

逆向wp-3

逆向题目wp

张程思

一,1z-RC4

1,ida打开发现一堆爆红,并且发现其16进制形式不太对劲

```
IDA View-A Hex View-1 Local Types
.t3xt:00000014001130C
.t3xt:000000140011311
.t3xt:000000140011311 ; ===== S U B R O U T I N E =====
.t3xt:000000140011311
.t3xt:000000140011311
.t3xt:000000140011311
.t3xt:000000140011311 sub_140011311 proc near ; CODE XREF: sub_140015CB0:loc_140015CDC↓p
.t3xt:000000140011311 push 0FFFFFFF8181EF2Bh
.t3xt:000000140011311 sub_140011311 endp ; sp-analysis failed
.t3xt:000000140011311
.t3xt:000000140011316 ; ===== S U B R O U T I N E =====
.t3xt:000000140011316
.t3xt:000000140011316
.t3xt:000000140011316 sub_140011316 proc near ; CODE XREF: sub_140014AB0+10↓p
.t3xt:000000140011316 push 0FFFFFFF8181EC26h
.t3xt:000000140011316 sub_140011316 endp ; sp-analysis failed
.t3xt:000000140011316
.t3xt:00000014001131B ; ===== S U B R O U T I N E =====
.t3xt:00000014001131B
.t3xt:00000014001131B sub_14001131B proc near ; CODE XREF: sub_140014A90+4↓p
.t3xt:00000014001131B push 0FFFFFFF8181D1B1h
.t3xt:00000014001131B sub_14001131B endp ; sp-analysis failed
.t3xt:00000014001131B
.t3xt:000000140011320 ; ===== S U B R O U T I N E =====
.t3xt:000000140011320
.t3xt:000000140011320
.t3xt:000000140011320 sub_140011320 proc near ; CODE XREF: sub_140016420+2B↓p
.t3xt:000000140011320 push 0FFFFFFF8181D15Ah
.t3xt:000000140011320 push 0FFFFFFF8181EC79h
.t3xt:00000014001132A push 0FFFFFFF8181D930h
.t3xt:00000014001132F push 0FFFFFFF8181D75Dh
.t3xt:000000140011334 push 0FFFFFFF8181C646h
.t3xt:000000140011334 sub_140011320 endp ; sp-analysis failed
.t3xt:000000140011334
.t3xt:000000140011339 ; ===== S U B R O U T I N E =====
.t3xt:000000140011339
.t3xt:000000140011339
.t3xt:000000140011339 sub_140011339 proc near ; CODE XREF: sub_140014D50+E↓p
.t3xt:000000140011339 push 0FFFFFFF8181ECE7h
0000070C 000000014001130C: sub_14001130C (Synchronized with Hex View-1)
```


Address	Length	Type	String
.rdata:00000041	C	00000041	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/-
.rdata:00000009	C	00000009	_ArgList
.rdata:00000006	C	00000006	t3xt
.rdata:00000011	C	00000011	plz input flag:
.rdata:00000015	C	00000015	Failed to read input
.rdata:0000000B	C	0000000B	wan la!!!\n
.rdata:00000020	C	00000020	Jinghong_is_B35T!!! && not you\n
.rdata:00000030	C	00000030	Congratulation!This is the true gift for you!!!
.rdata:00000010	C	00000010	try again!IT_T
.rdata:00000008	C	00000008	WTF!!!!
.rdata:0000001C	C	0000001C	Stack around the variable '
.rdata:00000011	C	00000011	' was corrupted.
.rdata:0000000F	C	0000000F	The variable
.rdata:0000002B	C	0000002B	' is being used without being initialized.
.rdata:000000DD	C	000000DD	The value of ESP was not properly saved across a function call. This is usually a result of calling a function declared with one calling convention with a function pointer declared with a
.rdata:0000011D	C	0000011D	A cast to a smaller data type has caused a loss of data. If this was intentional, you should mask the source of the cast with the appropriate bitmask. For example: \r\n\tchar c = (i & 0
.rdata:0000001D	C	0000001D	Stack memory was corrupted\r\n
.rdata:00000036	C	00000036	A local variable was used before it was initialized\r\n
.rdata:0000002C	C	0000002C	Stack memory around _alloca was corrupted\r\n
.rdata:0000001E	C	0000001E	Unknown Runtime Check Error\r\n
.rdata:00000011	C	00000011	Unknown Filename
.rdata:00000014	C	00000014	Unknown Module Name
.rdata:00000020	C	00000020	Run-Time Check Failure #&d - %s
.rdata:00000026	C	00000026	Stack corrupted near unknown variable
.rdata:00000006	C	00000006	% 2X
.rdata:00000049	C	00000049	Stack area around _alloca memory reserved by this function is corrupted\r\n
.rdata:00000009	C	00000009	\nData: <
.rdata:0000002A	C	0000002A	\nAllocation number within this function:
.rdata:00000008	C	00000008	\nSize:
.rdata:0000000D	C	0000000D	\nAddress: 0x
.rdata:00000048	C	00000048	Stack area around _alloca memory reserved by this function is corrupted
.rdata:0000001A	C	0000001A	%s%s%s%zd%s%s%s%s%s
.rdata:00000034	C	00000034	A variable is being used without being initialized.
.rdata:00000019	C	00000019	Stack pointer corruption
.rdata:0000002A	C	0000002A	Cast to smaller type causing loss of data
.rdata:00000018	C	00000018	Stack memory corruption
.rdata:0000002A	C	0000002A	Local variable used before initialization
.rdata:0000001F	C	0000001F	Stack around _alloca corrupted
.rdata:0000000E	C	0000000E	RegOpenKeyExW
.rdata:00000011	C	00000011	RegQueryValueExW
.rdata:0000000C	C	0000000C	RegCloseKey
.rdata:00000011	C	00000011	PDBOpenValidate5
.rdata:0000002A	C	0000002A	D:\vsTEST\Project1\x64\Debug\Project1.pdb
.data:00000031	C	00000031	8SVCVs/pH1z17NX7wrV+FdMgq4VdH6Tp2kQIckZR1f7NgXSG
.data:00000011	C	00000011	Jinghong_is_B35T
.idata:00000011	C	00000011	GetModuleHandleW
.idata:0000000D	C	0000000D	KERNEL32.dll
.idata:00000015	C	00000015	_C_specific_handler
.idata:0000001E	C	0000001E	_C_specific_handler_noexcept

