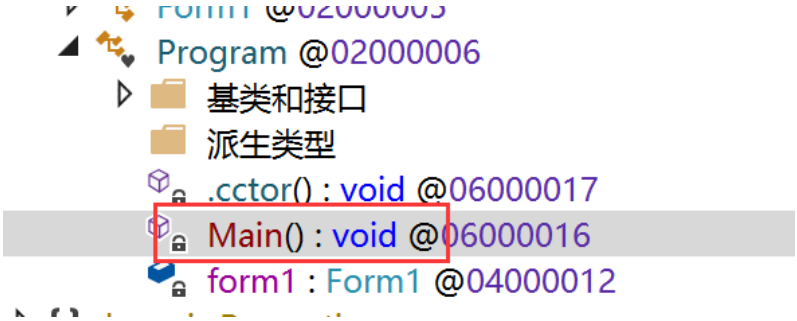
Hgame 2023 week3 Reverse

1. kmusic

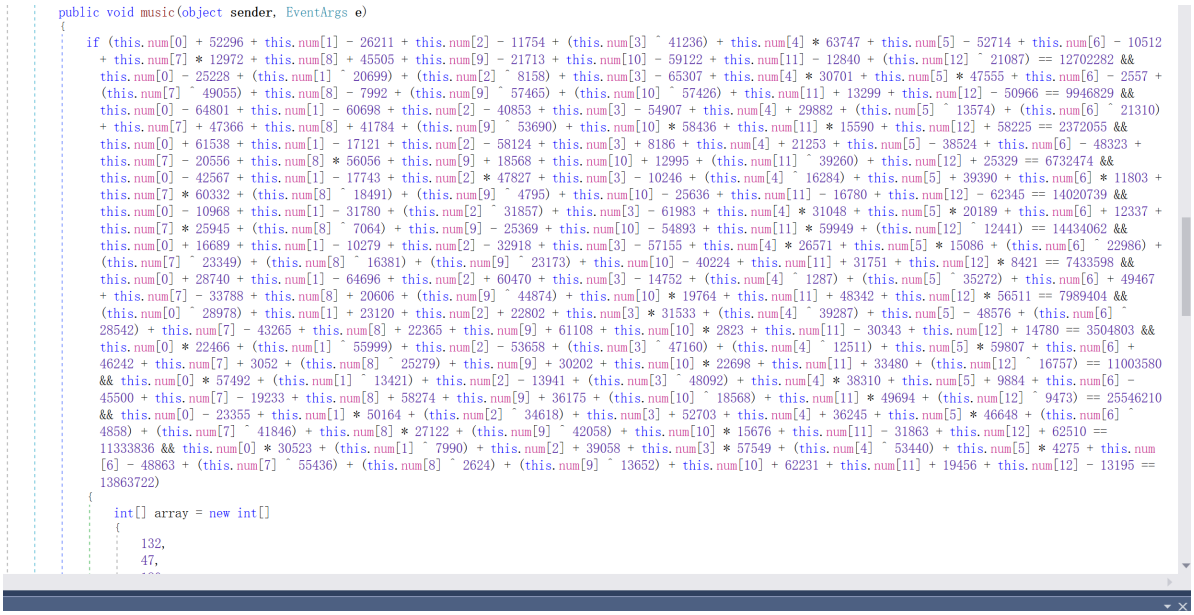
首先点开.exe文件运行(如果没有安装.net runtime,那么他会提醒你先下载,也可以在这里手动下载)。打开是一个如下界面:

点击会有对应的声音,到这里首先确定的是这是一个C#逆向，那么用dnSpy打开.dll文件。去到主类观看代码逻辑




```
[STAThread]
private static void Main()
{
    ApplicationConfiguration.Initialize();
    byte[] data = Resources.data;
    for (int i = 0; i < data.Length; i++)
    {
        byte[] array = data;
        int num = i;
        array[num] ^= 104;
    }
    Activator.CreateInstance(Assembly.Load(data).GetType("WinFormsLibrary1.Class1"), new objec
    {
        Program.form1
    });
    Application.Run(Program.form1);
}
```

