

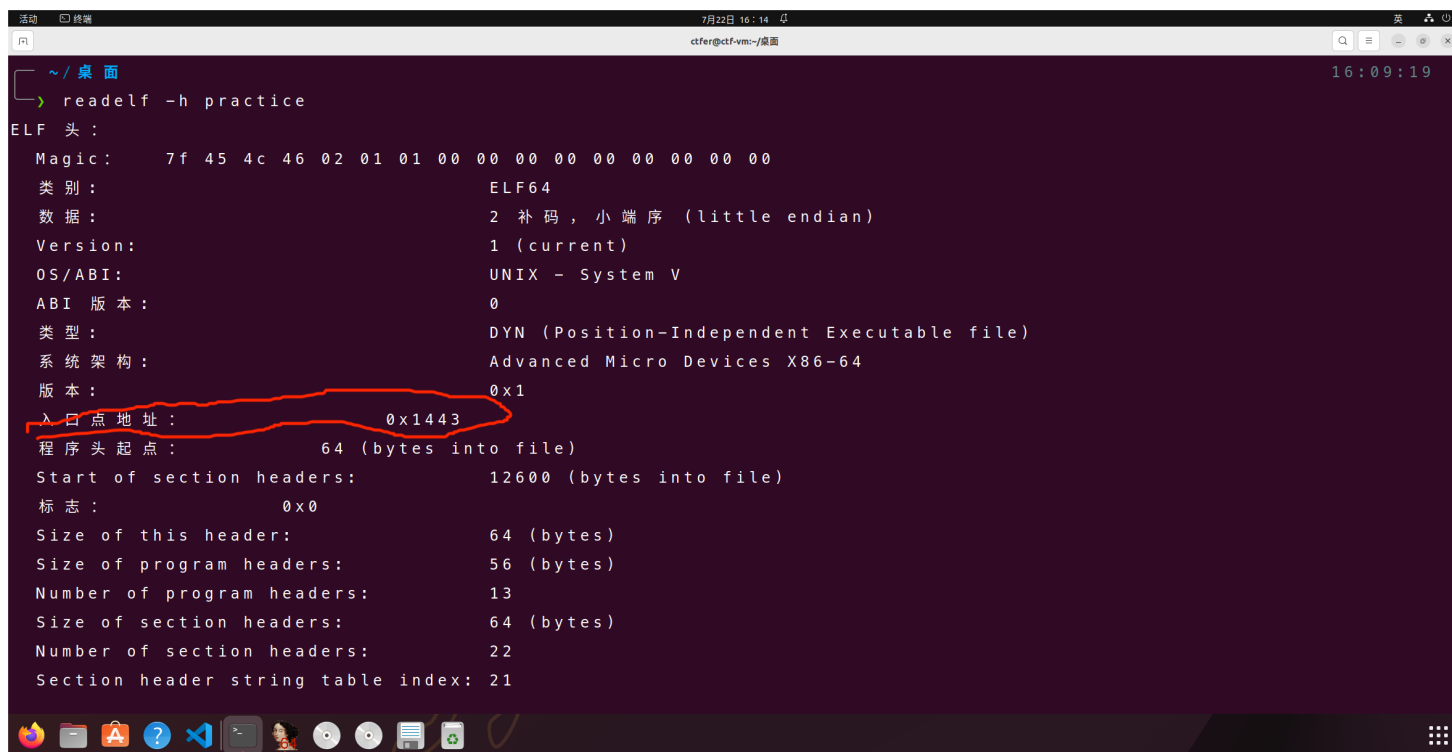
Reverse Lab1:Baby Reverse

3220102732-周伟战

Task1

1.1 Part 1

先利用zsh中的readelf -h practice命令行得到入口地址为0x1443
可以看到这个运行的是start()函数



```
ctfer@ctf-vm:~/桌面
> readelf -h practice
ELF 头:
Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
类别:                               ELF64
数据:                               2 补码, 小端序 (little endian)
Version:                             1 (current)
OS/ABI:                               UNIX - System V
ABI 版本:                             0
类型:                               DYN (Position-Independent Executable file)
系统架构:                           Advanced Micro Devices X86-64
版本:                               0x1
入口点地址:                           0x1443
程序头起点:                           64 (bytes into file)
Start of section headers:              12600 (bytes into file)
标志:                                0x0
Size of this header:                   64 (bytes)
Size of program headers:               56 (bytes)
Number of program headers:              13
Size of section headers:               64 (bytes)
Number of section headers:              22
Section header string table index:     21
```