4,两部分,一个是假的,一个是真的flag

```

IDA View-A Pseudocode-B Pseudocode-A Strings Hex View-1 Local Types Exports
27 sub_1400113B1(&byte_1400250A4);
28 p_Buffer = 0;
29 v11[0] = 2;
30 v11[1] = 0;
31 v11[2] = 2;
32 v11[3] = 4;
33 n64 = 64;
34 sub_14001118D("plz input flag: ");
35 Stream = _acrt_iob_func(0);
36 if ( fgets(Buffer, 256, Stream) )
37 {
38     n36 = j_strlen(Buffer);
39     if ( n36 && v7[n36 + 15] == 10 )
40     {
41         v17 = n36 - 1;
42         if ( n36 - 1 >= 0x100 )
43             sub_1400112DF();
44         Buffer[v17] = 0;
45         --n36;
46     }
47     n16 = 16;
48     if ( n36 == 36 )
49     {
50         sub_1400113A2(Buffer, 36, p_Jinghong_is_B35T, n16); // "J1nghong_is_B35T"
51         Size = 4 * ((n36 + 2) / 3) + 1;
52         Str1 = (char *)malloc(Size);
53         sub_14001100F(Buffer, n36, Str1);
54         if ( !j_strcmp(Str1, Str2) ) // "8SVCVs/pH1z17NX7wrV+FdMgq4VdH6Tp2kQIckZR1f7NgXSG"
55         {
56             sub_14001118D("wan la!!!\n");
57             sub_14001118D("J1nghong_is_B35T!!! && not you\n");
58         }
59         free(Str1);
60         v3 = 0;
61     }
62     else if ( n36 == 40 )
63     {
64         p_Buffer = Buffer;
65         for ( n5 = 0; n5 < 5; ++n5 )
66             sub_140011041(n64, &p_Buffer[8 * n5], v11);
67         sub_1400111E5(p_Buffer, 10);
68         if ( (unsigned int)sub_14001121C(p_Buffer, &qword_140020050) )
69             sub_14001118D("Congratulation!This is the true gift for you!!!");
70         else
71             sub_14001118D("...");
72     }
73 }
00002D96 sub_140013A90:42 (140013B96)

```

```

    v3 = 0;
}
else if ( n36 == 40 )
{
    p_Buffer = Buffer;
    for ( n5 = 0; n5 < 5; ++n5 )
        sub_140011041(n64, &p_Buffer[8 * n5], v11);
    sub_1400111E5(p_Buffer, 10);
    if ( (unsigned int)sub_14001121C(p_Buffer, &qword_140020050) )
        sub_1400111BD("Congratulation!This is the true gift for you!!!");
    else
        sub_1400111BD("try angin!!!T_T");
    v3 = 0xFFFFFFFFLL;
}
else
{
    sub_1400111BD("WTF!!!!");
    v3 = 0xFFFFFFFFLL;
}
}
else
{
    perror("Failed to read input");
    v3 = 1;
}
v4 = v3;
sub_140011348(v6, &unk_14001CE10);
return v4;
}

```

00002F45 sub_140013A90:76 (140013D45)

5,第一个if是rc4加上base64,解出来假的了

```

1  __int64 __fastcall sub_1400136E0(__int64 p_Buffer, unsigned int n36, __int64 p_J1ngho
2  {
3  char *v4; // rdi
4  __int64 n114; // rcx
5  _BYTE v7[32]; // [rsp+0h] [rbp-20h] BYREF
6  char v8; // [rsp+20h] [rbp+0h] BYREF
7  _BYTE v9[276]; // [rsp+30h] [rbp+10h] BYREF
8  int v10; // [rsp+144h] [rbp+124h]
9  int v11; // [rsp+164h] [rbp+144h]
10 int v12; // [rsp+184h] [rbp+164h]
11 unsigned int n36_1; // [rsp+1A4h] [rbp+184h]
12 char v14; // [rsp+1C4h] [rbp+1A4h]
13
14 v4 = &v8;
15 for ( n114 = 114; n114; --n114 )
16 {
17     *(_DWORD *)v4 = -858993460;
18     v4 += 4;
19 }
20 sub_1400113B1(byte_1400250A4);
21 sub_14001130C(v9, p_J1nghong_is_B35T, n16);
22 v10 = 0;
23 v11 = 0;
24 for ( n36_1 = 0; n36_1 < n36; ++n36_1 )
25 {
26     v10 = (v10 + 1) % 256;
27     v11 = ((unsigned __int8)v9[v10] + v11) % 256;
28     v14 = v9[v10];
29     v9[v10] = v9[v11];
30     v9[v11] = v14;
31     v12 = ((unsigned __int8)v9[v11] + (unsigned __int8)v9[v10]) % 256;
32     *(_BYTE *)(p_Buffer + n36_1) ^= v9[v12];
33 }
34 return sub_140011348(v7, &unk_14001CD50);
35 }

```

```

Data Unexplored External symbol Lumina function
IDA View-A Pseudocode-B Pseudocode-A Strings Hex View-1
25 v13 = *(_BYTE *)(n40_1 + p_Buffer);
26 ++n40_1;
27 v9 = v13;
28 if ( n40_1 >= n40 )
29 {
30     v14 = 0;
31 }
32 else
33 {
34     v14 = *(_BYTE *)(n40_1 + p_Buffer);
35     ++n40_1;
36 }
37 v10 = v14;
38 if ( n40_1 >= n40 )
39 {
40     v15 = 0;
41 }
42 else
43 {
44     v15 = *(_BYTE *)(n40_1 + p_Buffer);
45     ++n40_1;
46 }
47 v11 = v15 | (v10 << 8) | (v9 << 16);
48 *(_BYTE *)(v5 + Str1) = abcdefghijklmn[(v11 >> 18) & 0x3F];
49 v6 = v5 + 1;
50 *(_BYTE *)(v6 + Str1) = abcdefghijklmn[(v11 >> 12) & 0x3F];
51 v7 = v6 + 1;
52 if ( v12 <= 1 )
53     n61 = abcdefghijklmn[(v11 >> 6) & 0x3F];
54 else
55     n61 = 61;
56 *(_BYTE *)(v7 + Str1) = n61;
57 v8 = v7 + 1;
58 if ( v12 )
59     n61_1 = 61;
60 else
61     n61_1 = abcdefghijklmn[v11 & 0x3F];
62 *(_BYTE *)(v8 + Str1) = n61_1;
63 v5 = v8 + 1;
64 }
65 result = v5 + Str1;
66 *(_BYTE *)(v5 + Str1) = 0;
67 return result;
68 }

