

RE题 Chaos2

解题思路:

IDA打开main函数里面有花指令的,发现都是一些无意义的跳转使用Nop大法即可,之后就几个函数,下面是关键函数.

```
sub_4017D0(v4, g_key, 128);
sub_4018A0(v4, v6, 128);
sub_401050("your flag is %s", v6);
sub_401050(Format_);
getchar();
return 0;
```

使用AI插件分析一下:

```
' ## 变更说明
' 1. 函数名: `RC4_Crypt` - 明确表示RC4算法的加密/解密功能
' 2. 参数:
'   - `stateArray[258]`: 取代原a1, 明确数组大小(256+2)
'   - `dataBuffer`: 取代原a2, 说明是数据处理缓冲区
'   - `dataLength`: 取代原n128, 使用无符号类型更合适
' 3. 返回类型: 改为`uint8_t`更精确表示返回值的范围
' 4. 添加了函数注释说明
'
' ---WPeChat_END---
var __cdecl sub_4018A0(_BYTE *a1, char *a2, int n128)

char result; // a1
char v5; // [esp+4h] [ebp-14h]
char v6; // [esp+8h] [ebp-10h]
int v7; // [esp+Ch] [ebp-Ch]
int v8; // [esp+10h] [ebp-8h]

LOBYTE(v8) = a1[256];
LOBYTE(v7) = a1[257];
while ( n128-- )
{
    v8 = (unsigned __int8)(v8 + 1);
    v6 = a1[v8];
    v7 = (unsigned __int8)(v6 + v7);
    v5 = a1[v7];
    a1[v8] = v5;
    a1[v7] = v6;
    *a2++ ^= a1[(unsigned __int8)(v5 + v6)];
}
a1[256] = v8;
result = v7;
a1[257] = v7;
return result;
```

发现是RC4加密算,再看其他几个函数.

```

6 // ### 建议的新函数名及参数:
7 // ...c
8 // unsigned char InitializeSBox(
9 //     unsigned char sBox[256], // 状态表(S-box)数组
10 //     const unsigned char *key, // 密钥数据指针
11 //     unsigned int keyLength // 密钥长度(字节数)
12 // )
13 // {
14 //     // 函数体保持不变
15 //     return sBox[0]; // 实际返回值可能是无意义的, 可以调整返回类型为 void
16 // }
17 // ...
18 //
19 // ### 返回类型,
20 // 原始的返回类型是 `char`, 但实际返回值可能是无意义的(比如循环后的 `result`)。可以修改为 `void`, 因为主要功能是初始化 `sBox`, 无需返回值。
21 //
22 // ---WPeChat_END---
23 char __cdecl sub_4017D0(_BYTE *a1, int g_key, unsigned int n128)
24 {
25     char result; // a1
26     char v4; // [esp+0h] [ebp-10h]
27     int v5; // [esp+4h] [ebp-Ch]
28     int v6; // [esp+8h] [ebp-8h]
29     unsigned int n0x100; // [esp+Ch] [ebp-4h]
30     unsigned int n0x100_1; // [esp+Ch] [ebp-4h]
31
32     v5 = 0;
33     LOBYTE(v6) = 0;
34     result = (char)a1;
35     a1[257] = 0;
36     a1[256] = 0;
37     for ( n0x100 = 0; n0x100 < 0x100; ++n0x100 )
38     {
39         result = n0x100 + (_BYTE)a1;
40         a1[n0x100] = n0x100;
41     }
42     for ( n0x100_1 = 0; n0x100_1 < 0x100; ++n0x100_1 )
43     {
44         v4 = a1[n0x100_1];
45         v6 = (unsigned __int8)(v6 + v4 + *(_BYTE *)(v5 + g_key));
46         a1[n0x100_1] = a1[(unsigned __int8)v6];
47         result = v4;
48         a1[v6] = v4;
49         if ( ++v5 >= n128 )
50             v5 = 0;
51     }
52     return result;
53 }

```

这个就是初始化密钥了,那剩下的找到Key就行了.

```

1 ] = -09;
2 ] = -18;
3 ] = 13;
4 ] = -42;
5 ] = 4;
6 ] = 115;
7 ] = 85;
8 ] = 94;
9 ] = 62;
10 ] = -109;
11 ] = -92;
12 ] = 52;
13 ] = 41;
14 ] = 103;
15 ] = -4;
16 ] = 35;
17 ] = 121;
18 ] = 25;
19 ] = -40;
20 ] = -55;
21 ] = 43;
22 ] = -49;