1.1.1在题目中有一个函数是加密相关的函数，请找出这个函数的地址（Hex 格式作答，5 points）

```
BYTE *__fastcall sub_10F0(_BYTE *a1, int a2, _BYTE *a3)
{
    _BYTE *result; // rax
    unsigned __int8 v4; // [rsp+21h] [rbp-7h]
    unsigned __int8 v5; // [rsp+22h] [rbp-6h]
    char v6; // [rsp+23h] [rbp-5h]
    int i; // [rsp+24h] [rbp-4h]
    int j; // [rsp+24h] [rbp-4h]

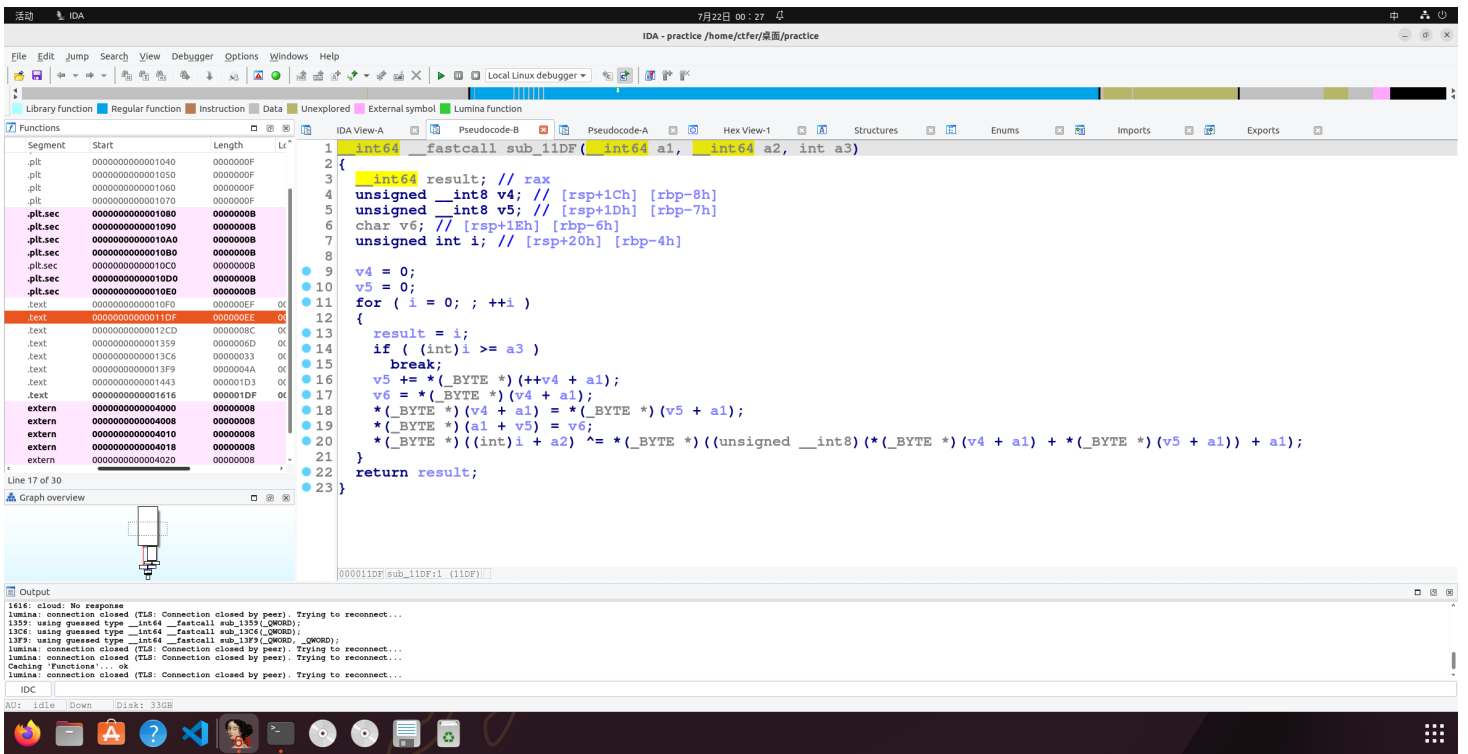
    for ( i = 0; i <= 255; ++i )
    {
        result = &a3[i];
        *result = i;
    }
    v4 = 0;
    v5 = 0;
    for ( j = 0; j <= 255; ++j )
    {
        v5 += a3[j] + a1[v4];
        v6 = a3[j];
        a3[j] = a3[v5];
        a3[v5] = v6;
        result = (_BYTE *) (unsigned int) ((v4 + 1) % a2);
        v4 = (v4 + 1) % a2;
    }
    return result;
}
```