```

=From_Base64('A-Za-z0-9%2B/%3D';true,false)RC4(%7B'option':'UTF8','string':'J1nghong_is_B35T%7D','Latin1','Latin1')&input=OFNDVIRzL3BIMXoxN05YN3dyVitGZE1ncTRWZEg2VHAya1... @

Last build: 2 years ago - Version 10 is here! Read about the new features here Options

Recipe	Input
<p>From Base64 ⏸</p> <p>Alphabet A-Za-z0-9+/= ☑ Remove non-alphabet chars</p> <p><input type="checkbox"/> Strict mode</p> <hr/> <p>RC4 ⏸</p> <p>Passphrase: J1nghong_i... UTF8</p> <p>Input format: Latin1 Output format: Latin1</p>	<p>8SCVTs/pH1z17NX7wrV+FdMgq4VdH6Tp2kQIcKZR1f7NgXSqj</p>
	<p>Output</p> <p>BaseCTF{Th1s_i5_n0T_th3_fl@g_hahaha}</p>

rec 48 1 48

6,第二个if是tea解码,这里把key方向的40字节告诉gpt,还有关键逻辑也扔给他就得出flag

```
0000000140020050 00 00 00 00 00 00 00 00 4A 51 0E 07 08 0F 0E 07 .....Jingnong
0000000140020040 5F 69 73 5F 42 33 35 54 00 00 00 00 00 00 00 00 _is_B35T.....
0000000140020050 36 C4 05 33 B4 C3 BC E9 B8 9B 05 2B F4 41 79 85 6..3.ü·鶴··+. ....
0000000140020060 C6 26 1D E0 C9 FB 17 34 AB C5 90 D3 C1 00 21 15 .....4.Ó·...!.
0000000140020070 88 D3 BE 04 C7 98 B0 DE 00 00 00 00 00 00 00 00 .*.ú·.....
0000000140020080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000000140020090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000001400200A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000001400200B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000001400200C0 32 A2 DF 2D 99 2B 00 00 00 00 00 00 00 00 00 00 2.....+. ....
00000001400200D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000001400200E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000001400200F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

7,解题脚本如下

```
# We'll decrypt the 40-byte block you provided using the reverse of the TEA-
like algorithm
# described in the conversation. This code prints the recovered 40-byte flag
(bboth hex and UTF-8 if possible)
# and also shows the intermediate uint32 words for inspection.
import struct

# raw bytes from your memory dump lines (addresses + bytes)
hex_lines = [
    "36 C4 05 33 B4 C3 BC E9 B8 9B 05 2B F4 41 79 85",
    "C6 26 1D E0 C9 FB 17 34 AB C5 90 D3 C1 00 21 15",
    "88 D3 BE 04 C7 98 B0 DE 00 00 00 00 00 00 00 00"
]

# build bytearray
b = bytearray()
for line in hex_lines:
    parts = line.split()
    for p in parts:
        b.append(int(p, 16))

# verify length
assert len(b) == 40, f"expected 40 bytes, got {len(b)}"

# parse as 10 little-endian uint32 words
words_le = [struct.unpack_from("<I", b, i*4)[0] for i in range(10)]

def byteswap32(x):
    return ((x & 0xFF) << 24) | ((x & 0xFF00) << 8) | ((x & 0xFF0000) >> 8) |
    ((x >> 24) & 0xFF)

# apply byteswap to each word (this reverses the byteswap done in the program)
words_after_byteswap = [byteswap32(w) for w in words_le]
```

```

# TEA-like decryption parameters (derived from the disassembly)
DELTA = 1640531527 # 0x61C886F7
ROUNDS = 64
KEY = [2, 0, 2, 4]

mask32 = 0xFFFFFFFF

def decrypt_block(v0, v1, rounds=ROUNDS):
    # v0,v1 are unsigned 32-bit integers (the encrypted outputs after the
    forward algorithm)
    # Start v7 = -DELTA * rounds (mod 2^32)
    v7 = (-DELTA * rounds) & mask32
    # Work on local copies
    v5 = v0 & mask32
    v6 = v1 & mask32
    for _ in range(rounds):
        # undo v6 += (...) ^ (...) ^ 0x99 (note order in original)
        idx = ((v7 >> 11) & 3)
        part = ((KEY[idx] + v7) & mask32) ^ ( (v5 + (((v5 >> 5) ^ (16 * v5)) &
mask32)) & mask32) ^ 0x99
        v6 = (v6 - part) & mask32
        # undo v7 -= DELTA -> v7 = v7 + DELTA
        v7 = (v7 + DELTA) & mask32
        # undo v5 += (...) ^ (...) ^ 0x88
        idx2 = (v7 & 3)
        part2 = ((v7 + (KEY[idx2] & mask32) + 102) & mask32) ^ ( (v6 + (((v6
>> 5) ^ (16 * v6)) & mask32)) & mask32) ^ 0x88
        v5 = (v5 - part2) & mask32
    return v5, v6

# Decrypt all 5 blocks (each block = two uint32 words)
plaintext_words = []
for i in range(0, 10, 2):
    enc_v0 = words_after_byteswap[i]
    enc_v1 = words_after_byteswap[i+1]
    dec_v0, dec_v1 = decrypt_block(enc_v0, enc_v1)
    plaintext_words.extend([dec_v0, dec_v1])

# pack plaintext words back into bytes little-endian
plaintext_bytes = bytearray()
for w in plaintext_words:
    plaintext_bytes += struct.pack("<I", w & mask32)

# Output results
print("Input bytes (40):", b.hex())

```

```

print("Words (little-endian read from memory):")
for i, w in enumerate(words_le):
    print(f" w[{i:02}] = 0x{w:08X}")
print("\nAfter applying byteswap (to undo program's byteswap):")
for i, w in enumerate(words_after_byteswap):
    print(f" wbs[{i:02}] = 0x{w:08X}")

print("\nDecrypted words (uint32 little-endian):")
for i, w in enumerate(plaintext_words):
    print(f" p[{i:02}] = 0x{w:08X}")

print("\nRecovered 40 bytes (hex):", plaintext_bytes.hex())
# Try to decode as utf-8 / printable ASCII
try:
    s = plaintext_bytes.decode('utf-8')
    print("\nRecovered as UTF-8 string:")
    print(s)
except Exception as e:
    # show printable representation
    printable = ''.join(chr(c) if 32 <= c < 127 else f"\\x{c:02x}" for c in
plaintext_bytes)
    print("\nRecovered bytes contain non-UTF8/Non-printable bytes. Printable
view:")
    print(printable)

# Also show as possible flag candidate with escape sequences
print("\nPrintable/escaped:", repr(plaintext_bytes))

```

8,flag如下

