kunmusic

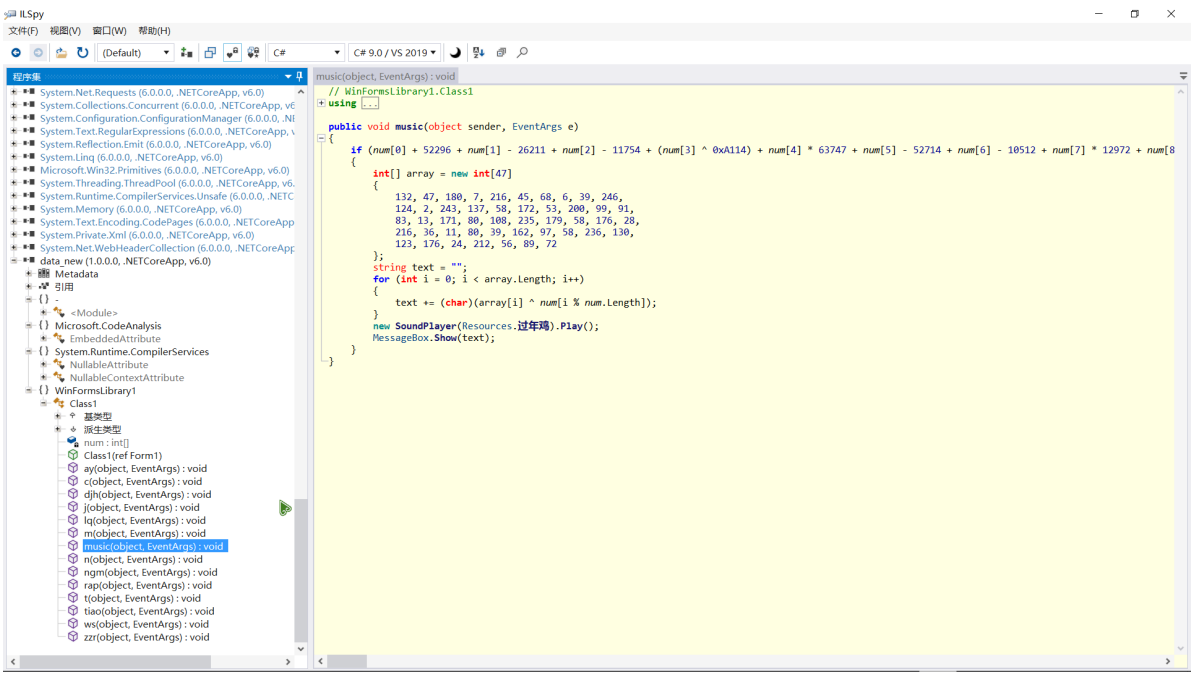
ILSpy打开，看看main函数

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

得知对data文件字节进行异或104操作

用python把data文件异或104，然后用ILspy打开data_new文件

```
with open('data', 'rb') as f:
    s = f.read()
    s = bytearray(s)
    for i in range(len(s)):
        s[i] ^= 104
    with open('data_new', 'wb') as f:
        f.write(s)
```



