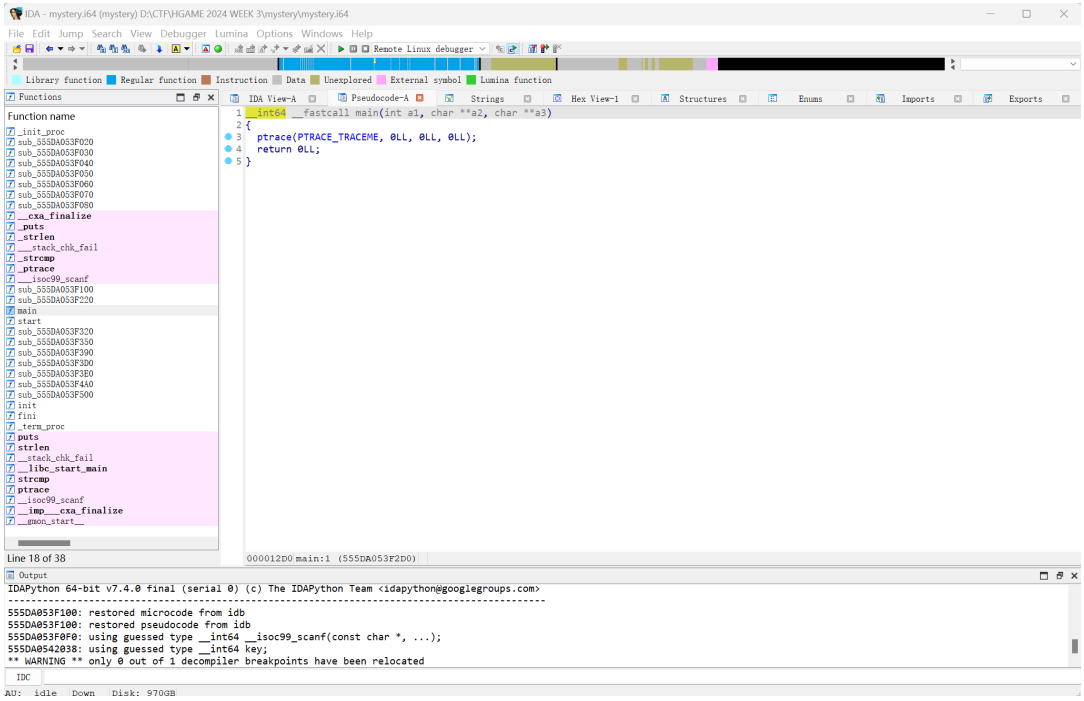
HGAME 2024 WEEK3 Reverse

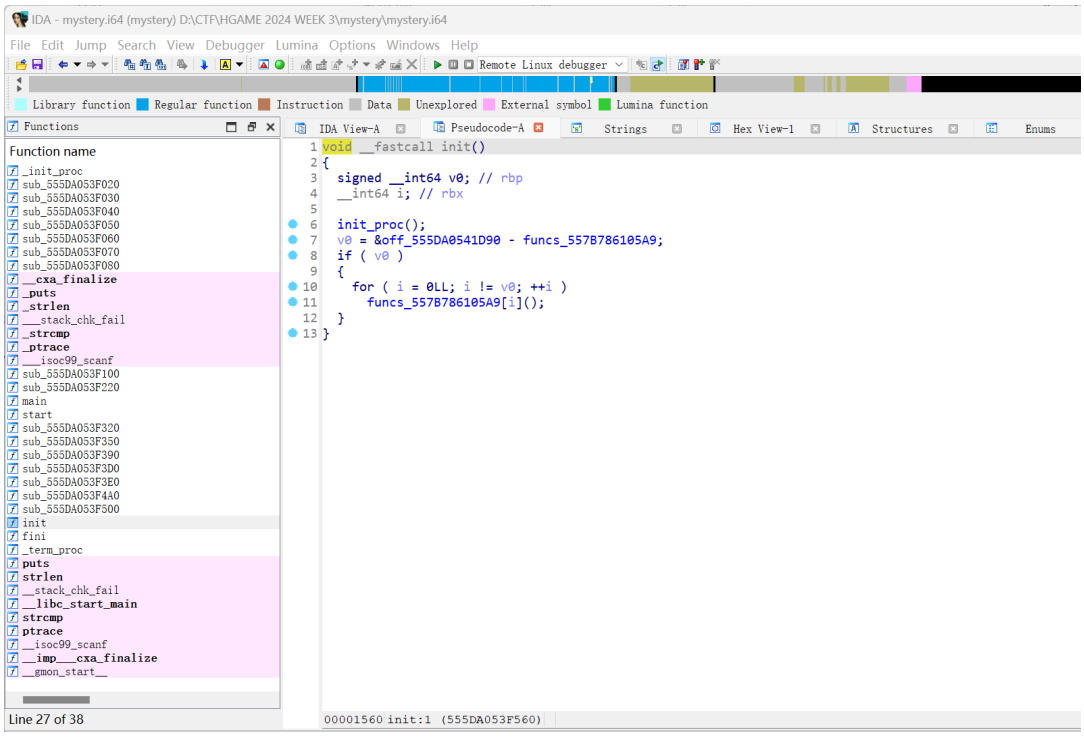
废话不多说，直接进入正题

1. mystery

考点：Init函数，rc4魔改

Ida打开，静态发现main()函数里什么也没有，猜测是start()里的Init()函数进行了修改





打开后发现确实进行了修改，来看第一个函数，发现是用rc4加密来加密一个key

```
__int64 sub_555DA053F220()
{
  unsigned __int64 v0; // rax

  key ^= 0x2F2F2F2F2F2F2F2FuLL;
  word_555DA0542040 ^= 0x2F2Fu;
  *aDjvdjv ^= 0x2F2F2F2Fu;
  *&aDjvdjv[4] ^= 0x2F2Fu;
  v0 = strlen(aDjvdjv);
  sub_555DA053F3E0(&s_box, aDjvdjv, v0);
  return sub_555DA053F4A0(&s_box, &key, strlen(&key));
}
```

然后点开第二个函数，发现是真正的加密流程，而这轮rc4所使用的key，就是上一个函数加密的结果

```c
int sub_555DA053F100()
{
  puts("please input your flag:\n");
  __isoc99_scanf("%s", s1);
  memset(&s_box, 0, 0x100uLL);
  rc4_init(&s_box, &key, strlen(&key));
  rc4_enc(&s_box, s1, strlen(s1));
  if ( !strcmp(s1, s2) )
    return puts("Congratulations!\n");
  else
    return puts("Wrong!please try again!");
}
```

需要注意的是，第二处rc4加密有魔改，是减法而非异或。

而众所周知，rc4是密钥流加密，反正最后只要获取到变量result的值就行了

```c
__int64 __fastcall sub_555DA053F500(__int64 s, _BYTE *a2, __int64 a3)
{
  _BYTE *v3; // r10
  unsigned int j; // r9d
  unsigned int i; // r8d
  char *s_ip; // rax
  char s_i; // dl
  char *v8; // rcx
  __int64 result; // rax

  if ( a3 )
  {
    v3 = &a2[a3];