```
BaseCTF{Th1s_1s_A_g1ft_fr0m_NshIdE_4U}~~
```

这后面做几个简单的,尝试一下不用ai写脚本(随便找的简单题)

二,exe1

1,无壳,ida查看分析

```
IDA View-A Pseudocode-A Hex View-1 Local
1 int __fastcall main(int argc, const char **argv, const char **envp)
2 {
3     int n3; // [rsp+2Ch] [rbp-4h]
4
5     sub_401800(argc, argv, envp);
6     puts("<--- moectf2021 --->");
7     puts("[CheckIn] Welcome to moectf2021.");
8     puts("This is a really eazy chall. I believe it is easy for you to get the flag");
9     puts("Now input your flag and I will check it : ");
10    scanf("%100s", &unk_407040);
11    for ( n3 = 0; n3 <= 3; ++n3 )
12        ((void (__fastcall *) (void *))funcs_4016DE[n3])(&unk_407040);
13    if ( byte_403020 )
14    {
15        puts("Congratulations!!!");
16        puts("See you next chall!!!");
17    }
18    else
19    {
20        puts("Something went wrong! QwQ");
21        puts("Try againt!!!");
22    }
23    puts(&Buffer_);
24    getchar();
25    getchar();
26    return 0;
27 }
```

2,点开exe试一下,发现提示有长度不对,且告诉了我们flag头

```
C:\Users\zhang\Desktop\第三 x + v
<--- moectf2021 --->
[CheckIn] Welcome to moectf2021.
This is a really eazy chall. I believe it is easy for you to get the flag
Now input your flag and I will check it :
123
Wrong length!
In this game, every flag begin with moectf{
where is your '}' ???
Wrong flag!
Something went wrong! QwQ
Try againt!!!
按任意键以继续
```

3,查看关键函数,发现需要长度为58

```
.data:0000000000403028 off_403028 dq offset aMoectfW31c0meT
.data:0000000000403028 ; DATA XREF: sub_401606:loc_40161E
.data:0000000000403028 ; "moectf{W31C0Me_t0_m03CTF_2021_w
.data:0000000000403030 align 20h
.data:0000000000403040 funcs_4016DE dq offset sub_401550 ; DATA XREF: main+66↑o
.data:0000000000403040 ; main+6D↑r
.data:0000000000403048 dq offset sub_401585
.data:0000000000403050 dq offset sub_4015C5
.data:0000000000403058 dq offset sub_401606
.data:0000000000403060 off_403060 dq offset aWend_402F80 ; DATA XREF: sub_401750:4↑n
```

```

IDA View-A      Pseudocode-C      Pseudocode-B
1 int __fastcall sub_401550(const char *a1)
2 {
3     size_t n58; // rax
4
5     n58 = strlen(a1);
6     if ( n58 != 58 )
7     {
8         LODWORD(n58) = puts("Wrong length!");
9         byte_403020 = 0;
10    }
11    return n58;
12 }

```

4. 闹麻了,我是傻逼,查看string页面直接出来flag了

Data
Unexplored
External symbol
Lumina function

IDA View-A
Pseudocode-D
Strings
Pseudocode-C
Pseudocode-B
Hex View-1

Address	Length	Type	String
.rdata:0...	0000000E	C	Wrong length!
.rdata:0...	00000008	C	moectf{
.rdata:0...	0000002C	C	In this game, every flag begin with moectf{
.rdata:0...	00000016	C	where is your '}' ???
.rdata:0...	0000003B	C	moectf{W31C0Me_t0_m03CTF_2021_w0o0c0o0o0o0o0o0o0o0o0!!!}
.rdata:0...	0000000C	C	Wrong flag!
.rdata:0...	00000017	C	<--- moectf2021 --->
.rdata:0...	00000022	C	[CheckIn] Welcome to moectf2021.
.rdata:0...	0000004A	C	This is a really eazy chall. I believe it is easy for you to get the flag
.rdata:0...	0000002B	C	Now input your flag and I will check it :
.rdata:0...	00000006	C	%100s
.rdata:0...	00000013	C	Congratulations!!!

三,exe2

1,无壳,ida查看分析,单纯的异或,循环五次,异或五次,注意数组是dword形式,

```
on Data Unexplored External symbol Lumina function
IDA View-A Pseudocode-A Hex View-1
1 int __fastcall main(int argc, const char **argv, const char **envp)
2 {
3     _BYTE v4[36]; // [rsp+20h] [rbp-30h] BYREF
4     int n5; // [rsp+44h] [rbp-Ch]
5     _DWORD *v6; // [rsp+48h] [rbp-8h]
6
7     _main();
8     puts("please input your flag");
9     scanf("%s", v4);
10    v6 = v4;
11    for ( n5 = 0; n5 <= 5; ++n5 )
12    {
13        if ( *v6 != (enc[n5] ^ 0x12345678) )
14        {
15            printf("Data3rr0r!");
16            exit(0);
17        }
18        ++v6;
19    }
20    printf("you are right!");
21    return 0;
22 }
```

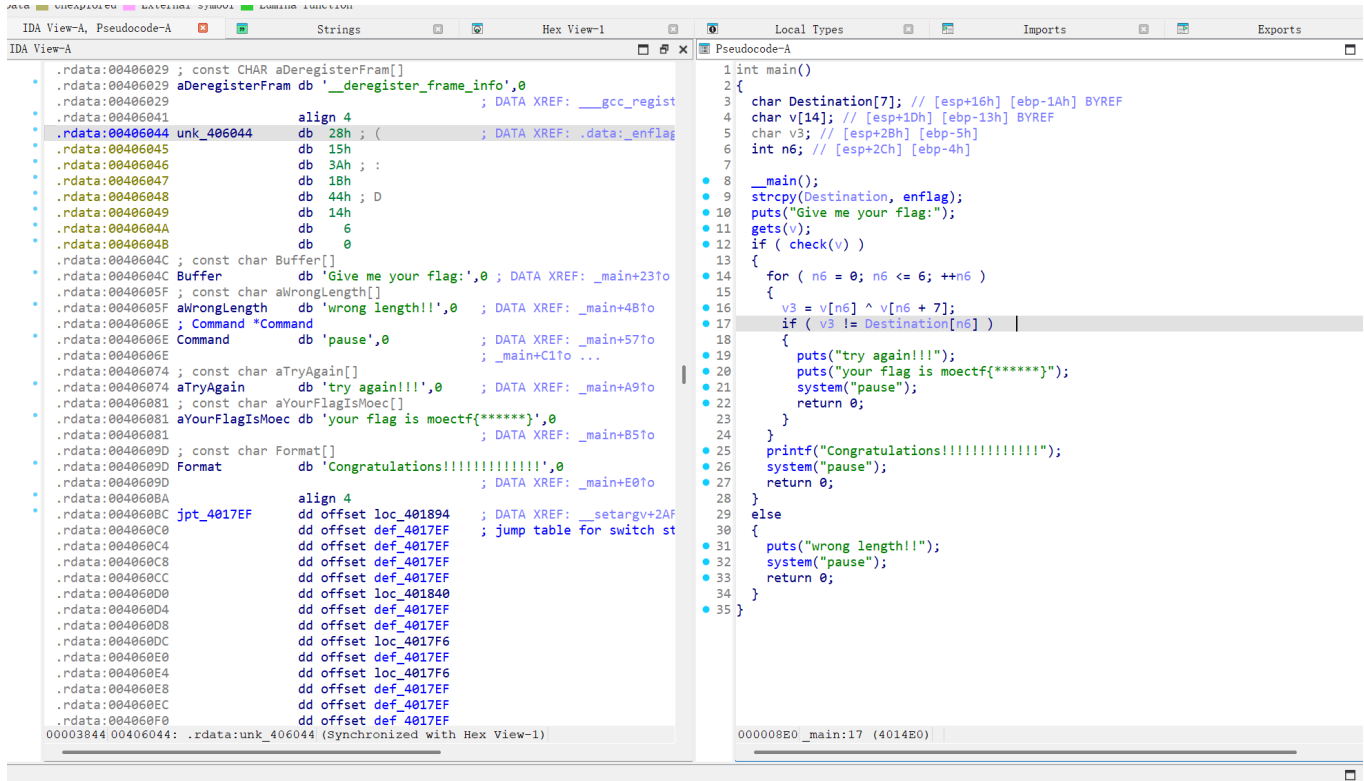
```
-0000000000000010 // padding byte
-000000000000000F // padding byte
-000000000000000E // padding byte
-000000000000000D // padding byte
-000000000000000C _DWORD var_C;
-0000000000000008 _QWORD var_8;
:0000000000000000 _QWORD saved_register_
```

2,脚本如下,这个还挺好写的喵