为了加快运行速度,直接由`hgame{`开头得到num[0]~num[5]

```
from z3 import *

num = [BitVec('num%d' % i, 50) for i in range(13)]

solver = Solver()
```

```
solver.add(num[0] + 52296 + num[1] - 26211 + num[2] - 11754 + (num[3] ^ 0xA114)
+ 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] ^ 0x525F) == 12702282)
solver.add(
    num[0] - 25228 + (num[1] ^ 0x50DB) + (num[2] ^ 0x1FDE) + num[3] - 65307 +
num[4] * 30701 + num[5] * 47555 + num[
        6] - 2557 + (num[7] ^ 0xBF9F) + num[8] - 7992 + (num[9] ^ 0xE079) +
(num[10] ^ 0xE052) + num[11] + 13299 + num[
        12] - 50966 == 9946829)
solver.add(num[0] - 64801 + num[1] - 60698 + num[2] - 40853 + num[3] - 54907 +
num[4] + 29882 + (num[5] ^ 0x3506) + (
        num[6] ^ 0x533E) + num[7] + 47366 + num[8] + 41784 + (num[9] ^ 0xD1BA) +
num[10] * 58436 + num[11] * 15590 +
           num[12] + 58225 == 2372055)
solver.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] ^ 0x995C) + num[
        12] + 25329 == 6732474)
solver.add(
    num[0] - 42567 + num[1] - 17743 + num[2] * 47827 + num[3] - 10246 + (num[4]
^ 0x3F9C) + num[5] + 39390 + num[
        6] * 11803 + num[7] * 60332 + (num[8] ^ 0x483B) + (num[9] ^ 0x12BB) +
num[10] - 25636 + num[11] - 16780 + num[
        12] - 62345 == 14020739)
solver.add(num[0] - 10968 + num[1] - 31780 + (num[2] ^ 0x7C71) + num[3] - 61983
+ num[4] * 31048 + num[5] * 20189 + num[
    6] + 12337 + num[7] * 25945 + (num[8] ^ 0x1B98) + num[9] - 25369 + num[10] -
54893 + num[11] * 59949 + (
                    num[12] ^ 0x3099) == 14434062)
solver.add(num[0] + 16689 + num[1] - 10279 + num[2] - 32918 + num[3] - 57155 +
num[4] * 26571 + num[5] * 15086 + (
        num[6] ^ 0x59CA) + (num[7] ^ 0x5B35) + (num[8] ^ 0x3FFD) + (num[9] ^
0x5A85) + num[10] - 40224 + num[
               11] + 31751 + num[12] * 8421 == 7433598)
solver.add(
    num[0] + 28740 + num[1] - 64696 + num[2] + 60470 + num[3] - 14752 + (num[4]
^ 0x507) + (num[5] ^ 0x89C8) + num[
        6] + 49467 + num[7] - 33788 + num[8] + 20606 + (num[9] ^ 0xAF4A) +
num[10] * 19764 + num[11] + 48342 + num[
        12] * 56511 == 7989404)
solver.add((num[0] ^ 0x7132) + num[1] + 23120 + num[2] + 22802 + num[3] * 31533
+ (num[4] ^ 0x9977) + num[5] - 48576 + (
        num[6] ^ 0x6F7E) + num[7] - 43265 + num[8] + 22365 + num[9] + 61108 +
num[10] * 2823 + num[11] - 30343 +
           num[12] + 14780 == 3504803)
solver.add(
    num[0] * 22466 + (num[1] ^ 0xDABF) + num[2] - 53658 + (num[3] ^ 0xB838) +
(num[4] ^ 0x30DF) + num[5] * 59807 + num[
        6] + 46242 + num[7] + 3052 + (num[8] ^ 0x62BF) + num[9] + 30202 +
num[10] * 22698 + num[11] + 33480 + (
            num[12] ^ 0x4175) == 11003580)
```

```
solver.add(
    num[0] * 57492 + (num[1] ^ 0x346D) + num[2] - 13941 + (num[3] ^ 0xBBDC) +
num[4] * 38310 + num[5] + 9884 + num[
        6] - 45500 + num[7] - 19233 + num[8] + 58274 + num[9] + 36175 + (num[10]
^ 0x4888) + num[11] * 49694 + (
            num[12] ^ 0x2501) == 25546210)
solver.add(num[0] - 23355 + num[1] * 50164 + (num[2] ^ 0x873A) + num[3] + 52703
+ num[4] + 36245 + num[5] * 46648 + (
        num[6] ^ 0x12FA) + (num[7] ^ 0xA376) + num[8] * 27122 + (num[9] ^
0xA44A) + num[10] * 15676 + num[
                11] - 31863 + num[12] + 62510 == 11333836)
solver.add(
    num[0] * 30523 + (num[1] ^ 0x1F36) + num[2] + 39058 + num[3] * 57549 +
(num[4] ^ 0xD0C0) + num[5] * 4275 + num[
        6] - 48863 + (num[7] ^ 0xD88C) + (num[8] ^ 0xA40) + (num[9] ^ 0x3554) +
num[10] + 62231 + num[11] + 19456 + num[
        12] - 13195 == 13863722)
solver.add(num[0] == 236, num[1] == 72, num[2] == 213, num[3] == 106, num[4] ==
189, num[5] == 86)
if solver.check() == sat:
    m = solver.model()
    for i in range(len(m)):
        print('num[%d]' % i, '=', int(str(m[num[i]])))
else:
    print('unsat')
'''
num[0] = 236
num[1] = 72
num[2] = 213
num[3] = 106
num[4] = 189
num[5] = 86
num[6] = 62
num[7] = 53
num[8] = 120
num[9] = 199
num[10] = 15
num[11] = 93
num[12] = 133
'''
```