```

Direction	Type	Address	Text
Up	r	sub_401090:loc_40110A	mov ecx, g_key
Up	r	sub_401200:loc_40127B	mov edx, g_key
Up	r	sub_4012A0+CD	mov ecx, g_key
Up	r	sub_4013A0:loc_40141B	mov edx, g_key
Up	w	sub_401440+60	mov g_key, ecx
Up	r	sub_401440:loc_4014AA	mov eax, g_key
Up	r	_main+82	mov ecx, g_key

Line 5 of 7

OK Cancel Search Help

对着g_key按x查找引用,发现来自sub_401440+60,进去看下

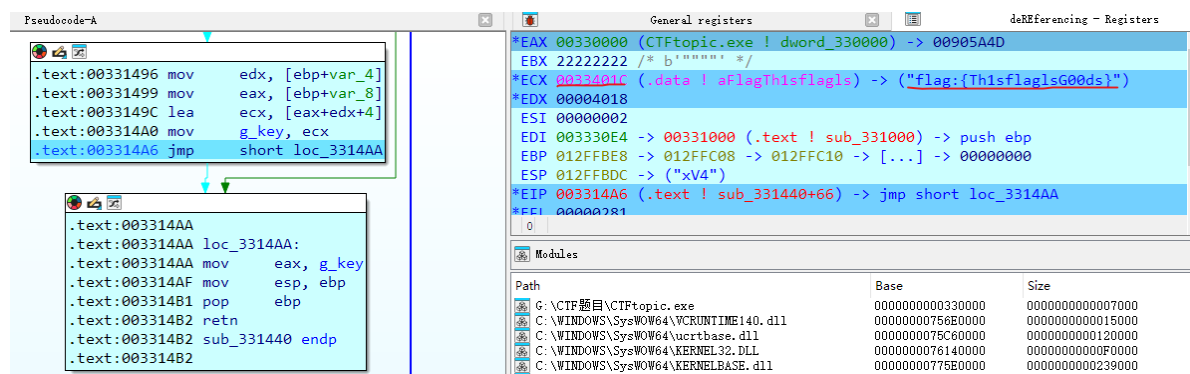
```

int sub_401440()
{
    HMODULE ModuleHandleA; // [esp+4h] [ebp-8h]
    unsigned int n0x10000; // [esp+8h] [ebp-4h]

    ModuleHandleA = GetModuleHandleA(0);
    for ( n0x10000 = 0; n0x10000 < 0x10000; ++n0x10000 )
    {
        if ( *(_DWORD *)((char *)ModuleHandleA + n0x10000) == 0x12345678 && *(_BYTE *)ModuleHandleA + n0x10000 + 4 != 0x75 )
        {
            g_key = (int)ModuleHandleA + n0x10000 + 4;
            return g_key;
        }
    }
    return g_key;
}

```

发现来自一个基址加偏移的位置,动态跟踪一下.



发现一个字符串,然后继续运行发现不会断在初始化算法的地方,使用xdbg试一下.

00331756	C645 FB CF	mov byte ptr ss:[ebp-5],CF	
0033175A	68 80000000	push 80	
0033175E	8B95 70FEFFFF	mov edx,dword ptr ss:[ebp-190]	[ebp-190]: "flag:{Th1sflaglsGo0ds}"
00331765	52	push edx	edx: "flag:{Th1sflaglsGo0ds}"
00331766	8D85 74FEFFFF	lea eax,dword ptr ss:[ebp-18C]	
0033176C	50	push eax	
0033176D	E8 5E000000	call ctftopic.3317D0	
00331772	83C4 0C	add esp,C	
00331775	68 80000000	push 80	
0033177A	8D8D 7CFFFFFF	lea ecx,dword ptr ss:[ebp-84]	
00331780	51	push ecx	ecx: "flag:{Th1sflaglsGo0ds}"
00331781	8D95 74FEFFFF	lea edx,dword ptr ss:[ebp-18C]	
00331787	52	push edx	edx: "flag:{Th1sflaglsGo0ds}"
00331788	E8 13010000	call ctftopic.3318A0	
0033178D	83C4 0C	add esp,C	

发现也有类似字符串,仔细看可以发现两者有点不一样,edx=0033401C "flag:{Th1sflaglsGo0ds}",运行之后发现是乱的,感觉是key不对,根据提示这是一个关于反调试的题目,到目前为止没感觉到又反调试.看看有哪些地方修改了0x0033401C地址,我在OEP断下后对内存下访问断点看看.测试发现就赋值的地方会断下.