这里将result,a1,a3的类型都变成了_BYTE*类型(a3是一个指针数组)，更改这些变量的类型之后发现已经有些C语言雏形了。

这个加密算法其实是RC4加密算法，下面这个P2更能验证这个加密算法是RC4

Line20的^ XOR运算就能大概知道这是加密算法

(Sub_11DF)

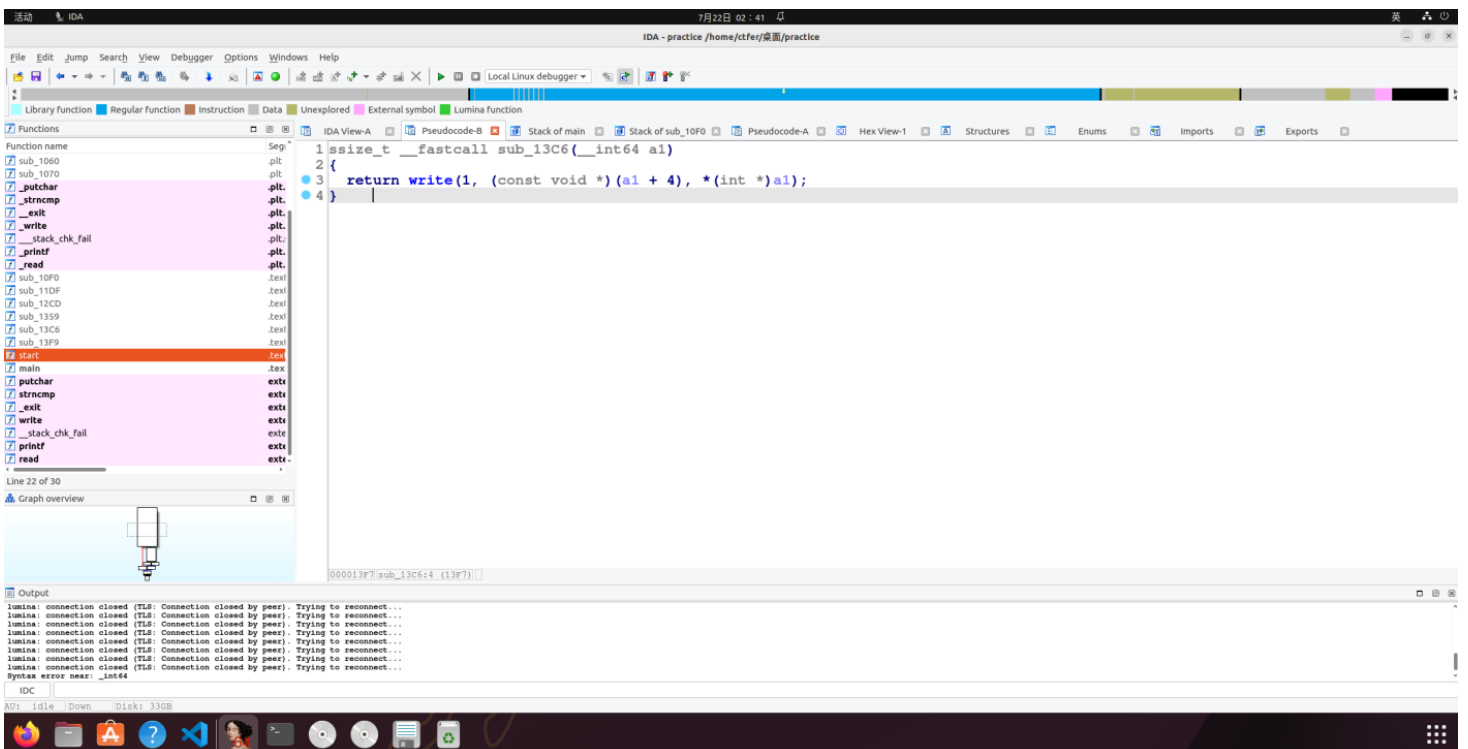


从左边的导航栏可以看见，这个加密函数的地址是000000000000011DF

1.1.2 当你找到了这个加密函数，请找出程序在加密过程中所使用到的密钥（5 points）

观察start()函数，可以知道密钥是'uwin@aaa'

1.1.3 在这个题目中，程序简单封装了短字符串类型，请在 IDA 中恢复它的结构体（截图或用 C 语言表示该结构，15 points）



将sub_13C6函数输入的类型改成_int64，发现write函数中偏移+4的位置，推断得出，封装的字符串类型string中前4个字节代表长度，后面3*8=24字节是拿来放string的内容；
如int v6, __int64 v7[3]就是这样4+3 * 8字节

1.1.4给出你解答的 flag 内容及 Writeup （15 points）

RC4加密算法和解密算法的密钥是同一个

Step1我们获得了密钥是'aaa@niwu'

Step2 由1.1.3中可以得知，我们把string的结构体分析出来了，是4字节带上3*8字节，需要解密的密文是

```
sub_12C0(\__int64 v5, v4, (\__int64 v1, v0),  
v16 = 18;  
v17 = 0xDBF40AEF840761FBLL;  
v18 = 0x9FD1DAB8555975CBLL;  
v19 = 0x861ALL;  
if ( sub_13F9((const char *)&v0, (const char *)&v16) )
```

v17 v18 v19之间顺序相连，但内部是倒序相加

得到需要利用RC4的密文是FB610784EF0AF4DBCB755955B8DAD19F1A86

剩下我们利用cyberchef

The screenshot shows the CyberChef web application. On the left, the 'Recipe' panel is active, showing a selected 'RC4' recipe with a passphrase of 'uwin@aaa'. The 'Input' panel on the right contains the hex string 'FB610784EF0AF4DBCB755955B8DAD19F1A86'. The 'Output' panel at the bottom displays the decrypted result: 'AAA{y0u_c4tch_Me!}'.

