看雪.纽盾 KCTF 2019 Q2 | 第四题点评及解题思路

KCTF 看雪学院 6月28日

连日的阴雨之后，上海的天气终于放晴，今天太阳露出明媚的笑脸，似乎我们的心情也变得更轻松舒畅些。

前三道题的题目解析我们已于看雪学院公众号平台持续放出，有疑惑的小伙伴请点击文章末尾原文链接。

今天我们对第四道题进行题目解析，一起来破解神秘的达芬奇密码。

题目简介

恭喜你成功进入金字塔，但这并不是结束。一阵冷风慢慢的由脚底爬到背后，让人不由得一哆嗦，映入眼帘的是两排高耸的圆形柱子，柱子上刻着古埃及的种种神话与传说。空气中弥漫着寂静的味道，只能听到你的脚步声和火把上火苗孜孜爆裂的声音。

沿着这排柱子走去，看到了道路那头坐着一扇长方形的门。门上刻着一串古埃及文字："open the door"，下方是一串密密麻麻的数字和符号.....能量宝石就藏在门后，只有破解这些密码，才能打开这扇门，成功获取能量宝石。

本题围观人数为2168人，观者众多，攻破人数和前三道题相比有所减少，数量为28人，纵观全局，在十道题之中攻破人数和难度都属于中等水平。

攻破此题的战队排名一览：

看雪评委crownless点评

本题有两个门槛，一个是变形，一个是序列号算法。变形非常的简单，追求的是趣味。算法的难点又有二，一个难点是还原算法的方程，另一个难点是求一个二元二次方程符合条件的整数解。

设计思路

本题出题战队**兔斯基保护协会**：

题目设计

本题有两个门槛，一个是变形，一个是序列号算法。

先说**变形部分**。

变形非常的简单，追求的是趣味。还有就是拦截一些直接IDA静态分析，过于草率的认为找到了序列号判断函数的玩家。

判断序列号是否有效的函数为TestKey，这个函数一开始的内容就是一个一般复杂的序列号判断程序（事实上解它的难度高于真实的判断逻辑）。程序运行之后，在初始化对话框的时候，会读取一块内存，解密后覆盖TestKey的代码内存，从而彻底的改变TestKey的逻辑。这就是本题简单的变形过程。

再说**序列号算法部分**。

只要程序跑起来，玩家就可以得到序列号算法的全部代码了。此处无混淆，除算法本身之外没有别的坑。

算法的难点又有二，一个难点是还原算法的方程，这个认真去逆或者有些“猜”的悟性可以很快得出方程。另一个难点是解方程，这个方程就是求一个二元二次方程符合条件的整数解。

方程为：x*x - 7y*y = 8
（参数是我深思熟虑反复调整的。这个方程的解序列对初始参数即为敏感，不会因为看起来简单而简单）

解的要求是：x长度为正好8字节，满足条件的有两组解，通过首位的判断把较大解排除，正确答案为较小的那个。从而避免多解。本题采用的大数算法排除了负解的可能性。（如果有人发现多解，我穿女装发自拍到群里）

至于方程的解怎么限定为0-9、A-Z、a-z的。我把用户的初始输入和预设字节流亦或了一下，以保证玩家要输入的正确序列号符合题目要求。

通过“输入在亦或后符合比赛要求”来排除部分输入后，蛮力搜索依然在复杂度上不可行。

常规堆题。简单直接的功能题，malloc,free,edit,show功能。同样直接的堆单字节溢出。知识点unlink+堆溢出。

功能分析:

show功能默认不可用。

edit功能默认可用一次。

malloc功能申请0x30大小堆块。单字节溢出。在程序init时提前申请了一个堆块。之后malloc功能申请出的地址必须在此堆块地址附近。直接给出heap地址。

破解思路

变形部分的破解太容易，只要程序跑起来，界面初始化完毕，变形就已经结束了。在GetDlgItemTextW下个断点，就能跑到确定长度的代码和判断序列号有效性的代码。喜欢dump出来慢慢分析，还是手动调试，看玩家口味。

然后，玩家发现长度限定为16的字符，通过判断序列号的函数逆出判断方程x*x - 7y*y = 8.这个方程，凭借蛮力枚举是解不了的。

有两种解法，

第一种解法

先枚举较小的解，排成一列，分析其规律~
最小的解(x,y)为(6,2).
其它解从小到大为：(90,34), (1434,542)...
有耐心的玩家会觉得，这有个递推公式吧?
试一试，构造一下。
发现递推公式是:
x[k + 1] = 8x[k] + 21y[k]
y[k + 1] = 3x[k] + 8y[k]
是不是很线性~
x[0],y[0]就是6.2.
15次递推之内就能得出本题要求的结果~

第二种解法

用线性代数方法求X[k+1]=M*X[k]，X为解序列的数对，M为2*2待定系数的矩阵，把(6,2)(90,34)(1434,542)代入确定系数，绝大多数递推公式都能这么不浪漫的机械求出。

