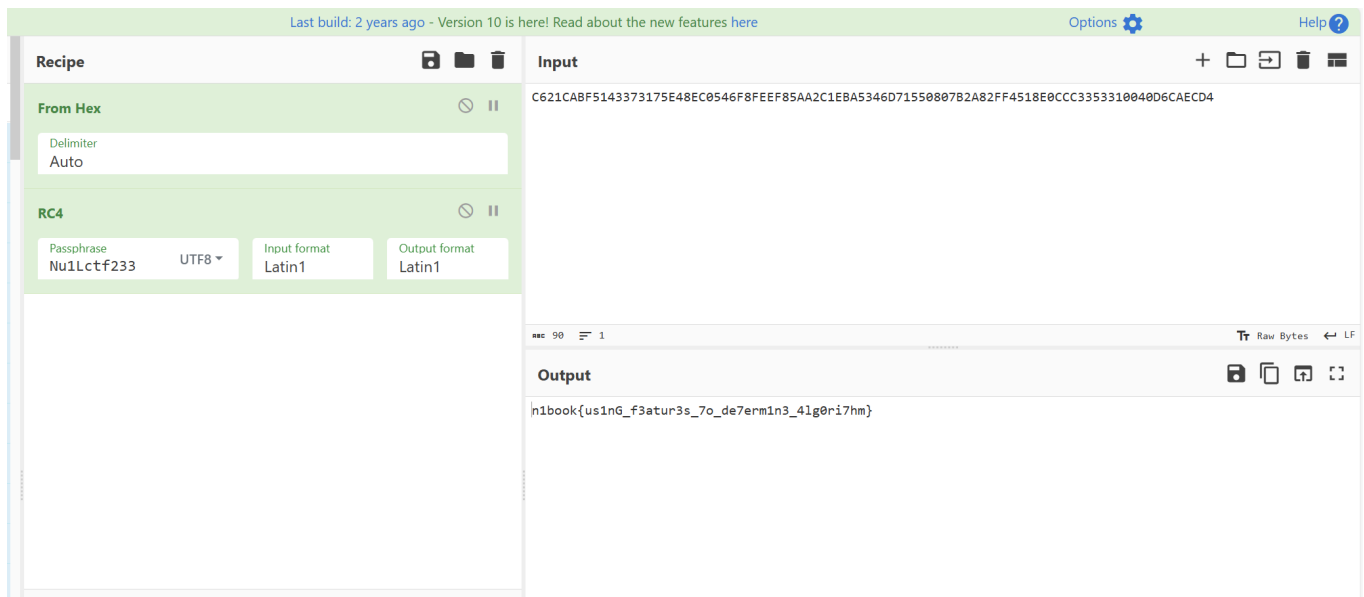
```
fixed_text = re.sub(r'0x([0-9A-Fa-f]{1,2})\b', fix_hex, text)
```

```
# 5 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()

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

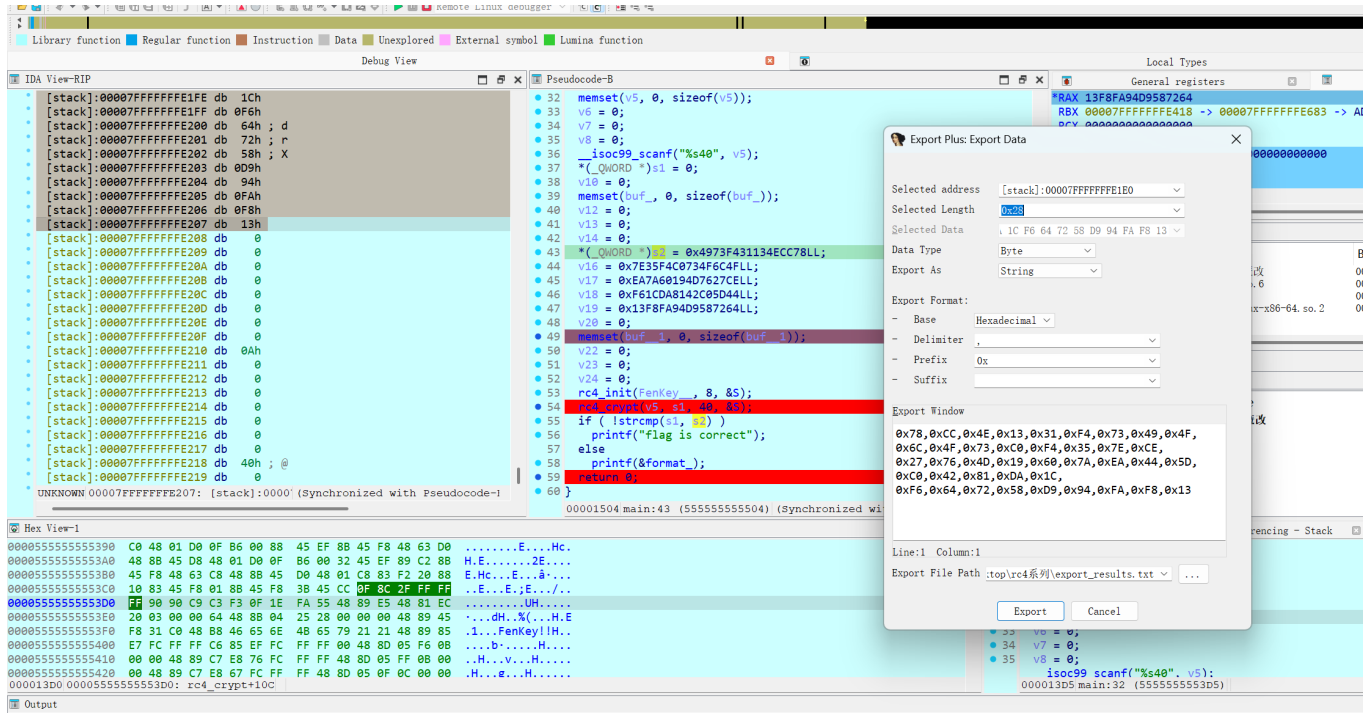
print("✅ 已修正完成并清理所有符号！")
print("📄 输出文件路径: ", output_file)
```

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

29 puts(&s_);
30 puts("rrrrrrcccc4444444!");
31 printf("you input flag:");
32 memset(v5, 0, sizeof(v5));
33 v6 = 0;
34 v7 = 0;
35 v8 = 0;

Execute script

Snippet list

Name

Default snippet

Please enter script body

```
1 add=0x7FFFFFFFDFE0
2 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]
3 for i in range(len(enc)):
4     patch_byte(add+i,enc[i])
```

Line 1 of 1

Scripting language Python Tab size 4 Run Export Import

Stack view

Decimal Hex

490 1EA

000013D5 main:32 (

IDA View-RIP

Debug View

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

00001599 main:54 (555555555599) (Synchronized with IDA View-RIP)

Hex View-1

0000555555555390 C0 48 01 D0 0F B6 00 88 45 EF 8B 45 F8 48 63 D0E....Hc.
00005555555553A0 48 8B 45 D8 48 01 D0 0F B6 00 32 45 EF 89 C2 8B H.E.....2E...
00005555555553B0 45 F8 48 63 C8 48 8B 45 D0 48 01 C8 83 F2 20 88 E.Hc....E...â...