得到的flag是AAA{y0u_c4tch_Me!}

1.2 Part 2

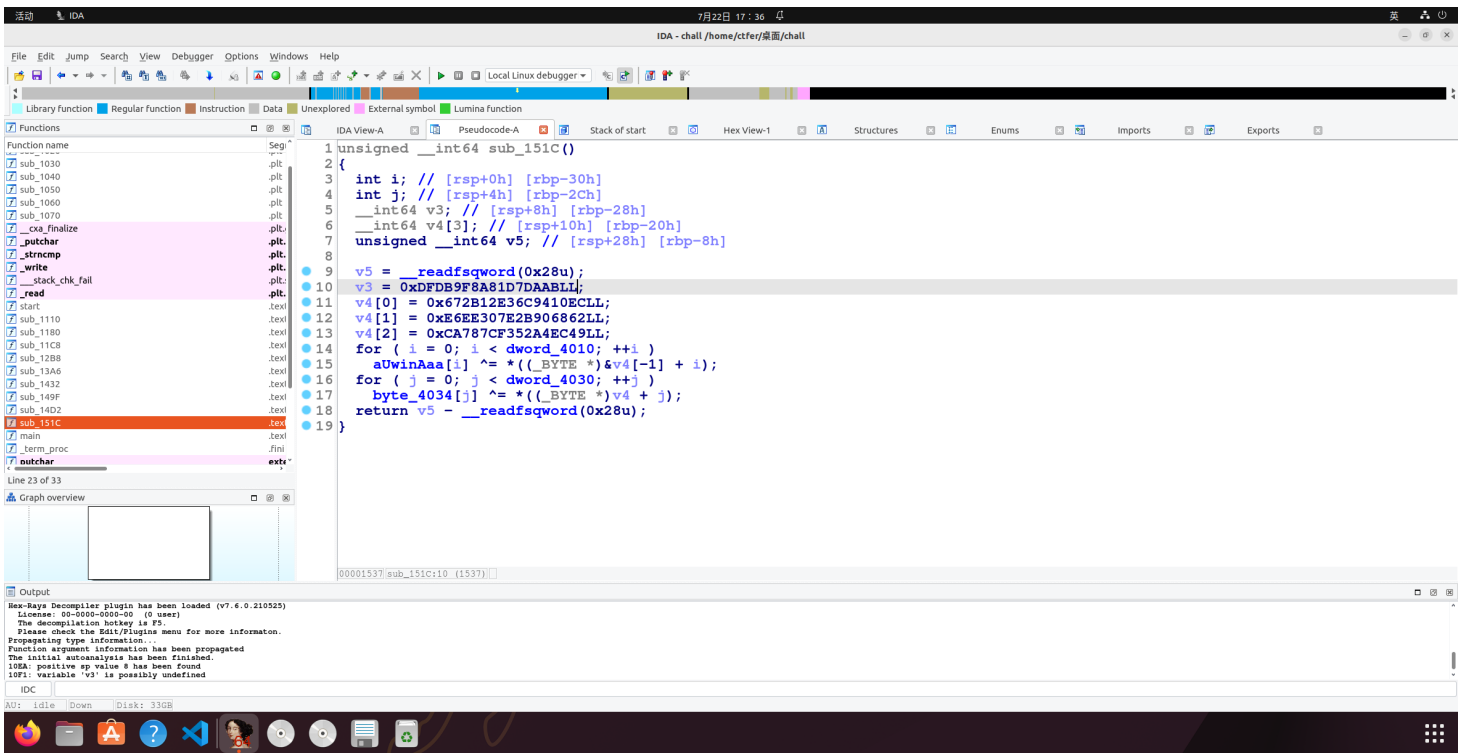
```

~/桌面
~/桌面
17:21:
17:21:41
> readelf -h chall
LF 头:
Magic:      7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
类别:      ELF64
数据:      2 补码, 小端序 (little endian)
Version:    1 (current)
OS/ABI:     UNIX - System V
ABI 版本:   0
类型:      DYN (Position-Independent Executable file)
系统架构:   Advanced Micro Devices X86-64
版本:      0x1
入口点地址: 0x10e0
程序头起点: 64 (bytes into file)
Start of section headers: 12680 (bytes into file)
标志:      0x0
Size of this header: 64 (bytes)
Size of program headers: 56 (bytes)
Number of program headers: 13
Size of section headers: 64 (bytes)
Number of section headers: 29
Section header string table index: 28

```

由readelf -h chall可以得到函数入口的地址是0x10e0，但进入了start()函数发现，其实看的是main函数，在了解相关的加密算法后发现 aUwinAaa 这个数组里面存放的就是密钥