```
enc=[0x51670536, 0x5E4F102C, 0x7E402211, 0x7C71094B, 0x7C553F1C,0x6F5A3816]
b=""
for i in range(len(enc)):
    a= enc[i] ^ 0x12345678
    b=a.to_bytes(4,byteorder='little').decode()
    print(b)
```

四,exe3

1,无壳,ida查看分析,提取出来密文,关键逻辑还是循环异或



2,脚本如下

```
flag='moectf{'
enc=[0x28,0x15,0x3A,0x1B,0x44,0x14,0x6]
a=[]
for i in range(len(flag)):
    a.append(ord(flag[i]))

for i in range(len(enc)):
    a.append(enc[i])
print(a)
c=''
for i in range(len(a)):
    try:
        d=a[i]^a[i+7]
        c+=chr(d)
    except:
        break

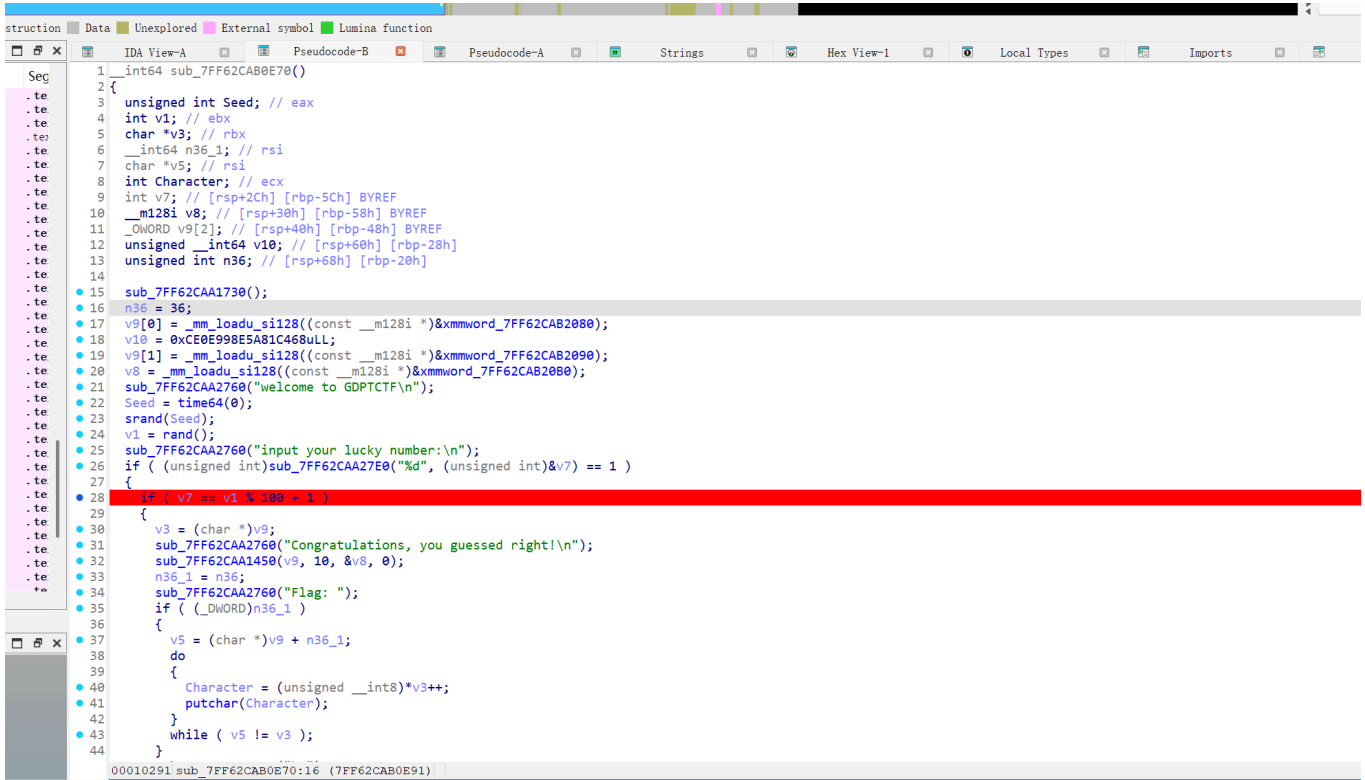
print(flag+c)
```

四,调试1

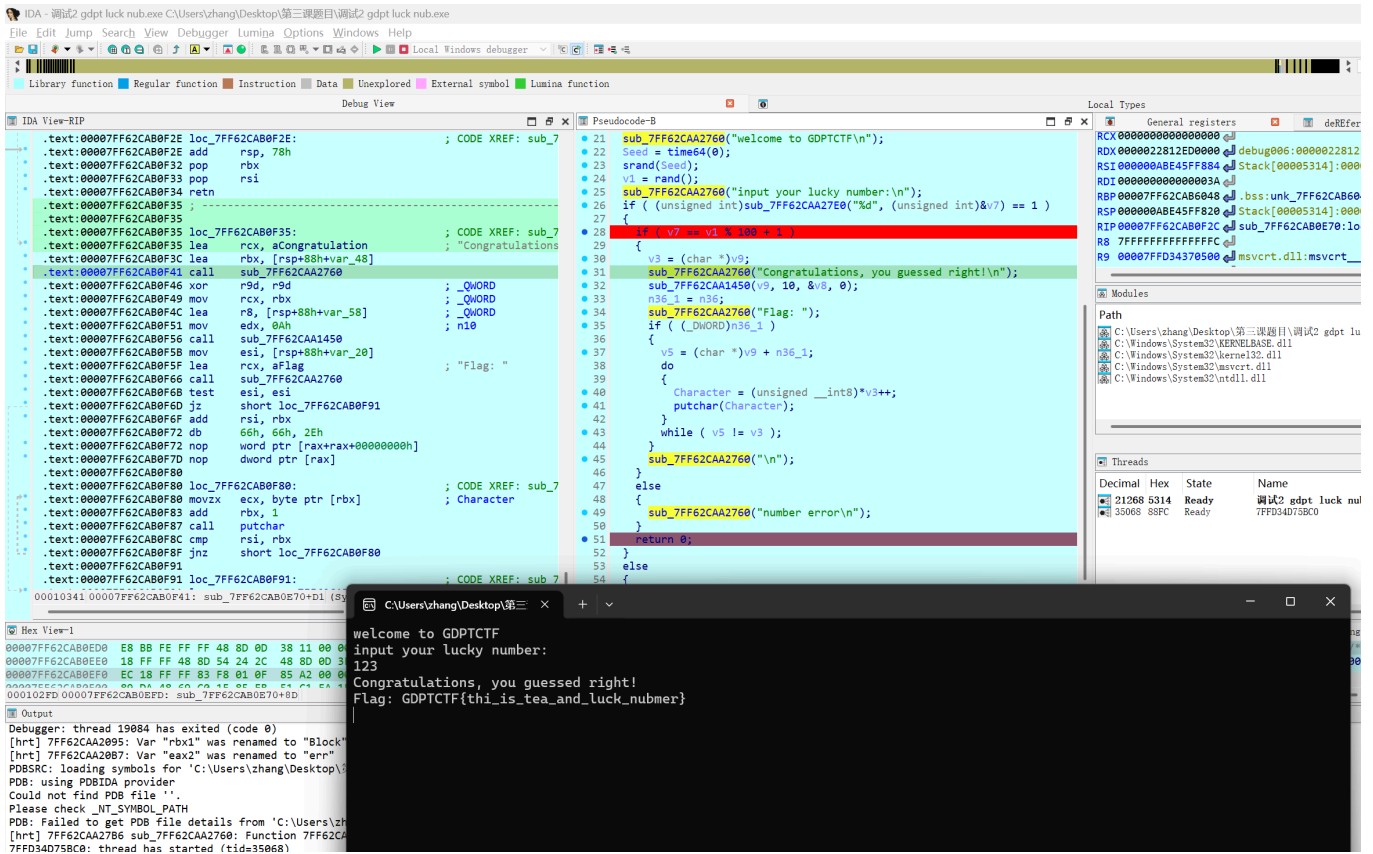
1,无壳,ida查看发现没有main函数,于是去string里找一下,找到了但是发现需要动调,先设一个断点

Seg	Address	Length	Type	String
.te	.rdata:00000014	00000014	C	welcome to GDPTCTF\n
.te	.rdata:0000001A	0000001A	C	input your lucky number:\n
.te	.rdata:0000000F	0000000F	C	invalid input\n
.te	.rdata:00000025	00000025	C	Congratulations, you guessed right!\n
.te	.rdata:00000007	00000007	C	Flag:
.te	.rdata:0000000E	0000000E	C	number error\n
.te	.rdata:0000001F	0000001F	C	Argument domain error (DOMAIN)
.te	.rdata:0000001C	0000001C	C	Argument singularity (SIGN)
.te	.rdata:00000020	00000020	C	Overflow range error (OVERFLOW)
.te	.rdata:00000025	00000025	C	Partial loss of significance (PLOSS)
.te	.rdata:00000023	00000023	C	Total loss of significance (TLOSS)
.te	.rdata:00000036	00000036	C	The result is too small to be represented (UNDERFLOW)
.te	.rdata:0000000E	0000000E	C	Unknown error
.te	.rdata:0000002B	0000002B	C	_matherr(): %s in %s(%g, %g) (retval=%g)\n
.te	.rdata:0000001C	0000001C	C	Mingw-w64 runtime failure:\n
.te	.rdata:00000020	00000020	C	Address %p has no image-section
.te	.rdata:00000031	00000031	C	VirtualQuery failed for %d bytes at address %p
.te	.rdata:00000027	00000027	C	VirtualProtect failed with code 0x%x
.te	.rdata:00000032	00000032	C	Unknown pseudo relocation protocol version %d.\n
.te	.rdata:0000002A	0000002A	C	Unknown pseudo relocation bit size %d.\n
.te	.rdata:00000053	00000053	C	%d bit pseudo relocation at %p out of range, targeting %p, yielding the value %p.\n
.te	.rdata:00000006	00000006	C	(nil)
.te	.rdata:00000006	00000006	C	inited
.te	.rdata:00000007	00000007	C	(null)
.te	.rdata:00000009	00000009	C	Infinity
.te	.rdata:00000014	00000014	C	0123456789abcdefNaN
.te	.rdata:00000009	00000009	C	Infinity
.te	.rdata:00000009	00000009	C	infinity
.te	.rdata:00000009	00000009	C	INFINITY
.te	.rdata:00000006	00000006	C	inited
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0
.te	.rdata:0000002B	0000002B	C	GCC: (Rev5, Built by MSYS2 project) 15.1.0