回到ida中main函数反汇编可以看到有个回调函数.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    _BYTE v4[260]; // [esp+14h] [ebp-18Ch] BYREF
    int v5; // [esp+118h] [ebp-88h]
    char v6[128]; // [esp+11Ch] [ebp-84h] BYREF

    sub_331050("He said that if all the key modifications involved in anti-debugging are identified, th
    v5 = dword_334018;
    EnumUILanguagesA(UILanguageEnumProc, 0, 0);
    v6[0] = 15;
    v6[1] = 26;
    v6[2] = -118;
    v6[3] = 90;
    v6[4] = 34;
    v6[5] = -85;
    v6[6] = 30;
    v6[7] = 99;
    v6[8] = 25;
    v6[9] = 25;
    v6[10] = 25;
    v6[11] = 25;
    v6[12] = 25;
    v6[13] = 25;
    v6[14] = 25;
    v6[15] = 25;
    v6[16] = 25;
    v6[17] = 25;
    v6[18] = 25;
    v6[19] = 25;
    v6[20] = 25;
    v6[21] = 25;
    v6[22] = 25;
    v6[23] = 25;
    v6[24] = 25;
    v6[25] = 25;
    v6[26] = 25;
    v6[27] = 25;
    v6[28] = 25;
    v6[29] = 25;
    v6[30] = 25;
    v6[31] = 25;
    v6[32] = 25;
    v6[33] = 25;
    v6[34] = 25;
    v6[35] = 25;
    v6[36] = 25;
    v6[37] = 25;
    v6[38] = 25;
    v6[39] = 25;
    v6[40] = 25;
    v6[41] = 25;
    v6[42] = 25;
    v6[43] = 25;
    v6[44] = 25;
    v6[45] = 25;
    v6[46] = 25;
    v6[47] = 25;
    v6[48] = 25;
    v6[49] = 25;
    v6[50] = 25;
    v6[51] = 25;
    v6[52] = 25;
    v6[53] = 25;
    v6[54] = 25;
    v6[55] = 25;
    v6[56] = 25;
    v6[57] = 25;
    v6[58] = 25;
    v6[59] = 25;
    v6[60] = 25;
    v6[61] = 25;
    v6[62] = 25;
    v6[63] = 25;
    v6[64] = 25;
    v6[65] = 25;
    v6[66] = 25;
    v6[67] = 25;
    v6[68] = 25;
    v6[69] = 25;
    v6[70] = 25;
    v6[71] = 25;
    v6[72] = 25;
    v6[73] = 25;
    v6[74] = 25;
    v6[75] = 25;
    v6[76] = 25;
    v6[77] = 25;
    v6[78] = 25;
    v6[79] = 25;
    v6[80] = 25;
    v6[81] = 25;
    v6[82] = 25;
    v6[83] = 25;
    v6[84] = 25;
    v6[85] = 25;
    v6[86] = 25;
    v6[87] = 25;
    v6[88] = 25;
    v6[89] = 25;
    v6[90] = 25;
    v6[91] = 25;
    v6[92] = 25;
    v6[93] = 25;
    v6[94] = 25;
    v6[95] = 25;
    v6[96] = 25;
    v6[97] = 25;
    v6[98] = 25;
    v6[99] = 25;
    v6[100] = 25;
    v6[101] = 25;
    v6[102] = 25;
    v6[103] = 25;
    v6[104] = 25;
    v6[105] = 25;
    v6[106] = 25;
    v6[107] = 25;
    v6[108] = 25;
    v6[109] = 25;
    v6[110] = 25;
    v6[111] = 25;
    v6[112] = 25;
    v6[113] = 25;
    v6[114] = 25;
    v6[115] = 25;
    v6[116] = 25;
    v6[117] = 25;
    v6[118] = 25;
    v6[119] = 25;
    v6[120] = 25;
    v6[121] = 25;
    v6[122] = 25;
    v6[123] = 25;
    v6[124] = 25;
    v6[125] = 25;
    v6[126] = 25;
    v6[127] = 25;
    v6[128] = 25;
    v6[129] = 25;
    v6[130] = 25;
    v6[131] = 25;
    v6[132] = 25;
    v6[133] = 25;
    v6[134] = 25;
    v6[135] = 25;
    v6[136] = 25;
    v6[137] = 25;
    v6[138] = 25;
    v6[139] = 25;
    v6[140] = 25;
    v6[141] = 25;
    v6[142] = 25;
