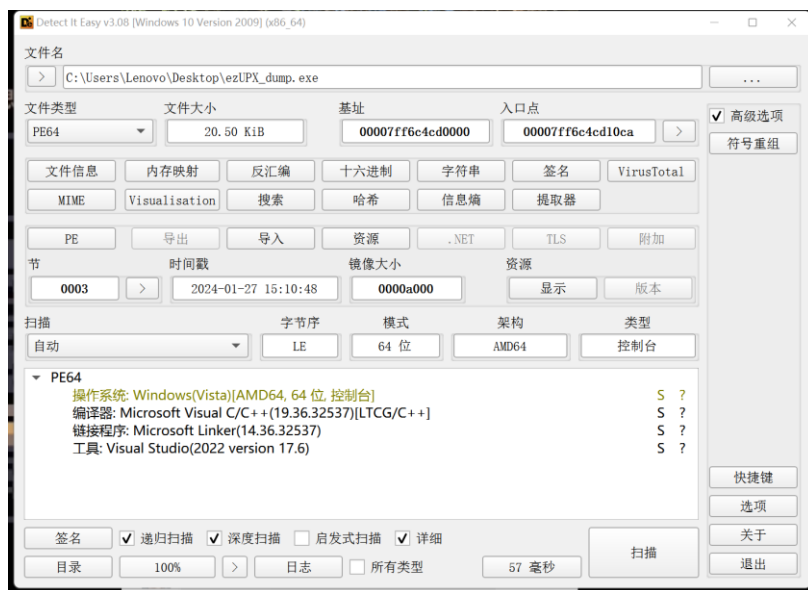


REVERSE

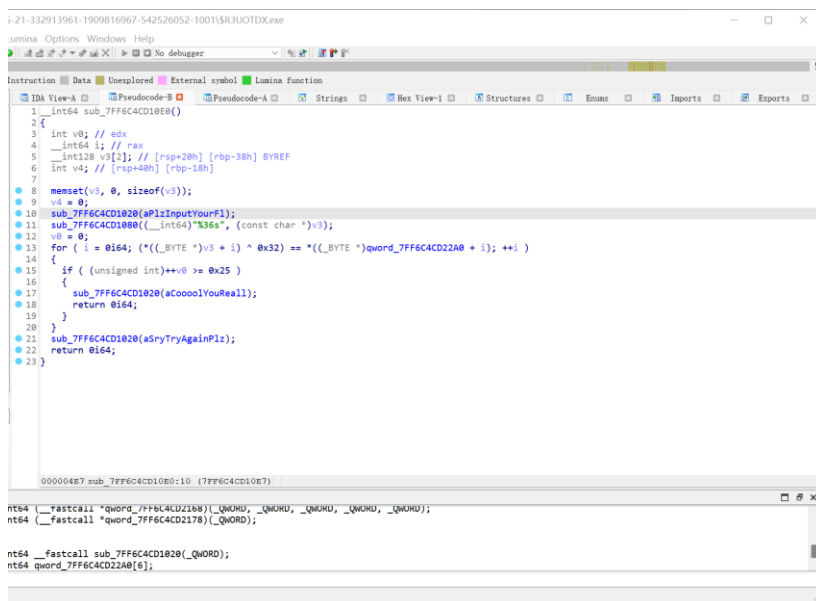
1. ezUPX

拖入 die 发现是 3.96 版本的 upx 壳，尝试用脱壳机一键脱，发现版本太低，于是尝试单步调试手脱，dump 下来成功得到脱壳



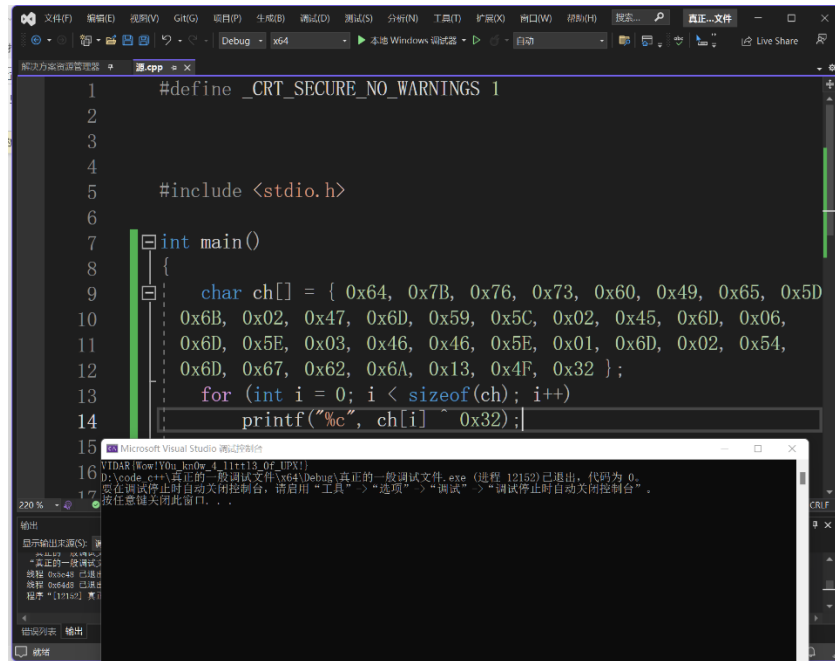
后的程序。

拖入 ida，通过搜索关键字字符串并且交叉引用找到主函数



分析可得，用户的输入与 0x32 异或与定义在内存里的密文对比后，若符合则输

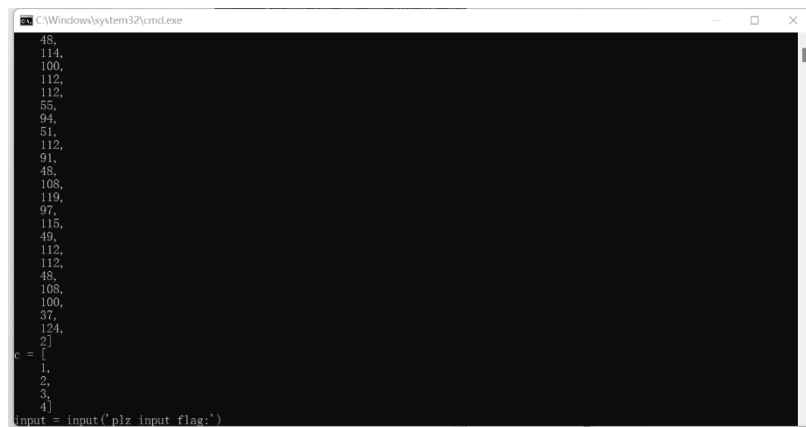
出正确，于是进入内存查看内存里的密文并 dump 下来，编写脚本，得到 flag。



```
1  #define _CRT_SECURE_NO_WARNINGS 1
2
3
4
5  #include <stdio.h>
6
7  int main()
8  {
9      char ch[] = { 0x64, 0x7B, 0x76, 0x73, 0x60, 0x49, 0x65, 0x5D
10     0x6B, 0x02, 0x47, 0x6D, 0x59, 0x5C, 0x02, 0x45, 0x6D, 0x06,
11     0x6D, 0x5E, 0x03, 0x46, 0x46, 0x5E, 0x01, 0x6D, 0x02, 0x54,
12     0x6D, 0x67, 0x62, 0x6A, 0x13, 0x4F, 0x32 };
13     for (int i = 0; i < sizeof(ch); i++)
14     {
15         printf("%c", ch[i] ^ 0x32);
16     }
17 }
```