    LOBYTE(j) = 0;
    LOBYTE(i) = 0;
    do
    {
      i = (i + 1);
      s_ip = (s + i);
      s_i = *s_ip;
      j = (*s_ip + j);
      v8 = (s + j);
      *s_ip = *v8;
      *v8 = s_i;
      result = *(s + (*s_ip + s_i));
      *a2++ -= result;
    }
    while ( v3 != a2 );
  }
  return result;
}
```

Execute script

Snippet list

Name

Default snippet

Please enter script body

```c
#include<idc.idc>
static main()
{
auto eax_value = GetRegValue("AL");
Message("0x%x,", eax_value);
}
```

打开虚拟机进行动调，下断点获取result的值，最后脚本就非常简单了

```
#include<iostream>
unsigned char text[]={
    0x50, 0x42, 0x38, 0x4D, 0x4C, 0x54, 0x90, 0x6F, 0xFE, 0x6F,
    0xBC, 0x69, 0xB9, 0x22, 0x7C, 0x16, 0x8F, 0x44, 0x38, 0x4A,
    0xEF, 0x37, 0x43, 0xC0, 0xA2, 0xB6, 0x34, 0x2C, 0x00}; // 明文密文统一用text
unsigned char key[] =
{
    0x18,0x25,0x29,0x20,0x19,0x27,0xb9,0xc9,0x34,0xc7,0x71,0xc9,0xac,0x17,0xb4,0x1e,0xe5,
    0xe9,0xfc,0x2a,0x4a,0x1,0xea,0x79,0xc7,0x82,0xfe,0x51,0xe7,0xb1,0xae,0x28,0x15
};
int main()
{
    for(int i=0;i<sizeof(text);i++)
    {
        text[i]+=key[i];
        printf("%c",text[i]);
    }

    ret
}
```

```
D:\CTF\HGAME 2024 WEEK 3\
hgame{I826-2e904t-4t98-9i82}
--------------------------------
Process exited after 0.04896 seconds with return value 0
请按任意键继续. . .
```