    v6[143] = 25;
    v6[144] = 25;
    v6[145] = 25;
    v6[146] = 25;
    v6[147] = 25;
    v6[148] = 25;
    v6[149] = 25;
    v6[150] = 25;
    v6[151] = 25;
    v6[152] = 25;
    v6[153] = 25;
    v6[154] = 25;
    v6[155] = 25;
    v6[156] = 25;
    v6[157] = 25;
    v6[158] = 25;
    v6[159] = 25;
    v6[160] = 25;
    v6[161] = 25;
    v6[162] = 25;
    v6[163] = 25;
    v6[164] = 25;
    v6[165] = 25;
    v6[166] = 25;
    v6[167] = 25;
    v6[168] = 25;
    v6[169] = 25;
    v6[170] = 25;
    v6[171] = 25;
    v6[172] = 25;
    v6[173] = 25;
    v6[174] = 25;
    v6[175] = 25;
    v6[176] = 25;
    v6[177] = 25;
    v6[178] = 25;
    v6[179] = 25;
    v6[180] = 25;
    v6[181] = 25;
    v6[182] = 25;
    v6[183] = 25;
    v6[184] = 25;
    v6[185] = 25;
    v6[186] = 25;
    v6[187] = 25;
    v6[188] = 25;
    v6[189] = 25;
    v6[190] = 25;
    v6[191] = 25;
    v6[192] = 25;
    v6[193] = 25;
    v6[194] = 25;
    v6[195] = 25;
    v6[196] = 25;
    v6[197] = 25;
    v6[198] = 25;
    v6[199] = 25;
    v6[200] = 25;
    v6[201] = 25;
    v6[202] = 25;
    v6[203] = 25;
    v6[204] = 25;
    v6[205] = 25;
    v6[206] = 25;
    v6[207] = 25;
    v6[208] = 25;
    v6[209] = 25;
    v6[210] = 25;
    v6[211] = 25;
    v6[212] = 25;
    v6[213] = 25;
    v6[214] = 25;
    v6[215] = 25;
    v6[216] = 25;
    v6[217] = 25;
    v6[218] = 25;
    v6[219] = 25;
    v6[220] = 25;
    v6[221] = 25;
    v6[222] = 25;
    v6[223] = 25;
    v6[224] = 25;
    v6[225] = 25;
    v6[226] = 25;
    v6[227] = 25;
    v6[228] = 25;
    v6[229] = 25;
    v6[230] = 25;
    v6[231] = 25;
    v6[232] = 25;
    v6[233] = 25;
    v6[234] = 25;
    v6[235] = 25;
    v6[236] = 25;
    v6[237] = 25;
    v6[238] = 25;
    v6[239] = 25;
    v6[240] = 25;
    v6[241] = 25;
    v6[242] = 25;
    v6[243] = 25;
    v6[244] = 25;
    v6[245] = 25;
    v6[246] = 25;
    v6[247] = 25;
    v6[248] = 25;
    v6[249] = 25;
    v6[250] = 25;
    v6[251] = 25;
    v6[252] = 25;
    v6[253] = 25;
    v6[254] = 25;
    v6[255] = 25;
    v6[256] = 25;
    v6[257] = 25;
    v6[258] = 25;
    v6[259] = 25;
    v6[260] = 25;
    v6[261] = 25;
    v6[262] = 25;
    v6[263] = 25;
    v6[264] = 25;
    v6[265] = 25;
    v6[266] = 25;
    v6[267] = 25;
    v6[268] = 25;
    v6[269] = 25;
    v6[270] = 25;
    v6[271] = 25;
    v6[272] = 25;
    v6[273] = 25;
    v6[274] = 25;
    v6[275] = 25;
    v6[276] = 25;
    v6[277] = 25;
    v6[278] = 25;
    v6[279] = 25;
    v6[280] = 25;
    v6[281] = 25;
    v6[282] = 25;
    v6[283] = 25;
    v6[284] = 25;
    v6[285] = 25;
    v6[286] = 25;
    v6[287] = 25;
    v6[288] = 25;
    v6[289] = 25;
    v6[290] = 25;
    v6[291] = 25;
    v6[292] = 25;
    v6[293] = 25;
    v6[294] = 25;
    v6[295] = 25;
    v6[296] = 25;
    v6[297] = 25;
    v6[298] = 25;
    v6[299] = 25;
    v6[300] = 25;