最后得到flag

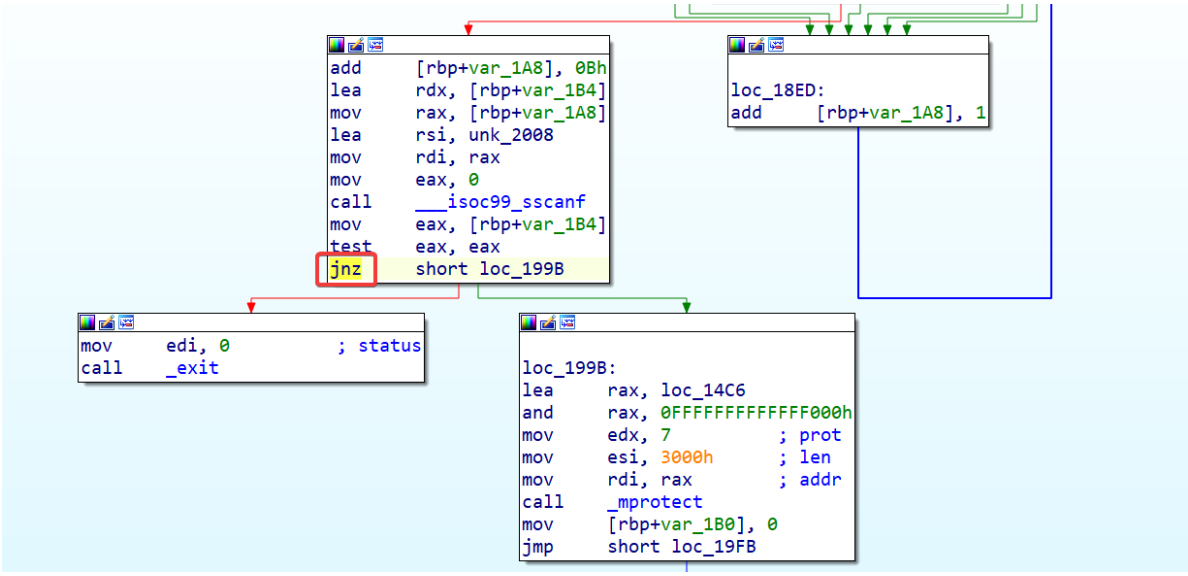
```
num = [0 for i in range(13)]
num[0] = 236
num[1] = 72
num[2] = 213
num[3] = 106
num[4] = 189
num[5] = 86
num[6] = 62
num[7] = 53
num[8] = 120
num[9] = 199
num[10] = 15
```

```
num[11] = 93
num[12] = 133

array = [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, 56, 89, 72]
for i in range(len(array)):
    print(chr(array[i] ^ num[i % len(num)]),end='')#
hgame{z3_1s_very_u5eful_1n_rever5e_engin3ering}
```