2. Encrypt

考点：AES CBC

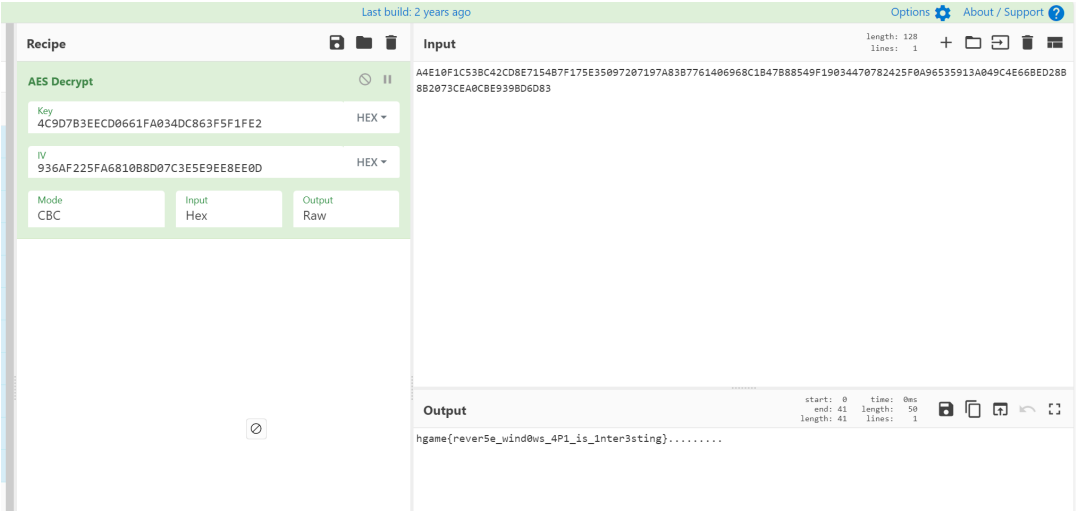
从pbSecret中提取key

```
memcpy(v11, &unk_7FF6914034A0, *v26);
v12 = 8i64;
*pbInput = _mm_xor_si128(_mm_load_si128(&xmmword_7FF691403500), _mm_loadu_si128(pbInput));
do
  *&pbInput[2 * v12++] ^= 0x55u;
while ( v12 < 15 );
if ( BCryptSetProperty(phAlgorithm, L"ChainingMode", pbInput, 0x20u, 0) >= 0
  && BCryptGenerateSymmetricKey(phAlgorithm, &phKey, v5, *pbOutput, &pbSecret, 0x10u, 0) >= 0
  && BCryptExportKey(phKey, 0i64, L"OpaqueKeyBlob", 0i64, 0, &cbOutput, 0) >= 0 )
{
  v13 = cbOutput;
  v14 = GetProcessHeap();
  v15 = HeapAlloc(v14, 0, v13);
  if ( v15 )
  {
    if ( BCryptExportKey(phKey, 0i64, L"OpaqueKeyBlob", v15, cbOutput, &cbOutput, 0) >= 0 )
    {
```

从v6中提取IV

```
v19 = GetProcessHeap();
v4 = HeapAlloc(v19, 0, v18);
if ( v4 )
{
  if ( BCryptEncrypt(phKey, v3, 0x32u, 0i64, v6, *v26, v4, v28, &pcbResult, 1u) >= 0
    && BCryptDestroyKey(phKey) >= 0 )
  {
    phKey = 0i64;
    v20 = GetProcessHeap();
    HeapFree(v20, 0, v3);
    v3 = 0i64;
    if ( !memcmp(v4, &unk_7FF691405050, v28) )
      puts("right flag!");
```