    v6[301] = 25;
    v6[302] = 25;
    v6[303] = 25;
    v6[304] = 25;
    v6[305] = 25;
    v6[306] = 25;
    v6[307] = 25;
    v6[308] = 25;
    v6[309] = 25;
    v6[310] = 25;
    v6[311] = 25;
    v6[312] = 25;
    v6[313] = 25;
    v6[314] = 25;
    v6[315] = 25;
    v6[316] = 25;
    v6[317] = 25;
    v6[318] = 25;
    v6[319] = 25;
    v6[320] = 25;
    v6[321] = 25;
    v6[322] = 25;
    v6[323] = 25;
    v6[324] = 25;
    v6[325] = 25;
    v6[326] = 25;
    v6[327] = 25;
    v6[328] = 25;
    v6[329] = 25;
    v6[330] = 25;
    v6[331] = 25;
    v6[332] = 25;
    v6[333] = 25;
    v6[334] = 25;
    v6[335] = 25;
    v6[336] = 25;
    v6[337] = 25;
    v6[338] = 25;
    v6[339] = 25;
    v6[340] = 25;
    v6[341] = 25;
    v6[342] = 25;
    v6[343] = 25;
    v6[344] = 25;
    v6[345] = 25;
    v6[346] = 25;
    v6[347] = 25;
    v6[348] = 25;
    v6[349] = 25;
    v6[350] = 25;
    v6[351] = 25;
    v6[352] = 25;
    v6[353] = 25;
    v6[354] = 25;
    v6[355] = 25;
    v6[356] = 25;
    v6[357] = 25;
    v6[358] = 25;
    v6[359] = 25;
    v6[360] = 25;
    v6[361] = 25;
    v6[362] = 25;
    v6[363] = 25;
    v6[364] = 25;
    v6[365] = 25;
    v6[366] = 25;
    v6[367] = 25;
    v6[368] = 25;
    v6[369] = 25;
    v6[370] = 25;
    v6[371] = 25;
    v6[372] = 25;
    v6[373] = 25;
    v6[374] = 25;
    v6[375] = 25;
    v6[376] = 25;
    v6[377] = 25;
    v6[378] = 25;
    v6[379] = 25;
    v6[380] = 25;
    v6[381] = 25;
    v6[382] = 25;
    v6[383] = 25;
    v6[384] = 25;
    v6[385] = 25;
    v6[386] = 25;
    v6[387] = 25;
    v6[388] = 25;
    v6[389] = 25;
    v6[390] = 25;
    v6[391] = 25;
    v6[392] = 25;
    v6[393] = 25;
    v6[394] = 25;
    v6[395] = 25;
    v6[396] = 25;
    v6[397] = 25;
    v6[398] = 25;
    v6[399] = 25;
    v6[400] = 25;
    v6[401] = 25;
    v6[402] = 25;
    v6[403] = 25;
    v6[404] = 25;
    v6[405] = 25;
    v6[406] = 25;
    v6[407] = 25;
    v6[408] = 25;
    v6[409] = 25;
    v6[410] = 25;
    v6[411] = 25;
    v6[412] = 25;
    v6[413] = 25;
    v6[414] = 25;
    v6[415] = 25;
    v6[416] = 25;
    v6[417] = 25;
    v6[418] = 25;
    v6[419] = 25;
    v6[420] = 25;
    v6[421] = 25;
    v6[422] = 25;
    v6[423] = 25;
    v6[424] = 25;
    v6[425] = 25;
    v6[426] = 25;
    v6[427] = 25;
    v6[428] = 25;
    v6[429] = 25;
    v6[430] = 25;
    v6[431] = 25;
    v6[432] = 25;
    v6[433] = 25;
    v6[434] = 25;
    v6[435] = 25;
    v6[436] = 25;
    v6[437] = 25;
    v6[438] = 25;
    v6[439] = 25;
    v6[440] = 25;
    v6[441] = 25;
    v6[442] = 25;
    v6[443] = 25;
    v6[444] = 25;
    v6[445] = 25;
    v6[446] = 25;
    v6[447] = 25;
    v6[448] = 25;
    v6[449] = 25;
    v6[450] = 25;
    v6[451] = 25;
    v6[452] = 25;
    v6[453] = 25;
    v6[454] = 25;
    v6[455] = 25;
    v6[456] = 25;
    v6[457] = 25;
    v6[458] = 25;