patchme

先过一下反调试,jz改成jnz让ida在调试状态不退出



此处使用mprotect,将一段内存区域标记为可读可写可执行,然后经过异或解密,所以可以直接去看这段解密后的代码

```
if ( !v2 )
  exit(0);
LODWORD(v0) = mprotect((void *)((unsigned __int64)sub_561C607864C6 & 0xFFFFFFFFFFFFF000LL), 0x3000uLL, 7);
for ( j = 0; j <= 960; ++j )
{
  v0 = (char *)sub_561C607864C6 + j;
  *v0 ^= 0x66u;
}
return (int)v0;
```

```
void __noreturn sub_56092A0864CA()
{
  __WAIT_STATUS stat_loc; // [rsp+Ch] [rbp-2C4h] BYREF
  int i; // [rsp+14h] [rbp-2BCh]
  __int64 v2; // [rsp+18h] [rbp-2B8h]
  int pipedes[2]; // [rsp+20h] [rbp-2B0h] BYREF
  int v4[2]; // [rsp+28h] [rbp-2A8h] BYREF
  char *argv[4]; // [rsp+30h] [rbp-2A0h] BYREF
  char v6[48]; // [rsp+50h] [rbp-280h] BYREF
  __int64 v7; // [rsp+80h] [rbp-250h]
  __int64 v8[5]; // [rsp+E0h] [rbp-1F0h]
  int v9; // [rsp+108h] [rbp-1C8h]
```

```
__int16 v10; // [rsp+10Ch] [rbp-1C4h]
char v11; // [rsp+10Eh] [rbp-1C2h]
__int64 v12[5]; // [rsp+110h] [rbp-1C0h]
int v13; // [rsp+138h] [rbp-198h]
__int16 v14; // [rsp+13Ch] [rbp-194h]
char v15; // [rsp+13Eh] [rbp-192h]
char buf[80]; // [rsp+140h] [rbp-190h] BYREF
char s1[8]; // [rsp+190h] [rbp-140h] BYREF
__int64 v18; // [rsp+198h] [rbp-138h]
char v19[280]; // [rsp+1A0h] [rbp-130h] BYREF
int v20; // [rsp+2B8h] [rbp-18h]
unsigned __int64 v21; // [rsp+2C8h] [rbp-8h]

v21 = __readfsqword(0x28u);
if ( dword_56092A089028 <= 1 )
{
  pipe(pipedes);
  pipe(v4);
  if ( fork() )
  {
    close(pipedes[0]);
    close(v4[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;
    v18 = 0LL;
    memset(v19, 0, sizeof(v19));
    v20 = 0;
    read(v4[0], s1, 0x12CuLL);
    wait((__WAIT_STATUS)&stat_loc);
    if ( !LODWORD(stat_loc.__uptr) && !strncmp(s1, buf, 0x14uLL) )
    {
      v8[0] = 0x5416D999808A28FALL;
      v8[1] = 0x588505094953B563LL;
      v8[2] = 0xCE8CF3A0DC669097LL;
      v8[3] = 0x4C5CF3E854F44CBDLL;
      v8[4] = 0xD144E49916678331LL;
      v9 = -631149652;
      v10 = -17456;
      v11 = 85;
      v12[0] = 0x3B4FA2FCEDEB4F92LL;
      v12[1] = 0x7E45A6C3B67EA16LL;
      v12[2] = 0xAFE1ACC8BF12D0E7LL;
      v12[3] = 0x132EC3B7269138CELL;
      v12[4] = 0x8E2197EB7311E643LL;
      v13 = -1370223935;
      v14 = -13899;
      v15 = 40;
```

```
        putchar(10);
        for ( i = 0; i <= 46; ++i )
          putchar((char)(*((_BYTE *)v8 + i) ^ *((_BYTE *)v12 + i)));
      }
      else
      {
        puts("\nthere are still bugs...");
      }
    }
    else
    {
      fflush(stdin);
      close(pipedes[1]);
      close(v4[0]);
      dup2(pipedes[0], 0);
      dup2(v4[1], 1);
      dup2(v4[1], 2);
      argv[0] = *(char **)qword_56092A089020;
      argv[1] = (char *)&unk_56092A087025;
      argv[2] = 0LL;
      sub_56092A086AA0(*(_QWORD *)qword_56092A089020, v6);
      v2 = v7;
      if ( v7 == 14472 )
        execve(*(const char **)qword_56092A089020, argv, 0LL);
      else
        puts("\nyou cannot modify the file size");
    }
  }
}
```

逻辑很清楚,直接写exp就行了

```
v8 = [0 for i in range(8)]
v8[0] = 0x5416D999808A28FA
v8[1] = 0x5885O5094953B563
v8[2] = 0xCE8CF3AODC669097
v8[3] = 0x4C5CF3E854F44CBD
v8[4] = 0xD144E49916678331
v8[5] = -631149652
v8[6] = -17456
v8[7] = 85
v12 = [0 for i in range(8)]
v12[0] = 0x3B4FA2FCEDEB4F92
v12[1] = 0x7E45A6C3B67EA16
v12[2] = 0xAFE1ACC8BF12D0E7
v12[3] = 0x132EC3B7269138CE
v12[4] = 0x8E2197EB7311E643
v12[5] = -1370223935
v12[6] = -13899
v12[7] = 40
for i in range(len(v8)):
    a = v8[i] ^ v12[i]
    print(bytearray.fromhex(hex(a)[2::]).decode()[::-1], end='')
    # hgame{You_4re_a_p@tch_master_Or_reverse_ma5ter}
```