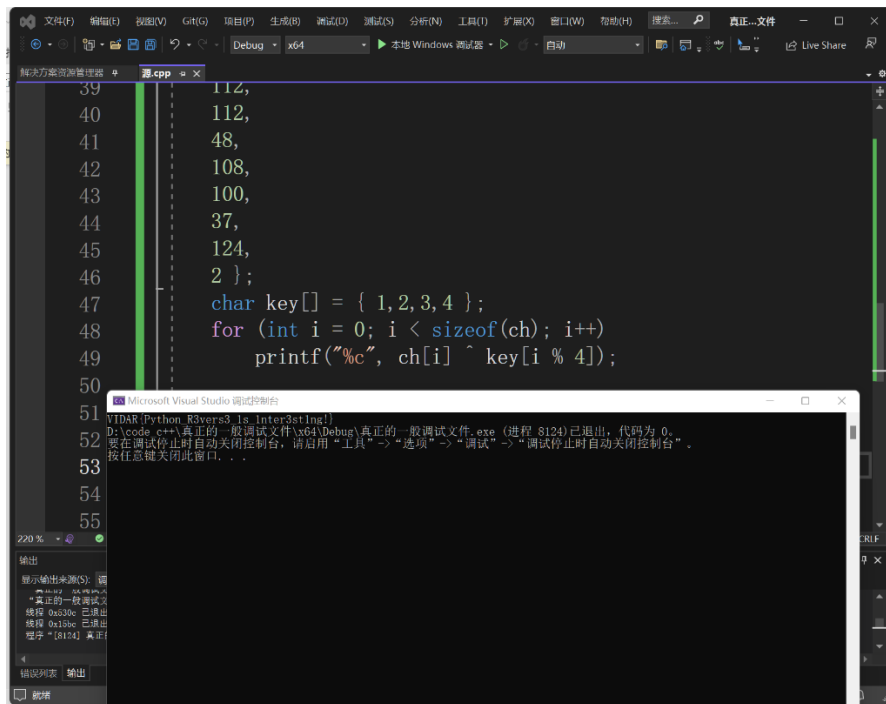
2. ezPYC

把 exe 文件用 pyinstxtractor 解包出 ezPYC.pyc 文件，用 pycdc.exe 对 ezPYC.pyc 反编译出代码



```
C:\Windows\system32\cmd.exe
48,
114,
100,
112,
112,
55,
94,
51,
112,
91,
48,
108,
119,
97,
115,
49,
112,
112,
48,
108,
100,
37,
124,
2
[
1,
2,
3,
4
]
input = input('plz input flag:')
```

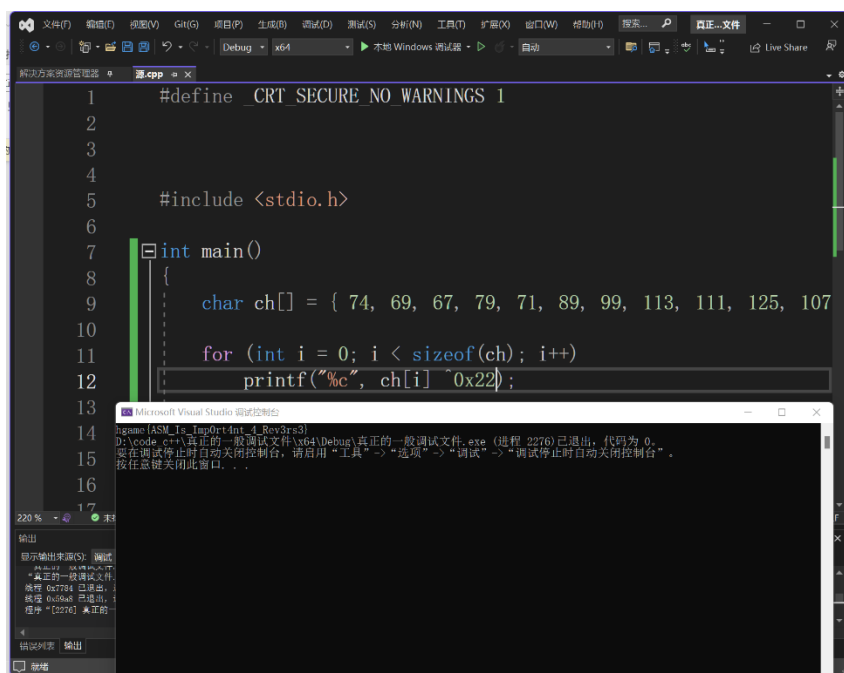
发现反编译的代码不完整，但大胆猜测是将数组里面的字符与 1, 2, 3, 4 异或，编写脚本得到 flag。



——后纠正发现是非预期，代码逻辑和猜想一致。

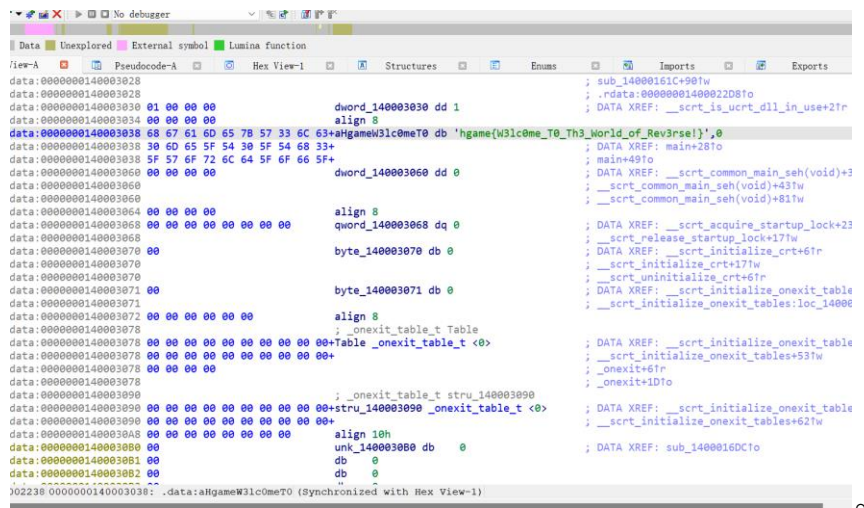
3. ezASM

简单的汇编语言，可知 c 是 flag 加密后的密文，里面的 check-flag 是简单的异或，逻辑就是用户输入的 flag 的每个字符都与 0x22 异或后等于 c，编写脚本得到 flag