1.2.1 程序中加密函数用到的的密钥是什么，你是如何找到它的 (10 points)



sub_151C函数是一个相对独立的函数，可以单独分析，看sub_151C函数可以发现是用来形成密钥以及获得密文的

The image shows a disassembly window in IDA Pro. The assembly code is as follows:

```

.data:0000000000004005      db      0
.data:0000000000004006      db      0
.data:0000000000004007      db      0
.data:0000000000004008      dq      offset off_4008      ; DATA XREF: sub_1180+1B+r
.data:0000000000004008      ; .data:off_4008+o
.data:0000000000004010      dword_4010      dd      8      ; DATA XREF: sub_151C:loc_15A9+r
.data:0000000000004010      ; main+D9+r
.data:0000000000004014      aUwinAaa      db      'uwin@aaa',0      ; DATA XREF: sub_151C+61+o
                                      ; sub_151C+7F+o ...
.data:000000000000401D      db      0
.data:000000000000401E      db      0
.data:000000000000401F      db      0
.data:0000000000004020      db      0
.data:0000000000004021      db      0
.data:0000000000004022      db      0
.data:0000000000004023      db      0
.data:0000000000004024      db      0
.data:0000000000004025      db      0
.data:0000000000004026      db      0
.data:0000000000004027      db      0
.data:0000000000004028      db      0
.data:0000000000004029      db      0
.data:000000000000402A      db      0
.data:000000000000402B      db      0
.data:000000000000402C      db      0
.data:000000000000402D      db      0
.data:000000000000402E      db      0

```

aUwinAaa[i] 中的初值是 'uwin@aaa'

先看 `aUwinAaa[i]^=((_BYTE *)&v4[-1] + i);` 这个语句，其实是把v[3]中存放的字符串与 aUwinAaa 这个密钥数组和 ABDAD7818A9FDBDF 进行异或操作得到密钥是 0xDEADBEEFCAFEBAFE

```

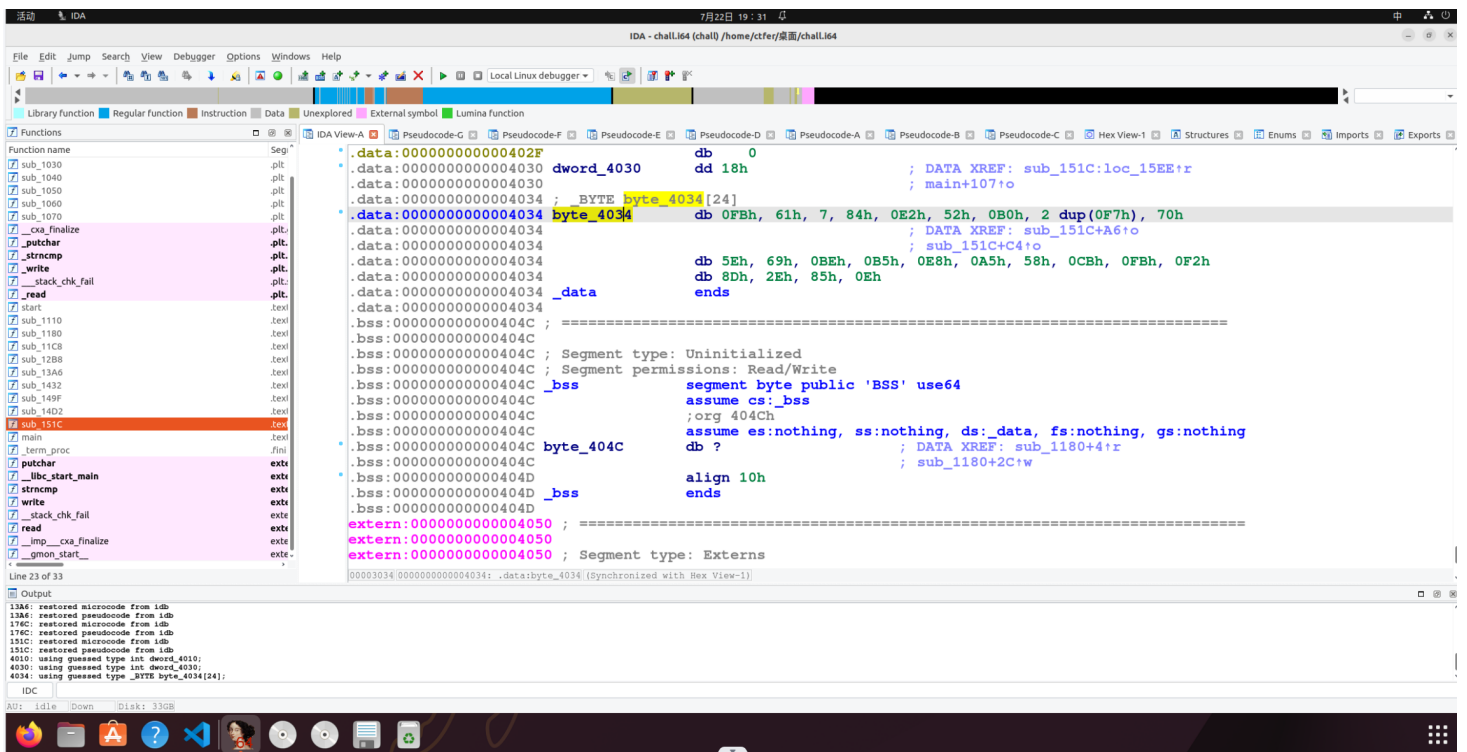
key=[0x75,0x77,0x69,0x6e,0x40,0x61,0x61,0x61]
key2=[0xab,0xda,0xd7,0x81,0x8a,0x9f,0xdb,0xdf]
key3=[0]*8
for x in range(8):
    key3[x]=key[x]^key2[x]
print(key3)