cpp

通过这些数据可以确定是chacha20算法

```
sub_7FF64CD54DF0((a1 + 64), 16ui64, v2);
v3 = a1 + 64;
v4 = *(a1 + 64);
*v4 = 0x61707865;
v5 = a1 + 64;
v6 = (*(a1 + 64) + 4i64);
*v6 = 0x3320646E;
v7 = a1 + 64;
v8 = (*(a1 + 64) + 8i64);
*v8 = 0x79622D32;
v9 = a1 + 64;
v10 = (*(a1 + 64) + 12i64);
*v10 = 0x6B206574;
```

A BIGGER NONCE

Okay, we need that constant. Still, that bigger nonce would have been a good thing, especially for stateless systems: if you can't chose a random nonce, you kinda have to keep track of previously nonces, and that's not always practical, or even possible.

I know of 2 approaches. One comes from the IETF, in the rfc7539. You basically use 3 words for the nonce, and one for the counter:

```
block[ 0]: "expa"    block[ 8]: key[4]
block[ 1]: "nd 3"    block[ 9]: key[5]
block[ 2]: "2-by"    block[10]: key[6]
block[ 3]: "te k"    block[11]: key[7]
block[ 4]: key[0]    block[12]: counter
block[ 5]: key[1]    block[13]: nonce[0]
block[ 6]: key[2]    block[14]: nonce[1]
block[ 7]: key[3]    block[15]: nonce[2]
```

This makes it easier to deal with nonces, and even allows wasting a bunch of them. Still, random nonces aren't very safe yet, and now the message length limitation starts getting real. Though not many people send over 128Gb messages over the wire.

Still, we can do even better...

通过动态调试得知chacha20的 `key` 为 `hgame{th` , `counter` 为 `0x12345678` , `nonce` 为 `['h','g','a']`

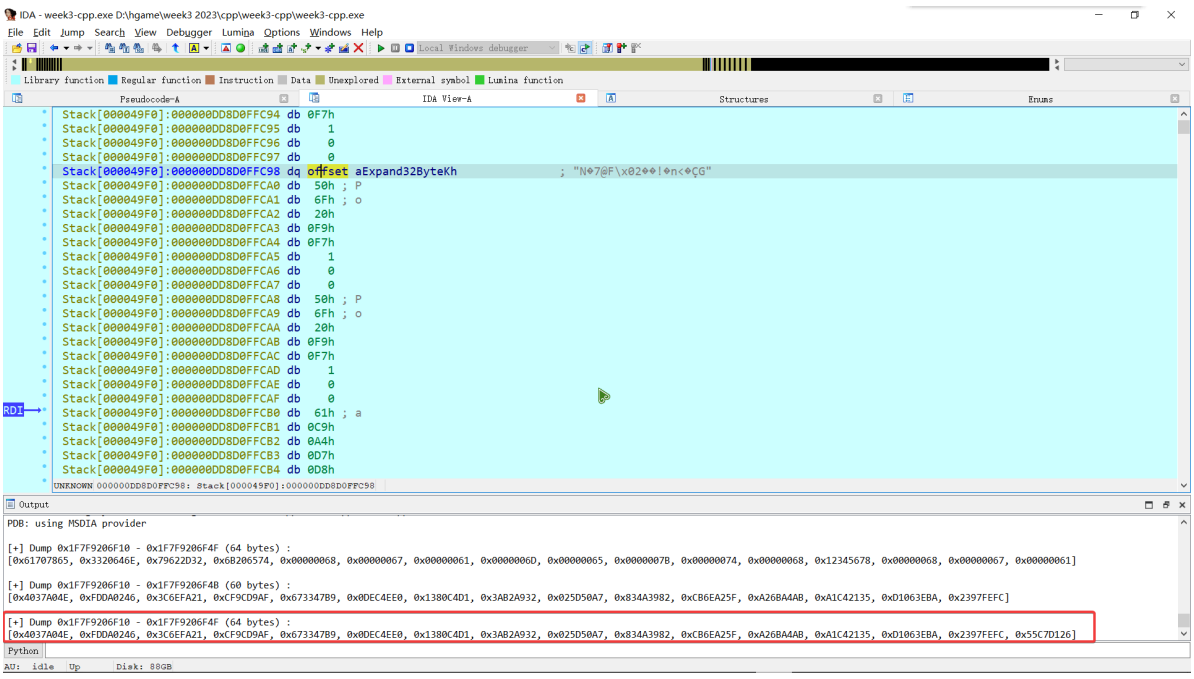
```
debug023:000002512ECB6EA0 dword_2512ECB6EA0 dd 'apxe'
debug023:000002512ECB6EA4 dd '3 dn'
debug023:000002512ECB6EA8 dd 'yb-2'
debug023:000002512ECB6EAC dd 'k et'
debug023:000002512ECB6EB0 dd 'h'
debug023:000002512ECB6EB4 dd 'g'
debug023:000002512ECB6EB8 dd 'a'
debug023:000002512ECB6EBC dd 'm'
debug023:000002512ECB6EC0 dd 'e'
debug023:000002512ECB6EC4 dd '{'
debug023:000002512ECB6EC8 dd 't'
debug023:000002512ECB6ECC dd 'h'
debug023:000002512ECB6ED0 dd 12345678h
debug023:000002512ECB6ED4 dd 'h'
debug023:000002512ECB6ED8 dd 'g'
debug023:000002512ECB6EDC dd 0
```