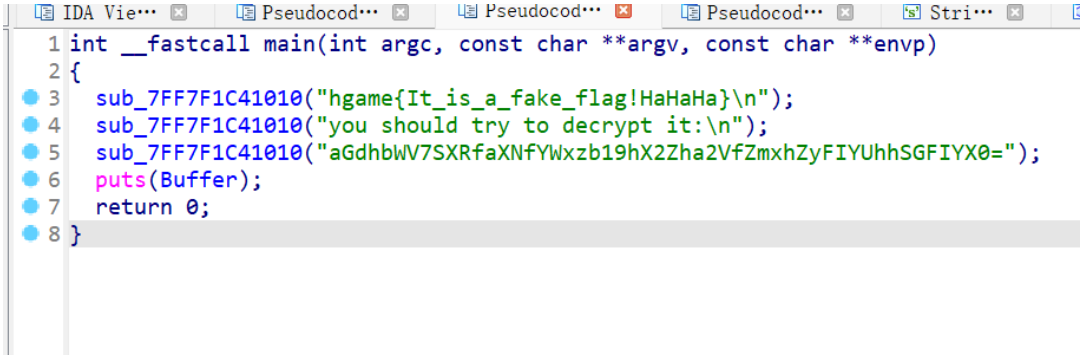
然后直接用解密脚本跑出来



3. findme

考点: dump，PE头，rc4魔改，动态调试

ida打开发现一个Fake_flag，然后base64码解出来也是个fake_flag



没办法，点开Buffer看看，惊喜地发现前两个字母是"MZ"，这也就意味着这里可能隐藏着一个PE文件

```
.data:00007FF7F1C44030 dword_7FF7F1C44030 dd 1                    ; DAT
.data:00007FF7F1C44034                 align 20h
.data:00007FF7F1C44040 ; char Buffer[2]
.data:00007FF7F1C44040 Buffer          db 'M',0                   ; DAT
.data:00007FF7F1C44042                 align 4
.data:00007FF7F1C44044 aZ              db 'Z',0
.data:00007FF7F1C44046                 align 8
.data:00007FF7F1C44048                 db  90h
.data:00007FF7F1C44049                 db    0
.data:00007FF7F1C4404A                 db    0
.data:00007FF7F1C4404B                 db    0
.data:00007FF7F1C4404C                 db    0
.data:00007FF7F1C4404D                 db    0
.data:00007FF7F1C4404E                 db    0
.data:00007FF7F1C4404F                 db    0
.data:00007FF7F1C44050                 db    3
.data:00007FF7F1C44051                 db    0
.data:00007FF7F1C44052                 db    0
.data:00007FF7F1C44053                 db    0
.data:00007FF7F1C44054                 db    0
.data:00007FF7F1C44055                 db    0
.data:00007FF7F1C44056                 db    0
.data:00007FF7F1C44057                 db    0
.data:00007FF7F1C44058                 db    0
.data:00007FF7F1C44059                 db    0
.data:00007FF7F1C4405A                 db    0
.data:00007FF7F1C4405B                 db    0
.data:00007FF7F1C4405C                 db    0
.data:00007FF7F1C4405D                 db    0
.data:00007FF7F1C4405E                 db    0
.data:00007FF7F1C4405F                 db    0
.data:00007FF7F1C44060                 db    4
.data:00007FF7F1C44061                 db    0
```

用010打开，找到对应部分，看到PE头的格式，更加验证了这个了这个猜想

```
2410  FF FF FF FF 01 00 00 00 01 00 00 00 02 00 00 00  ÿÿÿÿ............
2420  00 00 08 00 00 00 00 00 00 00 00 02 00 00 00 00  ................
2430  01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2440  4D 00 00 00 5A 00 00 00 90 00 00 00 00 00 00 00  M...Z...........
2450  03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2460  04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2470  FF 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00  ÿ...ÿ...........
2480  B8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ¸................
2490  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
24A0  40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  @...............
24B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
24C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
24D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
24E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
24F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2500  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2510  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2520  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2530  F8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ø...............
2540  0E 00 00 00 1F 00 00 00 BA 00 00 00 0E 00 00 00  ........º.......
2550  00 00 00 00 B4 00 00 00 09 00 00 00 CD 00 00 00  ....´.......Í...
2560  21 00 00 00 B8 00 00 00 01 00 00 00 4C 00 00 00  !...¸.......L...
2570  CD 00 00 00 21 00 00 00 54 00 00 00 68 00 00 00  Í...!...T...h...
2580  69 00 00 00 73 00 00 00 20 00 00 00 70 00 00 00  i...s... ...p...
2590  72 00 00 00 6F 00 00 00 67 00 00 00 72 00 00 00  r...o...g...r...
25A0  61 00 00 00 6D 00 00 00 20 00 00 00 63 00 00 00  a...m... ...c...
25B0  61 00 00 00 6E 00 00 00 6E 00 00 00 6F 00 00 00  a...n...n...o...
25C0  74 00 00 00 20 00 00 00 62 00 00 00 65 00 00 00  t... ...b...e...
25D0  20 00 00 00 72 00 00 00 75 00 00 00 6E 00 00 00   ...r...u...n...
25E0  20 00 00 00 69 00 00 00 6E 00 00 00 20 00 00 00   ...i...n... ...
25F0  44 00 00 00 4F 00 00 00 53 00 00 00 20 00 00 00  D...O...S... ...
2600  6D 00 00 00 6F 00 00 00 64 00 00 00 65 00 00 00  m...o...d...e...
2610  2E 00 00 00 0D 00 00 00 0D 00 00 00 0A 00 00 00  ................
2620  24 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  $...............
```