```
.rdata:00007FF62CAB202E ; const char p_d[]
.rdata:00007FF62CAB202E p_d db '%d',0 ; DATA XREF: sub_7FF62CAB0E70+7810
.rdata:00007FF62CAB2031 aInvalidInput db 'invalid input',0Ah,0
.rdata:00007FF62CAB2031 ; DATA XREF: sub_7FF62CAB0E70:loc_7FF62CAB0F9F10
.rdata:00007FF62CAB2040 aCongratulation db 'Congratulations, you guessed right!',0Ah,0
.rdata:00007FF62CAB2040 ; DATA XREF: sub_7FF62CAB0E70:loc_7FF62CAB0F3510
.rdata:00007FF62CAB2065 aFlag db 'Flag: ',0 ; DATA XREF: sub_7FF62CAB0E70+EF10
.rdata:00007FF62CAB206C asc_7FF62CAB206C db 0Ah,0 ; DATA XREF: sub_7FF62CAB0E70:loc_7FF62CAB0F9110
.rdata:00007FF62CAB206E aNumberError db 'number error',0Ah,0 ; DATA XREF: sub_7FF62CAB0E70+B010
.rdata:00007FF62CAB207C align 20h
.rdata:00007FF62CAB2080 xmmword_7FF62CAB2080 xmmword 64CF55F5B2DEA67C2B28BE96A43DA9Dh
.rdata:00007FF62CAB2080 ; DATA XREF: sub_7FF62CAB0E70+B110
.rdata:00007FF62CAB2090 xmmword_7FF62CAB2090 xmmword 0B008EF178B69A237A2D216E511F93DB3h
.rdata:00007FF62CAB2090 ; DATA XREF: sub_7FF62CAB0E70+2E10