但是把数据放到chacha20算法里面却不能输出flag

所以我直接找到对矩阵进行操作的函数,

```
4  unsigned __int64 k; // [rsp+40h] [rbp-98h]
5  unsigned __int64 j; // [rsp+50h] [rbp-88h]
6  void *v6[3]; // [rsp+A8h] [rbp-30h] BYREF
7
8  memset(v6, 0, sizeof(v6));
9  sub_7FF64CD55150(v6, 16i64);
10 for ( i = 0i64; i < 0x10; ++i )
11   *((_DWORD *)v6[0] + i) = *(_DWORD *)(a1[8] + 4 * i);
12 for ( j = 0i64; j < 0xA; ++j )
13 {
14   (*(void (__fastcall **)(_QWORD *, _QWORD, __int64, __int64, __int64))(*a1 + 8i64))(a1, 0i64, 4i64, 8i64, 12i64);
15   (*(void (__fastcall **)(_QWORD *, __int64, __int64, __int64, __int64))(*a1 + 8i64))(a1, 1i64, 5i64, 9i64, 13i64);
16   (*(void (__fastcall **)(_QWORD *, __int64, __int64, __int64, __int64))(*a1 + 8i64))(a1, 2i64, 6i64, 10i64, 14i64);
17   (*(void (__fastcall **)(_QWORD *, __int64, __int64, __int64, __int64))(*a1 + 8i64))(a1, 3i64, 7i64, 11i64, 15i64);
18   (*(void (__fastcall **)(_QWORD *, _QWORD, __int64, __int64, __int64))(*a1 + 8i64))(a1, 0i64, 5i64, 10i64, 15i64);
19   (*(void (__fastcall **)(_QWORD *, __int64, __int64, __int64, __int64))(*a1 + 8i64))(a1, 1i64, 6i64, 11i64, 12i64);
20   (*(void (__fastcall **)(_QWORD *, __int64, __int64, __int64, __int64))(*a1 + 8i64))(a1, 2i64, 7i64, 8i64, 13i64);
21   (*(void (__fastcall **)(_QWORD *, __int64, __int64, __int64, __int64))(*a1 + 8i64))(a1, 3i64, 4i64, 9i64, 14i64);
22 }
23 for ( k = 0i64; k < 0x10; ++k )
24   *((_DWORD *)v6[0] + k) += *(_DWORD *)(a1[8] + 4 * k);
25 sub_7FF64CD533B0(a2, (__int64 *)v6);
26 sub_7FF64CD53CA0(v6);
27 return a2;
28}
```