在这里可以查看这两个函数的用法，可知道是将data里面的数据当作程序来运行，典型的SMC技术。笔者再这里的时候没有想到将data里面的数据作为文件dump出来，因为我对SMC的技术就局限于汇编代码的部分没想到还可以直接加密可执行程序。但是笔者再问了出题人之后一直将data里面的数据dump成exe文件导致我浪费了大量的时间。在这里插一句exe和dll文件头的前两个标志4D 5A也就是MZ。所以将文件dump出来保存为dll文件再用dnSpy打开。就可以看到本题关键的逻辑。



```
public void music(object sender, EventArgs e)
{
    if (this.num[0] + 52296 + this.num[1] - 26211 + this.num[2] - 11754 + (this.num[3] ^ 41236) + this.num[4] * 63747 + this.num[5] - 52714 + this.num[6] - 10512
      + this.num[7] * 12972 + this.num[8] + 45505 + this.num[9] - 21713 + this.num[10] - 59122 + this.num[11] - 12840 + (this.num[12] ^ 21087) == 12702282 &&
      this.num[0] - 25228 + (this.num[1] ^ 20699) + (this.num[2] ^ 8158) + this.num[3] - 65307 + this.num[4] * 30701 + this.num[5] * 47555 + this.num[6] - 2557 +
      (this.num[7] ^ 49055) + this.num[8] - 7992 + (this.num[9] ^ 57465) + (this.num[10] ^ 57426) + this.num[11] + 13299 + this.num[12] - 50966 == 9946829 &&
      this.num[0] - 64801 + this.num[1] - 60698 + this.num[2] - 40853 + this.num[3] - 54907 + this.num[4] + 29882 + (this.num[5] ^ 13574) + (this.num[6] ^ 21310)
      + this.num[7] + 47366 + this.num[8] + 41784 + (this.num[9] ^ 53690) + this.num[10] * 58436 + this.num[11] * 15590 + this.num[12] + 58225 == 2372055 &&
      this.num[0] + 61538 + this.num[1] - 17121 + this.num[2] - 58124 + this.num[3] + 8186 + this.num[4] + 21253 + this.num[5] - 38524 + this.num[6] - 48323 +
      this.num[7] - 20556 + this.num[8] * 56056 + this.num[9] + 18568 + this.num[10] + 12995 + (this.num[11] ^ 39260) + this.num[12] + 25329 == 6732474 &&
      this.num[0] - 42567 + this.num[1] - 17743 + this.num[2] * 47827 + this.num[3] - 10246 + (this.num[4] ^ 16284) + this.num[5] + 39390 + this.num[6] * 11803 +
      this.num[7] * 60332 + (this.num[8] ^ 18491) + (this.num[9] ^ 4795) + this.num[10] - 25636 + this.num[11] - 16780 + this.num[12] - 62345 == 14020739 &&
      this.num[0] - 10968 + this.num[1] - 31780 + (this.num[2] ^ 31857) + this.num[3] - 61983 + this.num[4] * 31048 + this.num[5] * 20189 + this.num[6] + 12337 +
      this.num[7] * 25945 + (this.num[8] ^ 7064) + this.num[9] - 25369 + this.num[10] - 54893 + this.num[11] * 59949 + (this.num[12] ^ 12441) == 14434062 &&
      this.num[0] + 16689 + this.num[1] - 10279 + this.num[2] - 32918 + this.num[3] - 57155 + this.num[4] * 26571 + this.num[5] * 15086 + (this.num[6] ^ 22986) +
      (this.num[7] ^ 23349) + (this.num[8] ^ 16381) + (this.num[9] ^ 23173) + this.num[10] - 40224 + this.num[11] + 31751 + this.num[12] * 8421 == 7433598 &&
      this.num[0] + 28740 + this.num[1] - 64696 + this.num[2] + 60470 + this.num[3] - 14752 + (this.num[4] ^ 1287) + (this.num[5] ^ 35272) + this.num[6] + 49467
      + this.num[7] - 33788 + this.num[8] + 20606 + (this.num[9] ^ 44874) + this.num[10] * 19764 + this.num[11] + 48342 + this.num[12] * 56511 == 7989404 &&
      (this.num[0] ^ 28978) + this.num[1] + 23120 + this.num[2] + 22802 + this.num[3] * 31533 + (this.num[4] ^ 39287) + this.num[5] - 48576 + (this.num[6] ^
      28542) + this.num[7] - 43265 + this.num[8] + 22365 + this.num[9] + 61108 + this.num[10] * 2823 + this.num[11] - 30343 + this.num[12] + 14780 == 3504803 &&
      this.num[0] * 22466 + (this.num[1] ^ 55999) + this.num[2] - 53658 + (this.num[3] ^ 47160) + (this.num[4] ^ 12511) + this.num[5] * 59807 + this.num[6] +
      46242 + this.num[7] + 3052 + (this.num[8] ^ 25279) + this.num[9] + 30202 + this.num[10] * 22698 + this.num[11] + 33480 + (this.num[12] ^ 16757) == 11003580
      && this.num[0] * 57492 + (this.num[1] ^ 13421) + this.num[2] - 13941 + (this.num[3] ^ 48092) + this.num[4] * 38310 + this.num[5] + 9884 + this.num[6] -
      45500 + this.num[7] - 19233 + this.num[8] + 58274 + this.num[9] + 36175 + (this.num[10] ^ 18568) + this.num[11] * 49694 + (this.num[12] ^ 9473) == 25546210
      && this.num[0] - 23355 + this.num[1] * 50164 + (this.num[2] ^ 34618) + this.num[3] + 52703 + this.num[4] + 36245 + this.num[5] * 46648 + (this.num[6] ^
      4858) + (this.num[7] ^ 41846) + this.num[8] * 27122 + (this.num[9] ^ 42058) + this.num[10] * 15676 + this.num[11] - 31863 + this.num[12] + 62510 ==
      11333836 && this.num[0] * 30523 + (this.num[1] ^ 7990) + this.num[2] + 39058 + this.num[3] * 57549 + (this.num[4] ^ 53440) + this.num[5] * 4275 + this.num
      [6] - 48863 + (this.num[7] ^ 55436) + (this.num[8] ^ 2624) + (this.num[9] ^ 13652) + this.num[10] + 62231 + this.num[11] + 19456 + this.num[12] - 13195 ==
      13863722)
    {
        int[] array = new int[]
        {
            132,
            47,
```