    v6[459] = 25;
    v6[460] = 25;
    v6[461] = 25;
    v6[462] = 25;
    v6[463] = 25;
    v6[464] = 25;
    v6[465] = 25;
    v6[466] = 25;
    v6[467] = 25;
    v6[468] = 25;
    v6[469] = 25;
    v6[470] = 25;
    v6[471] = 25;
    v6[472] = 25;
    v6[473] = 25;
    v6[474] = 25;
    v6[475] = 25;
    v6[476] = 25;
    v6[477] = 25;
    v6[478] = 25;
    v6[479] = 25;
    v6[480] = 25;
    v6[481] = 25;
    v6[482] = 25;
    v6[483] = 25;
    v6[484] = 25;
    v6[485] = 25;
    v6[486] = 25;
    v6[487] = 25;
    v6[488] = 25;
    v6[489] = 25;
    v6[490] = 25;
    v6[491] = 25;
    v6[492] = 25;
    v6[493] = 25;
    v6[494] = 25;
    v6[495] = 25;
    v6[496] = 25;
    v6[497] = 25;
    v6[498] = 25;
    v6[499] = 25;
    v6[500] = 25;
    v6[501] = 25;
    v6[502] = 25;
    v6[503] = 25;
    v6[504] = 25;
    v6[505] = 25;
    v6[506] = 25;
    v6[507] = 25;
    v6[508] = 25;
    v6[509] = 25;
    v6[510] = 25;
    v6[511] = 25;
    v6[512] = 25;
    v6[513] = 25;
    v6[514] = 25;
    v6[515] = 25;
    v6[516] = 25;
    v6[517] = 25;
    v6[518] = 25;
    v6[519] = 25;
    v6[520] = 25;
    v6[521] = 25;
    v6[522] = 25;
    v6[523] = 25;
    v6[524] = 25;
    v6[525] = 25;
    v6[526] = 25;
    v6[527] = 25;
    v6[528] = 25;
    v6[529] = 25;
    v6[530] = 25;
    v6[531] = 25;
    v6[532] = 25;
    v6[533] = 25;
    v6[534] = 25;
    v6[535] = 25;
    v6[536] = 25;
    v6[537] = 25;
    v6[538] = 25;
    v6[539] = 25;
    v6[540] = 25;
    v6[541] = 25;
    v6[542] = 25;
    v6[543] = 25;
    v6[544] = 25;
    v6[545] = 25;
    v6[546] = 25;
    v6[547] = 25;
    v6[548] = 25;
    v6[549] = 25;
    v6[550] = 25;
    v6[551] = 25;
    v6[552] = 25;
    v6[553] = 25;
    v6[554] = 25;
    v6[555] = 25;
    v6[556] = 25;
    v6[557] = 25;
    v6[558] = 25;
    v6[559] = 25;
    v6[560] = 25;
    v6[561] = 25;
    v6[562] = 25;
    v6[563] = 25;
    v6[564] = 25;
    v6[565] = 25;
    v6[566] = 25;
    v6[567] = 25;
    v6[568] = 25;
    v6[569] = 25;
    v6[570] = 25;
    v6[571] = 25;
    v6[572] = 25;
    v6[573] = 25;
    v6[574] = 25;
    v6[575] = 25;
    v6[576] = 25;
    v6[577] = 25;
    v6[578] = 25;
    v6[579] = 25;
    v6[580] = 25;
    v6[581] = 25;
    v6[582] = 25;
    v6[583] = 25;
    v6[584] = 25;
    v6[585] = 25;
    v6[586] = 25;
    v6[587] = 25;
    v6[588] = 25;
    v6[589] = 25;
    v6[590] = 25;
    v6[591] = 25;
    v6[592] = 25;
    v6[593] = 25;
    v6[594] = 25;
    v6[595] = 25;
    v6[596] = 25;
    v6[597] = 25;
    v6[598] = 25;
    v6[599] = 25;
    v6[600] = 25;
    v6[601] = 25;
    v6[602] = 25;
    v6[603] = 25;
    v6[604] = 25;
    v6[605] = 25;
    v6[606] = 25;
    v6[607] = 25;
    v6[608] = 25;
    v6[609] = 25;
    v6[610] = 25;
    v6[611] = 25;
    v6[612] = 25;
    v6[613] = 25;
    v6[614] = 25;
    v6[615] = 25;
    v6[616] = 25;
    v6[617] = 25;