4. ezIDA

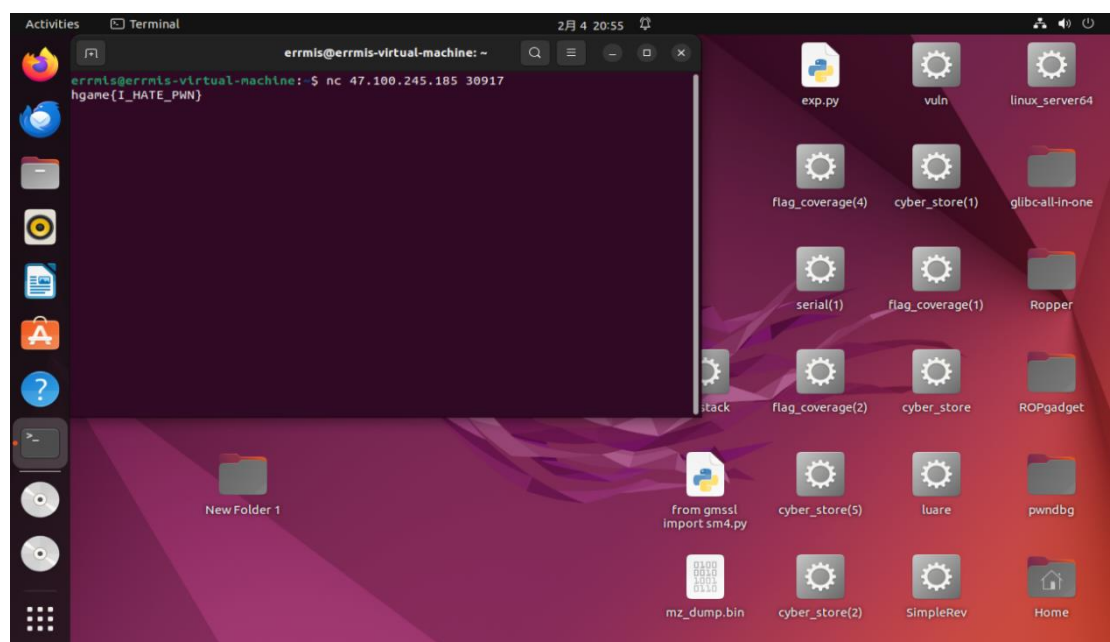
分析代码逻辑，发现与用户的输入对比的字符串就是 flag，点进内存就能找到 flag。



```
data:0000000140003028 ; sub_14000161C+907w
data:0000000140003028 ; .rdata:00000001400022D8to
data:0000000140003030 01 00 00 00 dword_140003030 dd 1
data:0000000140003034 00 00 00 00 align 8
data:0000000140003038 68 67 61 60 65 78 57 33 6C 63+aHgameW3lc0meT0 db 'hgame{W3lc0me_T0_Th3_W0rld_of_Rev3rse!}',0
data:0000000140003038 30 60 65 5F 54 30 5F 54 68 33+
data:0000000140003038 5F 57 6F 72 6C 64 5F 6F 66 5F+
data:0000000140003060 00 00 00 00 dword_140003060 dd 0
data:0000000140003060
data:0000000140003060
data:0000000140003064 00 00 00 00 align 8
data:0000000140003068 00 00 00 00 00 00 00 00 qword_140003068 dq 0
data:0000000140003070 00
data:0000000140003070 byte_140003070 db 0
data:0000000140003071 00
data:0000000140003072 00 00 00 00 00 00 00 00 align 8
data:0000000140003078 ; _onexit_table_t Table
data:0000000140003078 00 00 00 00 00 00 00 00 00+stru_140003090 _onexit_table_t <0>
data:0000000140003078 00 00 00 00 00 00 00 00
data:0000000140003078
data:0000000140003090 ; _onexit_table_t stru_140003090
data:0000000140003090 00 00 00 00 00 00 00 00 00+stru_140003090 _onexit_table_t <0>
data:0000000140003090 00 00 00 00 00 00 00 00 00+
data:00000001400030A0 align 10h
data:00000001400030B0 00 unk_1400030B0 db 0
data:00000001400030B1 00 db 0
data:00000001400030B2 00 db 0
J02238 0000000140003038: .data:aHgameW3lc0meT0 (Synchronized with Hex View-1)
```