这个逻辑很简单就是一堆的if判断然后与密文进行比较。那么就很自然的想到了z3.

```
from z3 import *

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

```
flag[0] + 16689 + flag[1] - 10279 + flag[2] - 32918 + flag[3] - 57155 +
flag[4] * 26571 + flag[5] * 15086 + (flag[6] ^ 22986) + (flag[7] ^ 23349) +
(flag[8] ^ 16381) + (flag[9] ^ 23173) + flag[10] - 40224 + flag[11] + 31751
+ flag[12] * 8421 == 7433598 ,
flag[0] + 28740 + flag[1] - 64696 + flag[2] + 60470 + flag[3] - 14752 +
(flag[4] ^ 1287) + (flag[5] ^ 35272) + flag[6] + 49467 + flag[7] - 33788 +
flag[8] + 20606 + (flag[9] ^ 44874) + flag[10] * 19764 + flag[11] + 48342 +
flag[12] * 56511 == 7989404 ,
(flag[0] ^ 28978) + flag[1] + 23120 + flag[2] + 22802 + flag[3] * 31533 +
(flag[4] ^ 39287) + flag[5] - 48576 + (flag[6] ^ 28542) + flag[7] - 43265 +
flag[8] + 22365 + flag[9] + 61108 + flag[10] * 2823 + flag[11] - 30343 +
flag[12] + 14780 == 3504803 ,
flag[0] * 22466 + (flag[1] ^ 55999) + flag[2] - 53658 + (flag[3] ^ 47160) +
(flag[4] ^ 12511) + flag[5] * 59807 + flag[6] + 46242 + flag[7] + 3052 +
(flag[8] ^ 25279) + flag[9] + 30202 + flag[10] * 22698 + flag[11] + 33480 +
(flag[12] ^ 16757) == 11003580 ,
flag[0] * 57492 + (flag[1] ^ 13421) + flag[2] - 13941 + (flag[3] ^ 48092) +
flag[4] * 38310 + flag[5] + 9884 + flag[6] - 45500 + flag[7] - 19233 +
flag[8] + 58274 + flag[9] + 36175 + (flag[10] ^ 18568) + flag[11] * 49694 +
(flag[12] ^ 9473) == 25546210 ,
flag[0] - 23355 + flag[1] * 50164 + (flag[2] ^ 34618) + flag[3] + 52703 +
flag[4] + 36245 + flag[5] * 46648 + (flag[6] ^ 4858) + (flag[7] ^ 41846) +
flag[8] * 27122 + (flag[9] ^ 42058) + flag[10] * 15676 + flag[11] - 31863 +
flag[12] + 62510 == 11333836 ,
flag[0] * 30523 + (flag[1] ^ 7990) + flag[2] + 39058 + flag[3] * 57549 +
(flag[4] ^ 53440) + flag[5] * 4275 + flag[6] - 48863 + (flag[7] ^ 55436) +
(flag[8] ^ 2624) + (flag[9] ^ 13652) + flag[10] + 62231 + flag[11] + 19456 +
flag[12] - 13195 == 13863722 ,
flag[0] == 236 , flag[1] == 72 , flag[2] == 213 , flag[3] == 106 , flag[4]
== 189 , flag[5] == 86 , flag[6] != 190 , flag[7] == 53
)
if s.check() == sat:
    model = s.model()
    for i in flag:
        print(model[i],end=',')
key = [236,72,213,106,189,86,190,53,120,71,15,93,133]
key = [236,72,213,106,189,86,62,53,120,199,15,93,133]
enc =
[132,47,180,7,216,45,68,6,39,246,124,2,243,137,58,172,53,200,99,91,83,13,171
,80,108,235,179,58,176,28,216,36,11,80,39,162,97,58,236,130,123,176,24,212,5
6,89,72]