    v6[618] = 25;
    v6[619] = 25;
    v6[620] = 25;
    v6[621] = 25;
    v6[622] = 25;
    v6[623] = 25;
    v6[624] = 25;
    v6[625] = 25;
    v6[626] = 25;
    v6[627] = 25;
    v6[628] = 25;
    v6[629] = 25;
    v6[630] = 25;
    v6[631] = 25;
    v6[632] = 25;
    v6[633] = 25;
    v6[634] = 25;
    v6[635] = 25;
    v6[636] = 25;
    v6[637] = 25;
    v6[638] = 25;
    v6[639] = 25;
    v6[640] = 25;
    v6[641] = 25;
    v6[642] = 25;
    v6[643] = 25;
    v6[644] = 25;
    v6[645] = 25;
    v6[646] = 25;
    v6[647] = 25;
    v6[648] = 25;
    v6[649] = 25;
    v6[650] = 25;
    v6[651] = 25;
    v6[652] = 25;
    v6[653] = 25;
    v6[654] = 25;
    v6[655] = 25;
    v6[656] = 25;
    v6[657] = 25;
    v6[658] = 25;
    v6[659] = 25;
    v6[660] = 25;
    v6[661] = 25;
    v6[662] = 25;
    v6[663] = 25;
    v6[664] = 25;
    v6[665] = 25;
    v6[666] = 25;
    v6[667] = 25;
    v6[668] = 25;
    v6[669] = 25;
    v6[670] = 25;
    v6[671] = 25;
    v6[672] = 25;
    v6[673] = 25;
    v6[674] = 25;
    v6[675] = 25;
    v6[676] = 25;
    v6[677] = 25;
    v6[678] = 25;
    v6[679] = 25;
    v6[680] = 25;
    v6[681] = 25;
    v6[682] = 25;
    v6[683] = 25;
    v6[684] = 25;
    v6[685] = 25;
    v6[686] = 25;
    v6[687] = 25;
    v6[688] = 25;
    v6[689] = 25;
    v6[690] = 25;
    v6[691] = 25;
    v6[692] = 25;
    v6[693] = 25;
    v6[694] = 25;
    v6[695] = 25;
    v6[696] = 25;
    v6[697] = 25;
    v6[698] = 25;
    v6[699] = 25;
    v6[700] = 25;
    v6[701] = 25;
    v6[702] = 25;
    v6[703] = 25;
    v6[704] = 25;
    v6[705] = 25;
    v6[706] = 25;
    v6[707] = 25;
    v6[708] = 25;
    v6[709] = 25;
    v6[710] = 25;
    v6[711] = 25;
    v6[712] = 25;
    v6[713] = 25;
    v6[714] = 25;
    v6[715] = 25;
    v6[716] = 25;
    v6[717] = 25;
    v6[718] = 25;
    v6[719] = 25;
    v6[720] = 25;
    v6[721] = 25;
    v6[722] = 25;
    v6[723] = 25;
    v6[724] = 25;
    v6[725] = 25;
    v6[726] = 25;
    v6[727] = 25;
    v6[728] = 25;
    v6[729] = 25;
    v6[730] = 25;
    v6[731] = 25;
    v6[732] = 25;
    v6[733] = 25;
    v6[734] = 25;
    v6[735] = 25;
    v6[736] = 25;
    v6[737] = 25;
    v6[738] = 25;
    v6[739] = 25;
    v6[740] = 25;
    v6[741] = 25;
    v6[742] = 25;
    v6[743] = 25;
    v6[744] = 25;
    v6[745] = 25;
    v6[746] = 25;
    v6[747] = 25;
    v6[748] = 25;
    v6[749] = 25;
    v6[750] = 25;
    v6[751] = 25;
    v6[752] = 25;
    v6[753] = 25;
    v6[754] = 25;
    v6[755] = 25;
    v6[756] = 25;
    v6[757] = 25;
    v6[758] = 25;
    v6[759] = 25;
    v6[760] = 25;
    v6[761] = 25;
    v6[762] = 25;
    v6[763] = 25;
    v6[764] = 25;
    v6[765] = 25;
    v6[766] = 25;
    v6[767] = 25;
    v6[768] = 25;
    v6[769] = 25;
    v6[770] = 25;
    v6[771] = 25;
    v6[772] = 25;
    v6[773] = 25;
    v6[774] = 25;
    v6[775] = 25
```