但需要注意的是，这里dump时要注意间隔4个字节再dump，因为插入了很多无意义的00 00 00

```
#include <stdio.h>
#include <stdlib.h>
#include<iostream>
#include<string>
#define N 100000
using namespace  std;
int main() {
    FILE *fp,*v5;
    int n=0,c;
    unsigned char str[N + 1],a,b;
    fp = fopen("findme.exe", "rb");
    v5 = fopen("real.exe", "wb");
    while ( feof(fp) ==0)
```

```
        {
            n++;
            a=fgetc(fp);
            //break;
            b=a;
              if(n>0x2440)
              {
                  if(n%4==1)
                  {
                      fputc(b, v5);
                  }
              }
        }

        //操作结束后关闭文件
        fclose(fp);
        fclose(v5);
        return 0;
    }
```

操作完成后得到真正的程序，但内部有许多花指令，建议写个idc或idapython脚本nop掉

```
.text:004F1190
.text:004F1190 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:004F1190 _main           proc near               ; CODE XREF: __scrt_common_main_seh(void)+F5↓p
.text:004F1190
.text:004F1190 Arglist         = byte ptr -14h
.text:004F1190 var_4           = dword ptr -4
.text:004F1190
.text:004F1190                 push    ebp
.text:004F1191                 mov     ebp, esp
.text:004F1193                 push    ecx
.text:004F1194                 push    ebx
.text:004F1195                 push    esi
.text:004F1196                 push    edi             ; ArgList
.text:004F1197                 nop
.text:004F1198                 nop
.text:004F1199                 nop
.text:004F119A                 nop
.text:004F119B                 nop
.text:004F119C                 push    offset Format   ; "plz input flag:\n"
.text:004F11A1                 call    sub_4F100C
.text:004F11A6                 mov     dword ptr [esp+14h+Arglist], offset input ; Arglist
.text:004F11AD                 push    offset a32s     ; "%32s"
.text:004F11B2                 call    sub_4F103A
.text:004F11B7                 pop     ecx
.text:004F11B8                 pop     ecx
.text:004F11B9                 mov     ecx, offset aDeadbeef ; "deadbeef"
.text:004F11BE                 lea     edx, [ecx+1]
.text:004F11C1
.text:004F11C1 loc_4F11C1:                             ; CODE XREF: _main+36↓j
.text:004F11C1                 mov     al, [ecx]
.text:004F11C3                 inc     ecx
.text:004F11C4                 test    al, al
.text:004F11C6                 jnz     short loc_4F11C1
.text:004F11C8                 sub     ecx, edx
.text:004F11CA                 mov     [ebp+var_4], ecx
.text:004F11CD                 nop
.text:004F11CE                 nop
.text:004F11CF                 nop
.text:004F11D0                 nop
.text:004F11D1                 nop
.text:004F11D2                 push    [ebp+var_4]
.text:004F11D5                 call    sub_4F1068
.text:004F11DA                 pop     ecx
.text:004F11DB                 mov     ecx, offset input
```

去除花指令后，不难发现这是个rc4加密

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // ecx
  char v5; // [esp+4h] [ebp-10h]
  char v6; // [esp+4h] [ebp-10h]

  sub_4F100C("plz input flag:\n", v5);
  sub_4F103A("%32s", input);
  rc4_init(strlen(aDeadbeef));
  rc4(strlen(input));
  v3 = 0;
  while ( input[v3] == byte_4F2148[v3] )
  {
    if ( ++v3 >= 32 )
    {
      sub_4F100C("Congratulations!", v6);
      return 0;
    }
  }
  sub_4F100C("Sry...try again", v6);
  return 0;
}
```

rc4密钥初始化阶段很正常，但点开rc4加密后发现这又是个魔改

关键点：通过动调可发现，result取值后面的 -(v4+s_box[v1]) 是一定会超出input原本的范围的，然后取到另一些内存的值，那么最好的做法仍然是直接动调获取到result，方法同mystery中的

```c
char __cdecl sub_4F110C(unsigned int a1)
{
  int v1; // ebx
  unsigned int v2; // edi
  int v3; // esi
  unsigned __int8 v4; // cl
  char result; // al

  v1 = 0;
  v2 = 0;
  if ( a1 )
  {
    v3 = 0;
    do
    {
      v1 = (v1 + 1) % 256;
      v4 = s_box[v1];
      v3 = (v4 + v3) % 256;
      s_box[v1] = s_box[v3];
      s_box[v3] = v4;
      result = input[-(v4 + s_box[v1])];
      input[v2++] += result;
    }
    while ( v2 < a1 );
  }
  return result;
}
```

取出result的值后，脚本同样很简单