for i in range(len(enc)):
    print(chr(enc[i] ^ key[i % len(key)]),end='')
```

在这里z3会有多解，那么我们就限定和game{}这几个已知的未知数。但是笔者在这里似乎还是没有解出正确的答案，结果里面还有不可见字符，所以我就手动算出了flag[6]算错了，于是又加了一个限定条件才跑出结果来。

2. patchme

这个题目再描述中写的很清楚了，就是修改漏洞。拖进IDA里面查看

```
__int64 __fastcall main(int a1, char **a2, char *
{
  char format[24]; // [rsp+10h] [rbp-20h] BYREF
  unsigned __int64 v5; // [rsp+28h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  dword_4028 = a1;
  qword_4020 = (__int64)a2;
  gets(format, a2, a3);
  printf(format);
  return 0LL;
}
```

再这里面就可以很清晰的看到两个漏洞，gets会造成栈溢出将其换成scanf，priintf存在字符串格式化漏洞，在这里传入一个“%s”参数就行了。这里就先要了解一下函数的传参方式了，32位参数使用栈传参，64位使用寄存器陈传参，从左到右依次是rdi,rsi,rdx,rcx,r8,r9,多的用栈传递参数。那么开始修改了，首先是printf函数

```
lea     rax, [rbp+format]
mov     rdi, rax
mov     eax, 0
call    _printf
```

```
lea     rsi, [rbp+format]

lea     rdi, format



nop
call    _printf
```

在这里笔者卡住的地方是穿如字符串是mov rdi,offset aS;我不知道为什么会出错，但以后传地址参数还是用lea指令吧。对了这里有一个"%s"的参数需要自己手动添加一般是在eh_frame段进行添加的，因为这里的数据改变不会影响整个程序的运行。

```
.eh_frame:00000000000020C0 25 73 00                    format db '%s',0
.eh_frame:00000000000020C3 00                          db    0
```

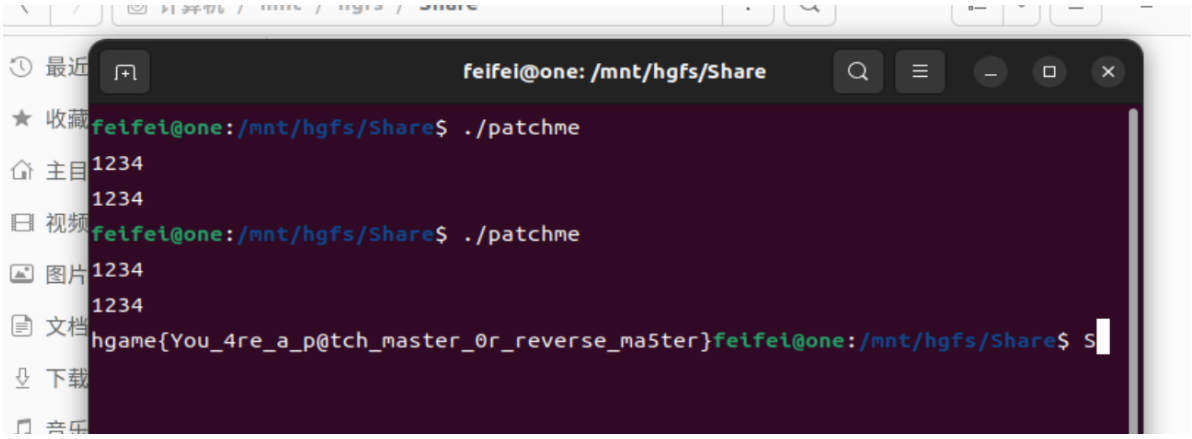
对了这里要严格注意字节码的位数，不能将call _printf函数破坏，之前他传format参数是借助rax寄存器了，多次一举了，我们只用一条指令就可以完成，剩下的mov rax,0就可以用来传"%s"参数了，可以看到这里还多了一个字节码的空间没有用到被nop掉了。接下来就是gets函数了，同样的道理定义一个“%24s”的字符串再传参、调用scanf函数就行了。（之后看了官方wp说这里是改为"%23s",要留一个给\x0作为字符串的结尾，但是我的这个附件是原来的附件所以难度降低了一些，后面给了hint之后就适当的讲题目难度加大了一点）。