```

(代码在 [XOR.py](#)) 中

1.2.2给出你解答的 flag 内容及 Writeup (20 points)

再看密文



可以得到加高亮的那一行后面的内容就是储存在byte_4034中的初始值是：0xFB, 0x61, 0x7, 0x84, 0xE2, 0x52, 0xB0, 0xF7, 0xF7, 0x70, 0x5E, 0x69, 0xBE, 0xB5, 0xE8, 0xA5, 0x58, 0xCB, 0xFB, 0xF2, 0x8D, 0x2E, 0x85h, 0xE

再看v4[0],v4[1],v4[2]中分别

0xEC,0x10,0x94,0x6C,0xE3,0x12,0x2B,0x67,0x62,0x68,0x90,0x2B,0x7E,0x30,0xEE,0xE6,0X49,0xEC,0xA4,0x52,0xF3,0x7C,0x78,0xCA

利用

```
key=[0]*24
key1=[0xEC,0x10,0x94,0x6C,0xE3,0x12,0x2B,0x67,0x62,0x68,0x90,0x2B,0x7E,0x30,0xEE,0xE6,0x49,0xEC,0xA4,0x52,0xF3,0x7C,0x78,0xCA]
key2=[0xFB,0x61,0x7,0x84,0xE2,0x52,0xB0,0xF7,0xF7,0x70,0x5E,0x69,0xBE,0xB5,0xE8,0xA5,0x58,0xCB,0xF2,0x8D,0x2E,0x85h,0xE]
for x in range(24):
    key[x]=key1[x]^key2[x]
print(key)
```

获得异或处理过后的密文（代码在p.py中）

利用cyberchef中的RC4解密,最终得到flag:AAA{amAz1ng_y0u_F1nd_M3}

Recipe

XOR

Key

uWin@aaa

HEX

Scheme

Cascade

☒ Null preserving

To Hex

Delimiter

Space

Bytes per line

0

RC4

Passphrase

DEADBEEFCAF...

HEX

Input format

Hex

Output format

Latin1

From Hex

Delimiter

Auto

RC4

Passphrase

uwin@aaa

UTF8

Input format

Hex

Output format

Latin1

From Base64

Alphabet

STEP

BAKE!

Auto Bake

Input

17 71 93 e8 01 40 9b 90 95 18 ce 42 c0 85 06 43 11 27 5f a0 7e 52 fd c4

71

1

Raw Bytes

LF

Output

AAA{amAzing_y0u_F1nd_M3}

Over!

Task2

看似随机却并不随机，看似模糊却又清晰，请你耐心分析并提交：
emm没太看懂题目
over!