```c
#include<iostream>
using namespace std;
unsigned char text[]={  0x7D, 0x2B, 0x43, 0xA9, 0xB9, 0x6B, 0x93, 0x2D, 0x9A, 0xD0,
   0x48, 0xC8, 0xEB, 0x51, 0x59, 0xE9, 0x74, 0x68, 0x8A, 0x45,
   0x6B, 0xBA, 0xA7, 0x16, 0xF1, 0x10, 0x74, 0xD5, 0x41, 0x3C,
   0x67, 0x7D};
unsigned char key[]={0x15,0xC4,0xE2,0x3C,0x54,0xF0,0x4D,0xC1,0x6A,0x59,0x15,0x56,0x78,0xF2,0x18,0x
int main()
{
    for(int i=0;i<sizeof(text);i++)
    {
        text[i]-=key[i];
        printf("%c",text[i]);
    }

    retu
}
```

```
D:\CTF\HGAME 2024 WEEK 3\
hgame{Fl0w3rs_Ar3_Very_fr4grant}
--------------------------------
Process exited after 0.05895 seconds with return value 0
请按任意键继续. . .
```

4. crackme

考点：c++，dump，反调试，流程混淆，异常处理，XTEA魔改

来到main()函数，猜测这些是TEA类加密的数据

```
24  char v25[16]; // [rsp+C0h] [rbp-88h] BYREF
25  char v26[24]; // [rsp+D0h] [rbp-78h] BYREF
26  char v27[32]; // [rsp+E8h] [rbp-60h] BYREF
27  char v28[24]; // [rsp+108h] [rbp-40h] BYREF
28
29  sub_7FF72A1B1C80(v27, argv, envp);
30  sub_7FF72A1B2490(std::cin, v27);
31  delta = 857870677;
32  key0 = 1234;
33  key1 = 2345;
34  key2 = 3456;
35  key3 = 4567;
36  sub_7FF72A1B1C20(v27, 0i64);
37  sub_7FF72A1B1C20(v27, 1i64);
38  sub_7FF72A1B1A80(v26, 24i64);
39  v3 = unknown_libname_16(&v9);
40  sub_7FF72A1B1BA0(v26, 8i64, v3);
41  for ( i = 0; i < 8; i += 2 )
42  {
43    v20 = sub_7FF72A1B1AD0(v26, i);
44    v19 = 4 * i;
45    v4 = sub_7FF72A1B1C00(v27);
46    *v20 = *(v4 + v19);
47    v22 = sub_7FF72A1B1AD0(v26, i + 1);
48    v21 = 4 * i;
49    v5 = sub_7FF72A1B1C00(v27);
50    *v22 = *(v5 + v21 + 4);
51  }
52  sub_7FF72A1B1A80(v28, 24i64);
53  enc[0] = 0x32FC31EA;
54  enc[1] = 0xF0566F42;
55  enc[2] = 0xF905B0B2;
56  enc[3] = 0x5F4551BE;
57  enc[4] = 0xFB3EFCBB;
58  enc[5] = 0x6B6ADB30;
59  enc[6] = 0x4839879;
60  enc[7] = 0x2F4378DF;
61  qmemcpy(v24, std::u16string_view::basic_string_view<char16_t,std::char_traits<char16_t>>(v25, enc,
62  v6 = unknown_libname_16(v10);
63  sub_7FF72A1B1B20(v28, v24, v6);
64  sub_7FF72A1B1AD0(v26, 0i64);
65  sub_7FF72A1B1AD0(v26, 1i64);
66  v11 = 0;
67  pExceptionObject = "exception";
68  CxxThrowException(&pExceptionObject, &_TI2PEAD);
```

然后发现程序戛然而止，而再程序的末尾有一个ThrowException的抛出异常的函数，而动调发现到这里之后程序就无法正常进行

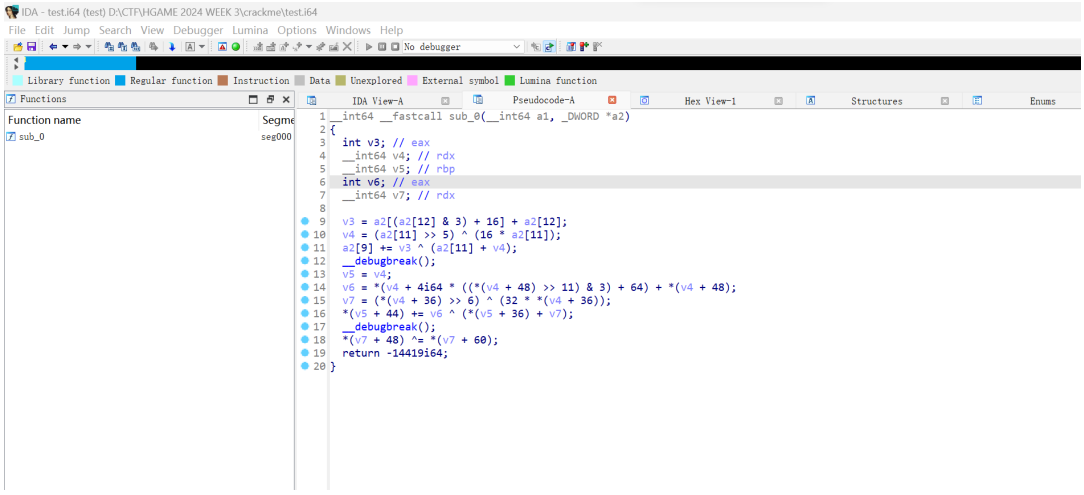
打开流程图后发现这是一种由Try和catch组成的结构，而catch部分是不能被正常反编译的，而这样的部分一共有三处。