```
lea     rsi, [rbp+format]

lea     rdi, a24s



nop
call    ___isoc99_scanf
```

修改后如下

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  char format[24]; // [rsp+10h] [rbp-20h] BYREF
  unsigned __int64 v5; // [rsp+28h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  dword_4028 = a1;
  qword_4020 = (__int64)a2;
  __isoc99_scanf("%24s", format);
  printf("%s", format);
  return 0LL;
}
```

运行即可得到结果。



3. cpp

首先这是一个c++的逆向，c++的逆向有一个特点就是反编译之后的伪代码特别难看：

```
std::string::string(&input);
std::operator>><char>(std::cin, &input);
v7 = operator new(0x70ui64);
if ( v7 )
{
  memset(v7, 0, sizeof(encrypt2));
  std::string::string(&v9, &input);
  encrypt2::encrypt2(v7, "hgame{th1s_is_4_fake_fl4g_hahaha}", 0x12345678u, "hgame{this_is_another_fake_flag}", v3);
  v8 = v4;
}
else
{
  v8 = 0i64;
}
v6 = v8;
(*(*v8 + 16i64))(v8);
(**v6)(v6);
if ( (*(*v6 + 48i64))(v6) )
  std::operator<<<std::char_traits<char>>(std::cout, "yes!");
else
  std::operator<<<std::char_traits<char>>(std::cout, "try again ... ");
std::string::_Tidy_deallocate(&input);
return 0;
}
```

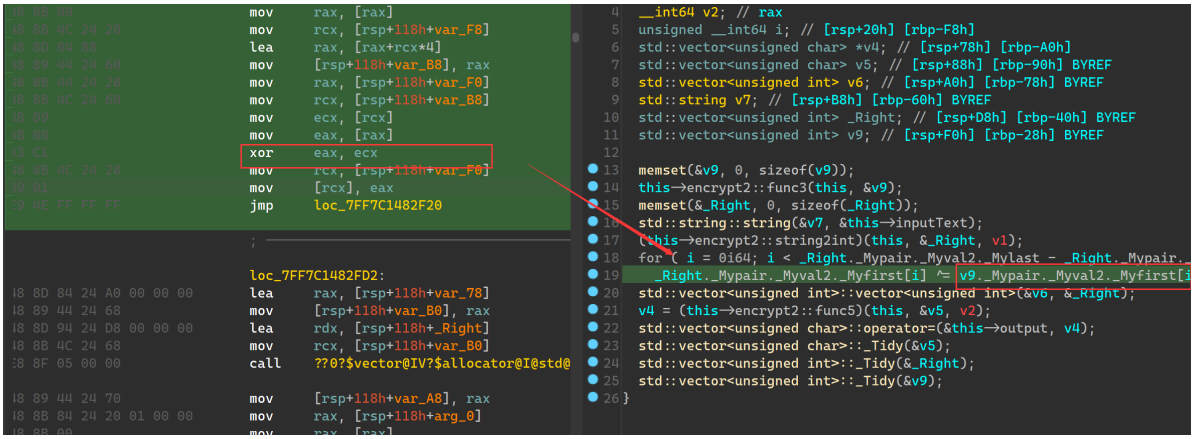
这里是函数的主题部分，是不是非常难看？特别是后面的((v8 + 16i64))(v8);(**v6)(v6);这两句。所以我们在这里将v8的类型改为encrypt2 *就可以看到清晰的函数调用

```
std::string input; // [rsp+80h] [rbp-38h] BYREF

std::string::string(&input);
std::operator>><char>(std::cin, &input);
v7 = operator new(0x70ui64);
if ( v7 )
{
  memset(v7, 0, sizeof(encrypt2));
  std::string::string(&v9, &input);
  encrypt2::encrypt2(v7, "hgame{th1s_is_4_fake_fl4g_hahaha}", 0x12345678u, "hgame{this_is_another_fake_flag}", v3);
  v8 = v4;
}
else
{
  v8 = 0i64;
}
v6 = v8;
v8->encrypt2::func2(v8);
v6->encrypt2::func6(v6);
if ( v6->encrypt2::func7(v6) )
  std::operator<<<std::char_traits<char>>(std::cout, "yes!");
else
  std::operator<<<std::char_traits<char>>(std::cout, "try again ... ");
std::string::_Tidy_deallocate(&input);
return 0;
```