```

int __cdecl sub_401200(int (__stdcall *a1)(HANDLE, int, int *, int, _DWORD))
{
    int result; // eax
    HANDLE CurrentProcess; // [esp+Ch] [ebp-10h]
    int v3; // [esp+14h] [ebp-8h] BYREF

    CurrentProcess = GetCurrentProcess();
    result = a1(CurrentProcess, 7, &v3, 4, 0);
    n18 = 14;
    if ( !v3 )
        *(_BYTE *)(n18 + g_key) = 'I';
    return result;
}

```

g_key[8]='I';

```

int sub_401090()
{
    int v1; // [esp+Ch] [ebp-Ch]
    uint8_t BeingDebugged; // [esp+13h] [ebp-5h]

    BeingDebugged = NtCurrentPeb()->BeingDebugged;
    sub_401440();
    n18 = 8;
    if ( !BeingDebugged )
        *(_BYTE *)(n18 + g_key) = 'i';
    return v1;
}

```

g_key[18]='o';

```

int __cdecl sub_4013A0(int (__stdcall *a1)(HANDLE, int, int *, int, _DWORD))
{
    int result; // eax
    HANDLE CurrentProcess; // [esp+Ch] [ebp-10h]
    int v3; // [esp+14h] [ebp-8h] BYREF

    CurrentProcess = GetCurrentProcess();
    result = a1(CurrentProcess, 31, &v3, 4, 0);
    g_index = 18;
    if ( v3 == 1 )
        *(_BYTE *)(g_index + g_key) = 'o';
    return result;
}

```

g_key[17]='o';

```

.text:00401328 loc_401328: ; CODE XREF: sub_401200+124↑j
.text:00401328 mov     g_index, 11h
.text:00401332 push    offset ProcName ; "NtClose"
.text:00401337 mov     eax, [ebp+hModule]
.text:0040133A push    eax ; hModule
.text:0040133B call    ds:GetProcAddress
.text:00401341 mov     [ebp+var_28], eax
.text:00401344 cmp     [ebp+var_28], 0
.text:00401348 jnz     short loc_40134C
.text:0040134A jmp     short loc_401383
.text:0040134C ; -----
.text:0040134C loc_40134C: ; CODE XREF: sub_401200+148↑j
.text:0040134C ; __try { // __except at loc_40136A
.text:0040134C mov     [ebp+var_4], 0
.text:00401353 push    99999999h
.text:00401358 call    [ebp+var_28]
.text:00401358 ; } // starts at 40134C
.text:0040135B mov     [ebp+var_4], 0FFFFFFFh
.text:00401362 jmp     short loc_401383
.text:00401364 ; -----
.text:00401364 loc_401364: ; DATA XREF: .rdata:stru_4036804o
.text:00401364 ; __except filter // owned by 40134C
.text:00401364 mov     eax, 1
.text:00401369 retn
.text:0040136A ; -----
.text:0040136A loc_40136A: ; DATA XREF: .rdata:stru_4036804o
.text:0040136A ; __except(loc_401364) // owned by 40134C
.text:0040136A mov     esp, [ebp+var_18]
.text:0040136D mov     ecx, g_key
.text:00401373 add     ecx, g_index
.text:00401379 mov     byte ptr [ecx], 'o'
.text:0040137C mov     [ebp+var_4], 0FFFFFFFh
.text:00401383 loc_401383: ; CODE XREF: sub_401200+126↑j
.text:00401383 ; sub_401200+14A↑j ...
.text:00401383 mov     ecx, [ebp+var_10]
.text:00401386 mov     large fs:0, ecx
.text:0040138D pop     ecx
.text:0040138E pop     edi
.text:0040138F pop     esi
.text:00401390 pop     ebx
.text:00401391 mov     ecx, [ebp+var_1C]
.text:00401394 xor     ecx, ebp ; StackCookie
.text:00401396 call    @_security_check_cookie@4 ; __security_check_cookie(x)
.text:0040139B mov     esp, ebp
.text:0040139D pop     ebp

```

上面几个函数都是反调试检测，如果没用检测到调试就会把RC4算法的密钥还原，其中横线标记的函数逻辑有点特殊，他是检测到了才会还原，以此来对抗使用工具无脑使用反反调试工具的人。

密钥为“flag:{ThisflagIsGoods}”。

```

v6[111] = 115;
v6[112] = 85;
v6[113] = 94;
v6[114] = 62;
v6[115] = -109;
v6[116] = -92;
v6[117] = 52;
v6[118] = 41;
v6[119] = 103;
v6[120] = -4;
v6[121] = 35;
v6[122] = 121;
v6[123] = 25;
v6[124] = -40;
v6[125] = -55;
v6[126] = 43;
v6[127] = -49;

unsigned char keybuffer2[128]{ 'f','l','a','g',' ','{','}','T','h','i','s',' ','f','l'

_RC4_CONTEXT rc4;
rc4_init(&rc4, (unsigned char *)keybuffer2, 128);
rc4_crypt(&rc4, v6, 128);
printf("%s", v6);

```

之后通过rc4_crypt函数输出真正flag。

RCTF{AntiDbg_Reversing_2025_v2.0_Ch4llenge}