```
mov     [rsp+148h+var_110], 0
jmp     short loc_7FF72A1B1915

loc_7FF72A1B1915:
cmp     [rsp+148h+var_110], 20h ; ' '
jge     short loc_7FF72A1B1993
;   } // starts at 7FF72A1B18A8

;   try {
lea     rax, aException ; "exception"
mov     [rsp+148h+pExceptionObject], rax
lea     rdx, __TI2PEAD  ; pThrowInfo
lea     rcx, [rsp+148h+pExceptionObject] ; pExceptionObject
call    _CxxThrowException

loc_7FF72A1B51E1:
;   catch(...) // owned by 7FF72A1B191C
mov     [rsp+148h+var_138], rdx
push    rbp
sub     rsp, 20h
mov     rbp, rdx
mov     eax, [rbp+30h]
and     eax, 3
mov     eax, [rbp+rax*4+40h]
mov     ecx, [rbp+30h]
add     ecx, eax
mov     eax, ecx
mov     ecx, [rbp+2Ch]
shr     ecx, 5
mov     edx, [rbp+2Ch]
shl     edx, 4
xor     edx, ecx
mov     ecx, edx
add     ecx, [rbp+2Ch]
xor     ecx, eax
mov     eax, ecx
mov     ecx, [rbp+24h]
add     ecx, eax
mov     eax, ecx
mov     [rbp+24h], eax
lea     rax, loc_7FF72A1B1942
add     rsp, 20h
pop     rbp
retn
```

这三处的内存地址是连续的，把他们dump下来再用ida打开反编译