并且得到了最终的矩阵



exp如下

```
import numpy as np
import struct
import hashlib
```

```
import warnings
from ctypes import *

# ChaCha20 Matrix: 16 words, 32-bits each (4 bytes each) for a total of 64-bytes
# cccccccc  cccccccc  cccccccc  cccccccc    #constants
# kkkkkkkk  kkkkkkkk  kkkkkkkk  kkkkkkkk    #key
# kkkkkkkk  kkkkkkkk  kkkkkkkk  kkkkkkkk    #key
# bbbbbbbb  nnnnnnnn  nnnnnnnn  nnnnnnnn    #block_number and nonce

# Disable overflow warnings for numpy uint as we need the overflow in ChaCha20's
addition
warnings.filterwarnings("ignore")


def quarterround(a, b, c, d):
    def circular_left(num, i):
        return ((num << i) & 0xFFFFFFFF) | (num >> (32 - i))

    # a, b, c, d are numpy uint32. The addition may overflow which is ok for
ChaCha20
    a += b
    d ^= a
    d = circular_left(d, 16)
    c += d
    b ^= c
    b = circular_left(b, 12)
    a += b
    d ^= a
    d = circular_left(d, 8)
    c += d
    b ^= c
    b = circular_left(b, 7)
    return (a, b, c, d)


def rounds(block):
    steps = [
        [0, 4, 8, 12],
        [1, 5, 9, 13],
        [2, 6, 10, 14],
        [3, 7, 11, 15],
        [0, 5, 10, 15],
        [1, 6, 11, 12],
        [2, 7, 8, 13],
        [3, 4, 9, 14],
    ]
    # print(type(block))
    for _ in range(10):
        for round in steps:
            block[round] = quarterround(*(block[round]))
    return block


def _xor(data_1, data_2):
    return [a ^ b for a, b in zip(data_1, data_2)]
```

```
def encrypt(plaintext):
    '''
    global matrix
    # Add the 4-byte block number
    # matrix[12] = counter
    state = matrix.copy()
    state = rounds(state)
    # Create the final state
    final = state + matrix
    # Serialize the final state
    # print(hex(c_uint32(final[0]).value))
    '''
    final = np.array(
        [0x4037A04E, 0xFDDA0246, 0x3C6EFA21, 0xCF9CD9AF, 0x673347B9, 0x0DEC4EE0,
0x1380C4D1, 0x3AB2A932, 0x025D50A7,
         0x834A3982, 0xCB6EA25F, 0xA26BA4AB, 0xA1C42135, 0xD1063EBA, 0x2397FEFC,
0x55C7D126], dtype=np.uint32)
    serial_out = struct.pack("<16L", *final)
    ciphertext = bytes(_xor(serial_out, plaintext))
    return ciphertext


def decrypt(ciphertext):
    return encrypt(ciphertext)

'''
matrix = np.array(
    [0x61707865, 0x3320646E, 0x79622D32, 0x6B206574,
     0x00000068, 0x00000067, 0x00000061, 0x0000006D,
     0x00000065, 0x0000007B, 0x00000074, 0x00000068,
     0x12345678, 0x00000068, 0x00000067, 0x00000061], dtype=np.uint32)
'''
_ciphertext = [40, 80, -63, 35, -104, -95, 65, 54, 76, 49, -53, 82, -112, -15,
-84, -52, 15, 108, 42, -119, 127, -33,
               17,
               -124,
               127, -26, -94, -32, 89, -57, -59, 70, 93, 41, 56, -109, -19, 21,
122, -1]
ciphertext = [0 for _ in range(40)]
for i in range(0, len(_ciphertext), 4):
    ciphertext[i] = _ciphertext[i + 3]
    ciphertext[i + 1] = _ciphertext[i + 2]
    ciphertext[i + 2] = _ciphertext[i + 1]
    ciphertext[i + 3] = _ciphertext[i]
# print(ciphertext)
for i in range(len(ciphertext)):
    if ciphertext[i] < 0:
        ciphertext[i] += 256
ciphertext = bytearray(ciphertext)
flag = decrypt(ciphertext)
for i in range(0, len(_ciphertext), 4):
```

```
 print(chr(flag[i+3]),chr(flag[i+2]),chr(flag[i+1]),chr(flag[i]),end='',sep='')#
hgame{Cpp_1s_much_mOr3_dlff1cult_th4n_C}
```