PWNN

1. EZsignin



MISC

1. Signin

打开是一张图片，第一反应从手机孔往上看，发现还真是 flag。

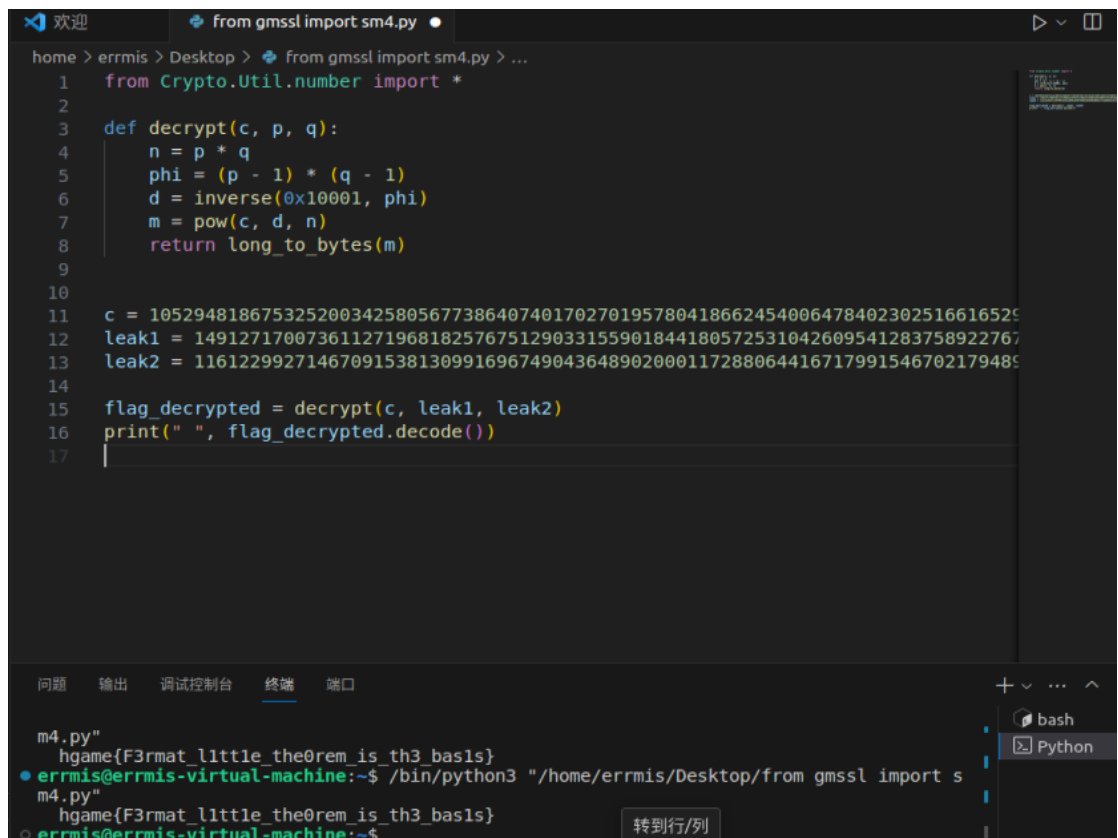
2. 签到

给长亭科技发 HGAME2024 就拿到 flag 了。

CRYPTO

1. ezRSA

一个简单的 RSA 加密，根据 leak1 和 leak2 可以用来生成密钥编写代码得到 flag。



```
from gmssl import sm4.py
1 from Crypto.Util.number import *
2
3 def decrypt(c, p, q):
4     n = p * q
5     phi = (p - 1) * (q - 1)
6     d = inverse(0x10001, phi)
7     m = pow(c, d, n)
8     return long_to_bytes(m)
9
10
11 c = 105294818675325200342580567738640740170270195780418662454006478402302516616525
12 leak1 = 14912717007361127196818257675129033155901844180572531042609541283758922767
13 leak2 = 11612299271467091538130991696749043648902000117288064416717991546702179485
14
15 flag_decrypted = decrypt(c, leak1, leak2)
16 print(" ", flag_decrypted.decode())
17
```

```
m4.py"
hgame{F3rmat_little_the0rem_is_th3_bas1s}
errmis@errmis-virtual-machine:~$ /bin/python3 "/home/errmis/Desktop/from gmssl import s
m4.py"
hgame{F3rmat_little_the0rem_is_th3_bas1s}
errmis@errmis-virtual-machine:~$
```