.rdata:00007FF62CAB20A0 qword_7FF62CAB20A0 dq 0CE0E998E5A81C468h
.rdata:00007FF62CAB20A0 ; DATA XREF: sub_7FF62CAB0E70+1310
.rdata:00007FF62CAB20A8 align 10h
.rdata:00007FF62CAB20B0 xmmword_7FF62CAB20B0 xmmword 42675F33323123696C6F346755676979h
.rdata:00007FF62CAB20B0 ; DATA XREF: sub_7FF62CAB0E70+4010
.rdata:00007FF62CAB20C0 off_7FF62CAB20C0 dq offset TlsCallback_0
.rdata:00007FF62CAB20C0 ; DATA XREF: .rdata:off_7FF62CAB2C8010
.rdata:00007FF62CAB20C8 align 20h
.rdata:00007FF62CAB20E0 TlsDirectory dq offset TlsStart
.rdata:00007FF62CAB20E8 TlsEnd_ptr dq offset TlsEnd
.rdata:00007FF62CAB20F0 TlsIndex_ptr dq offset TlsIndex
.rdata:00007FF62CAB20F8 TlsCallbacks_ptr dq offset TlsCallbacks
.rdata:00007FF62CAB2100 TlsSizeOfZeroFill dd 0
.rdata:00007FF62CAB2104 TlsCharacteristics dd 0
.rdata:00007FF62CAB2108 align 20h
.rdata:00007FF62CAB2120 aArgumentDomain db 'Argument domain error (DOMAIN)',0
```

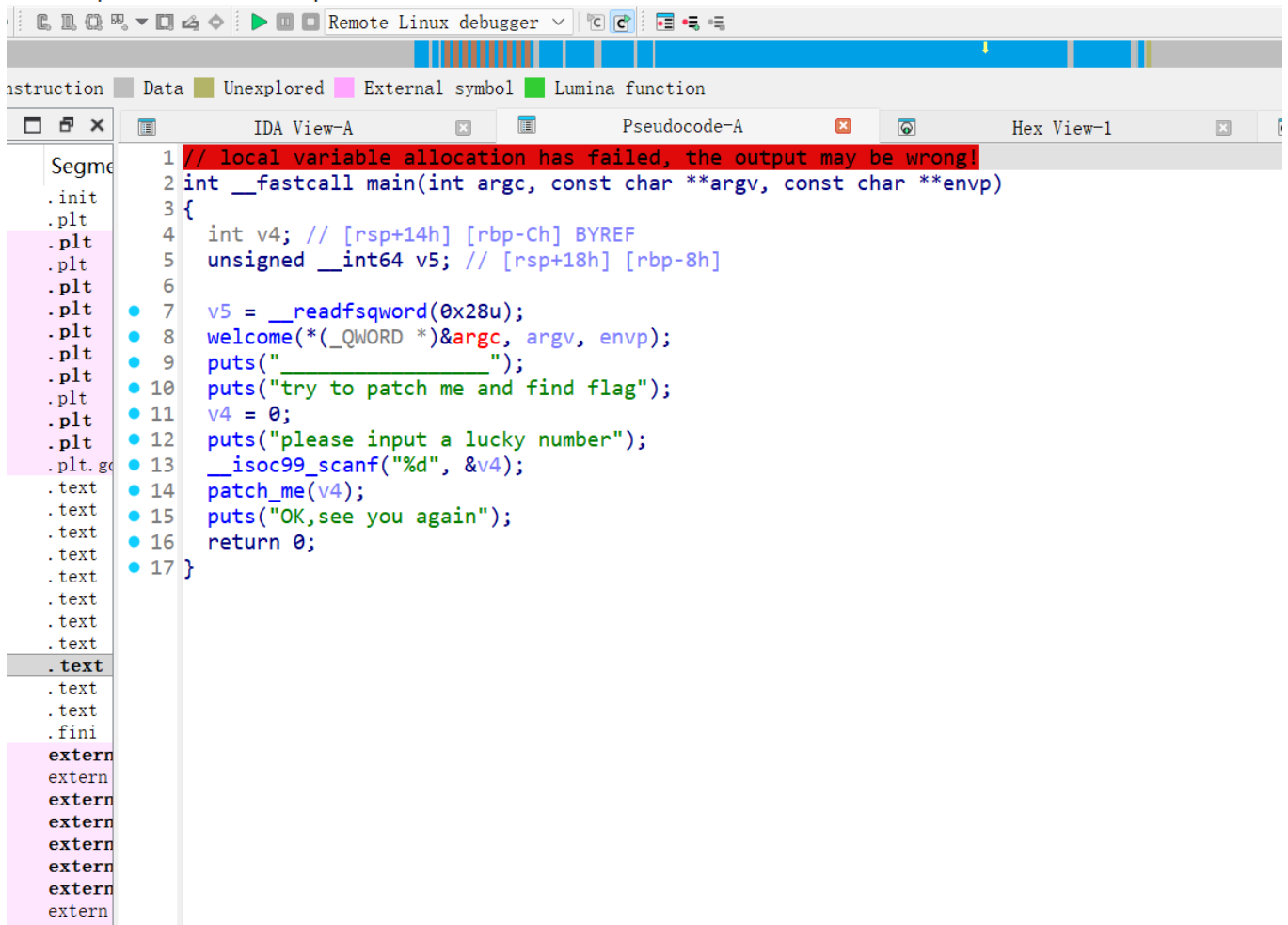


2,本地调试之后,改个ip,设置俩断点就行,但是改ip弄了一会,动调还是不熟练

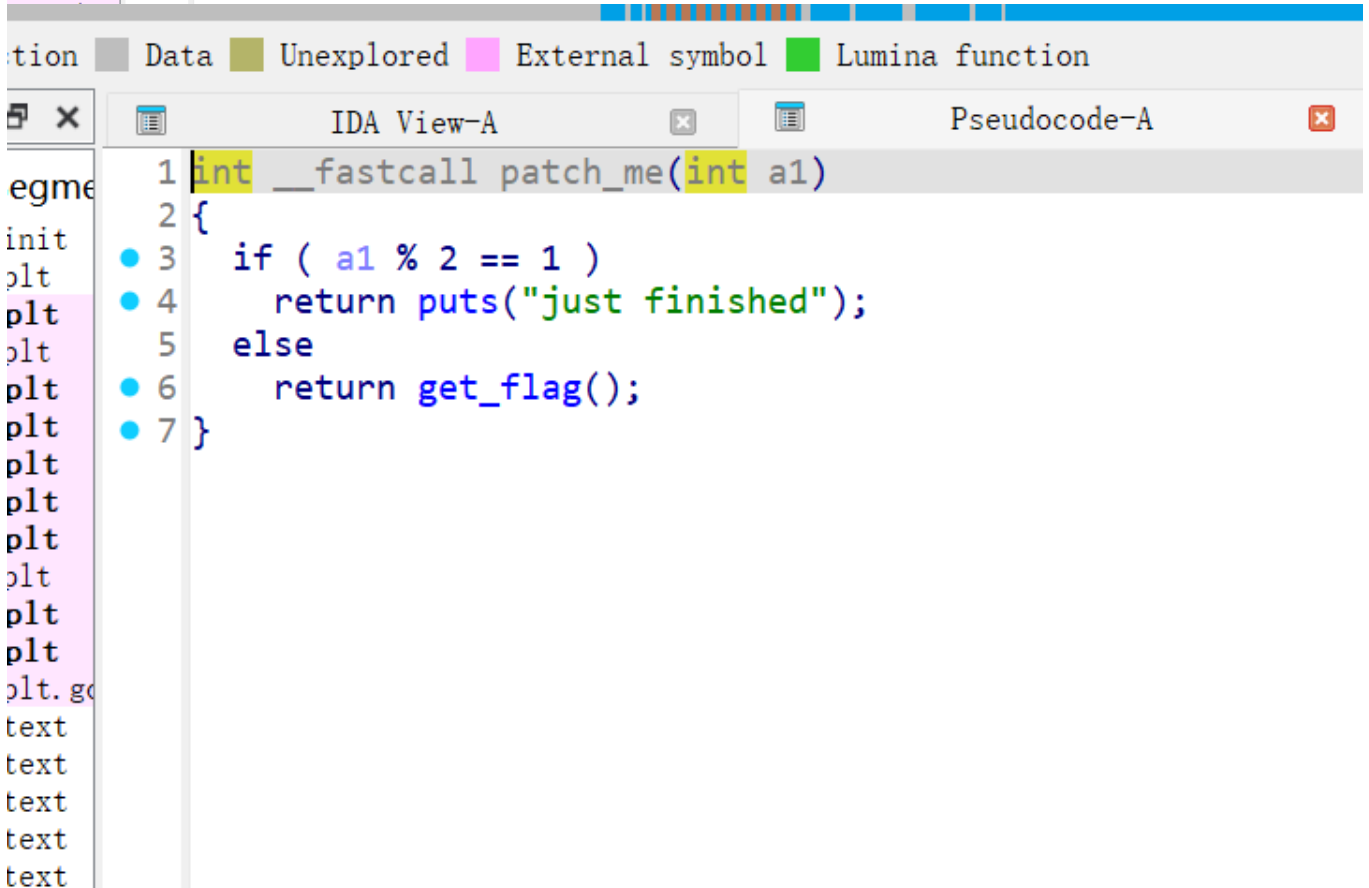


五,调试2

1,无壳,ida查看,还是要调试,由于是linux程序,我用wsl进行调试