```
.text:00007FF72A1B51C6                 push    rbp
.text:00007FF72A1B51C8                 sub     rsp, 20h
.text:00007FF72A1B51CC                 mov     rbp, rdx
.text:00007FF72A1B51CF                 lea     rcx, [rbp+108h]
.text:00007FF72A1B51D6                 call    sub_7FF72A1B1B00
.text:00007FF72A1B51DB                 add     rsp, 20h
.text:00007FF72A1B51DF                 pop     rbp
.text:00007FF72A1B51E0                 retn
.text:00007FF72A1B51E1 ; ---------------------------------------------------------------------------
.text:00007FF72A1B51E1
.text:00007FF72A1B51E1 loc_7FF72A1B51E1:                       ; DATA XREF: .rdata:00007FF72A1B6D34↓o
.text:00007FF72A1B51E1                                         ; .pdata:00007FF72A1BA6B4↓o ...
.text:00007FF72A1B51E1 ;   catch(...) // owned by 7FF72A1B191C
.text:00007FF72A1B51E1                 mov     [rsp+148h+var_138], rdx
.text:00007FF72A1B51E6                 push    rbp
.text:00007FF72A1B51E7                 sub     rsp, 20h
.text:00007FF72A1B51EB                 mov     rbp, rdx
.text:00007FF72A1B51EE                 mov     eax, [rbp+30h]
.text:00007FF72A1B51F1                 and     eax, 3
.text:00007FF72A1B51F4                 mov     eax, [rbp+rax*4+40h]
.text:00007FF72A1B51F8                 mov     ecx, [rbp+30h]
.text:00007FF72A1B51FB                 add     ecx, eax
.text:00007FF72A1B51FD                 mov     eax, ecx
.text:00007FF72A1B51FF                 mov     ecx, [rbp+2Ch]
.text:00007FF72A1B5202                 shr     ecx, 5
.text:00007FF72A1B5205                 mov     edx, [rbp+2Ch]
.text:00007FF72A1B5208                 shl     edx, 4
.text:00007FF72A1B520B                 xor     edx, ecx
.text:00007FF72A1B520D                 mov     ecx, edx
.text:00007FF72A1B520F                 add     ecx, [rbp+2Ch]
.text:00007FF72A1B5212                 xor     ecx, eax
.text:00007FF72A1B5214                 mov     eax, ecx
.text:00007FF72A1B5216                 mov     ecx, [rbp+24h]
.text:00007FF72A1B5219                 add     ecx, eax
.text:00007FF72A1B521B                 mov     eax, ecx
.text:00007FF72A1B521D                 mov     [rbp+24h], eax
.text:00007FF72A1B5220                 lea     rax, loc_7FF72A1B1942
.text:00007FF72A1B5227                 add     rsp, 20h
.text:00007FF72A1B522B                 pop     rbp
.text:00007FF72A1B522C                 retn
.text:00007FF72A1B522C ; ---------------------------------------------------------------------------
.text:00007FF72A1B522D                 align 2
.text:00007FF72A1B522E
.text:00007FF72A1B522E loc_7FF72A1B522E:                       ; DATA XREF: .rdata:00007FF72A1B6D3B↓o
.text:00007FF72A1B522E                                         ; .pdata:00007FF72A1BA6C0↓o ...
.text:00007FF72A1B522E ;   catch(...) // owned by 7FF72A1B1942
.text:00007FF72A1B522E                 mov     [rsp+arg_8], rdx
.text:00007FF72A1B5233                 push    rbp
.text:00007FF72A1B5234                 sub     rsp, 20h
.text:00007FF72A1B5238                 mov     rbp, rdx
.text:00007FF72A1B523B                 mov     eax, [rbp+30h]
.text:00007FF72A1B523E                 shr     eax, 0Bh
.text:00007FF72A1B5241                 and     eax, 3
.text:00007FF72A1B5244                 mov     eax, [rbp+rax*4+40h]
```

然后可以推断出这就是个XTEA加密，而且需要注意的是有魔改，不是sum + delta而是sum ^ delta



然后结合main()的数据得到解密脚本

```
#include <stdio.h>
#include <stdint.h>
```

```
 #include<iostream>
 using namespace std;

void decrypt ( unsigned int* v, unsigned int* k) {
    unsigned int v0=v[0], v1=v[1];
    unsigned int delta=857870677;
    unsigned int sum=0, i;
    for(i=0;i<32;i++)
    {
        sum^=delta;
    }
    for (i=0; i<32; i++) {
        sum ^= delta;
        v1 -= (((v0 >> 6) ^ ( v0 << 5 )) + v0) ^ (k[(sum >> 11) & 3] +
        v0 -= (((v1 >> 5) ^ (16 * v1)) + v1) ^ (k[sum & 3] + sum);
    }
    v[0]=v0; v[1]=v1;//解密后再重新赋值
}


int main()
{
    unsigned int v[8];
    v[0] = 0x32FC31EA;
  v[1] = 0xF0566F42;
  v[2] = 0xF905B0B2;
  v[3] = 0x5F4551BE;
  v[4] = 0xFB3EFCBB;
  v[5] = 0x6B6ADB30;
  v[6] = 0x4839879;
  v[7] = 0x2F4378DF;
    unsigned int k[4]={1234,2345,3456,4567};
    for(int i=0;i<8;i+=2)
    {
        decrypt(v+i, k);
    }
    unsigned char a;
    for(int i=0;i<8;i++)
    {
        for(int k=0;k<4;k++)
        {
            a=*((unsigned char *)(&v[i])+k);
            printf("%c",a);
        }
     }
```

```
    return 0;
}
```

flag就这么nice出来了

```c
int main()
{
    unsigned int v[8];
    v[0] = 0x32FC31EA;
  v[1] = 0xF0566F42;
  v[2] = 0xF905B0B2;
  v[3] = 0x5F4551BE;
  v[4] = 0xFB3EFCBB;
  v[5] = 0x6B6ADB30;
  v[6] = 0x4839879;
  v[7] = 0x2F4378DF;
    unsigned int k[4]={1234,2345,3456,4567};
    for(int i=0;i<8;i+=2)
    {
        decrypt(v+i, k);
    }
    unsigned char a;
    for(int i=0;i<8;i++)
    {
        for(int k=0;k<4;k++)
        {
            a=*((unsigned char *)(&v[i])+k);
            printf("%c",a);
        }
    }

    return 0;
}
```

```
D:\CTF\HGAME 2024 WEEK 3\
hgame{C_p1us_plus_exc3pti0n!!!!}
--------------------------------
Process exited after 0.06732 seconds with return value 0
请按任意键继续. . .
```