进去fun2看看

```
p_data4 = &this->data4;
Myfirst = this->data4._Mypair._Myval2._M
*Myfirst = 0x61707865;
v4 = &this->data4;
v5 = this->data4._Mypair._Myval2._Myfirs
*v5 = 0x3320646E;
v6 = &this->data4;
v7 = this->data4._Mypair._Myval2._Myfirs
*v7 = 0x79622D32;
v8 = &this->data4;
v9 = this->data4._Mypair._Myval2._Myfirs
*v9 = 0x6B206574;
v10 = &this->data4;
```

这四个数是chacha20的加密特征。而这种加密是流密码的加密特征，所以我们只需要将与原文异或的密钥流找到就行，进入fun6可以看到一行异或逻辑



这就很明确了，异或使用的值存在了ecx寄存器中，写一个脚本就可以dump出来.

```
from idaapi import *
print(get_reg_val("ecx"),end = ',')
```

```
PDB: using MSDIA provider
1077387342,4258923078,1013905953,3483163055,1731413945,233590496,327206097,984787250,39669927,2202679682,
```

然后看到上面一行还调用了一个string2int函数

```
v28 = &v14->_Mypair._Myval2._Bx._Buf[i + 3];
v5 = *v28 + (*v27 << 8) + (*v26 << 16) + (*v25 << 24);
v29 = &v5;
v30 = &v5;
```

这是按照大端序存储的，所以我们在写脚本的时候需要注意一下

```
#include <stdio.h>

int main() {
    unsigned int key[10] = {1077387342, 4258923078, 1013905953, 3483163055, 1731413945, 233590496, 327206097, 984787250, 39669927, 2202679682};
    unsigned char enc[40] = {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};

    for (int i = 0; i < 10; i++) {
        printf("%c", enc[i * 4 + 0] ^ key[i] >> 24);
        printf("%c", enc[i * 4 + 1] ^ key[i] >> 16);
        printf("%c", enc[i * 4 + 2] ^ key[i] >> 8);
        printf("%c", enc[i * 4 + 3] ^ key[i] >> 0);
    }
}
```

其实我们还可以通过输入的密文与原文异或得到异或的值，这个关键就是找到密文的位置，我是通过“h”和它被加密后的值异或得到第一个key，然后用它加密“1”再在调试中跟进if判断语句，然后一个一个数据看，直到我想要的密文出现

```
enc = [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]
enc_str = [0x71, 0x05, 0x93, 0x7A, 0xC8, 0xEC, 0x35, 0x7E, 0x05, 0x5E, 0xCB,
0x13, 0xFC, 0xA8, 0xEC, 0x99, 0x50, 0x0B, 0x7E, 0x89, 0x3C, 0xDE, 0x7D, 0xD4,
0x26, 0xB6, 0xF3, 0xE9, 0x03, 0x82, 0x98, 0x00, 0x31, 0x69, 0x65, 0x91, 0xB4,
0x72, 0x00, 0xB2]
str1 = "12345678901234567890123456789012345678901234567890"
for i in range(40):
    print(chr(enc[i] ^ (enc_str[i] ^ ord(str1[i]))),end='')
```

4. 总结

这一周里面我发现自己的思维比较“僵硬”对一些问题的处理不够灵活，这是我之后主要的进攻方向，还有一个就是自己的遗憾，因为家里的原因，并没有打完hgame只打了前两周……