```
1 // local variable allocation has failed, the output may be wrong!
2 int __fastcall main(int argc, const char **argv, const char **envp)
3 {
4     int v4; // [rsp+14h] [rbp-Ch] BYREF
5     unsigned __int64 v5; // [rsp+18h] [rbp-8h]
6
7     v5 = __readfsqword(0x28u);
8     welcome((_QWORD *)&argc, argv, envp);
9     puts(" ");
10    puts("try to patch me and find flag");
11    v4 = 0;
12    puts("please input a lucky number");
13    __isoc99_scanf("%d", &v4);
14    patch_me(v4);
15    puts("OK,see you again");
16    return 0;
17 }
```



```
1 int __fastcall patch_me(int a1)
2 {
3     if ( a1 % 2 == 1 )
4         return puts("just finished");
5     else
6         return get_flag();
7 }
```


The screenshot displays the IDA Pro interface with several windows open:

- Assembly View:** Shows assembly instructions for the `loc_400975` function, including `mov [rbp+var_28], 0`, `mov byte ptr [rbp+5], 69h`, and `switch (rand() % 200)`.
- Pseudocode-A:** Shows the corresponding C-like pseudocode, including `for (n4 = 0; n4 <= 4; ++n4)` and `switch (rand() % 200)`.
- Stack View:** Shows the stack frame for `get_flag:loc_400975`, with addresses ranging from `00007FFFFFDDA70` to `00007FFFFFDDA80`.
- Registers:** Shows the state of registers, including `RIP` at `00007FFFFFDDA70`.
- Threads:** Shows a single thread named `调试1 luck_gu` in a `Ready` state.

3.flag出来啦

```

please input a lucky number
2
Looking for GNU DWARF file at "/usr/lib/debug/.build-id/f6b7066f73e3b41dca3ab5b6da405f8edf2ec5.debug"... no.
OK, it's flag:
GXY{do_not_hate_me}emmm,you can't find flag 23333
emmm,you can't find flag 23333
OK,see you again
2025-11-09 14:05:59 [6] Closing connection from ::ffff:172.24.48.1...

```