只要用户想到从序列找规律，就不难了。所以本题叫序列探秘（Sequence Adventure），这本身也起到提示作用。

解题思路

本题解题思路由看雪论坛 **htg** 提供:

工具：IDA、Python、Notepad++、Resource_Hacker

一、题目主要流程

1、获取用户输入字符串sn：16位
2、解密函数代码sub_4010E0
3、进入 sub_4010E0内
4、将输入字符串sn分割为两个8位字符串，变成两个超大整数sn[0]、sn[1]，各8个字节长
5、两个超大整数与内置的两个超大整数异或运算，得到新的两个超大整数 X、Y
6、检测：X、Y内各个字节不能有00存在：目的是为了保证高位字节不能为00，也就是避免了多解的可能
7、检测：第7个字节（从0计算），需为01至0F之间，经过异或运算0x64还原，该输入字节只能为0x60-0x6F：即 ` a b c ^ o
8、检测：X^2 - 7 * Y^2 =8；这个是重点，丢番图方程，通过搜索网页，才知道这个方程解法
9、通过，即为正确字符串，L3mZ2k9aZ0a36DMM

二、解题思路

1、还原 sub_4010E0 ，Patch掉主程序，便于静态分析

1.1、找到关键CALL：004010E0
1.2、004010E0是由从00567B8拷贝0x330字节
1.3、00567B8的内容是通过XOR 0xAB算出来的
1.4、Patch文件，便于静态分析
1.4.1、修改 XOR 地址处的汇编为90
1.4.2、修改 00567B8的内容为解密的内容，即算出0xAB后，复原

2、学习丢番图方程的通解 X^2 - 7 * Y^2 =8 ： 找到满足条件的解

具体可以学习丢番图方程的解法，该通解是：(6 + 2 * sqrt(7)) (8 + 3 * sqrt(7))^n；我们可以通过程序来列出在一定范围的解

3、根据上述流程，反推输入字符串

三、主要源代码分析过程

1、IDA内找到软件入口：

因为是MFC程序，所以我们要找到按钮的触发方法才行。

方法：找到Button等控件的ID，通过 Resource_Hacker查找，打开Dialog表，

```
102 DIALOGEX 0, 0, 319, 117
STYLE DS_FIXEDSYS | DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION
EXSTYLE WS_EX_APPWINDOW
CAPTION "SequenceAdventure (CTF2019)"
LANGUAGE LANG_CHINESE, 0x2
FONT 9, "MS Shell Dlg"
{
CONTROL "Check", 1, BUTTON, BS_DEFPUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 7,
CONTROL "", 1000, EDIT, ES_LEFT | ES_AUTOHSCROLL | WS_CHILD | WS_VISIBLE | WS_BORDER |
CONTROL "Please Input The Serial:", -1, STATIC, SS_LEFT | WS_CHILD | WS_VISIBLE | WS_G
CONTROL "Close", 1002, BUTTON, BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 241
}
```

我们能发现 Button Check的ID为1（估计是作者故意的，搞那么小）；附近有一个Close，其ID是1002（0x3EA），我们通过它来找到对应的dialog。

IDA：Search 搜索立即数：0x3EA，很快就在.rdata里定位到具体内容
大佬一般看到后，就能很快找到对应的处理方法，我们可以先添加具体的消息处理结果，便于理解，添加两个结构 AFX_MSGMAP_ENTRY和AFX_MSGMAP，主要先后添加：
IDA→View→ Sub Views→Local Types：按Insert键，先后插入两个消息定义。

```
struct AFX_MSGMAP_ENTRY
{
UINT nMessage;
UINT nCode;
UINT nID;
UINT nLastID;
UINT_PTR nSig;
void (*pfn)(void);
```

```
};

struct AFX_MSGMAP
{
const AFX_MSGMAP *(__stdcall *pfnGetBaseMap)();
const AFX_MSGMAP_ENTRY *lpEntries;
};
```

选中刚才定义的消息（拖入到最下面，能看到最新添加的），然后右键，同步到 idb。

现在定位到刚才的 .rdata 的0x3EA处。

```
.rdata:005456D8 db 11h
.rdata:005456D9 db 1
.rdata:005456DA db 0
.rdata:005456DB db 0
.rdata:005456DC db 0
.rdata:005456DD db 0
.rdata:005456DE db 0
.rdata:005456DF db 0
.rdata:005456E0 db 0EAh
.rdata:005456E1 db 3
.rdata:005456E2 db 0
.rdata:005456E3 db 0
.rdata:005456E4 db 0EAh
.rdata:005456E5 db 3
.rdata:005456E6 db 0
.rdata:005456E7 db 0
.rdata:005456E8 db 39h ; 9
.rdata:005456E9 db 0
.rdata:005456EA db 0
.rdata:005456EB db 0
.rdata:005456EC dd offset sub_402000
```

在 .rdata:005456D8　按 Art +Q键盘，选择AFX_MSGMAP_ENTRY， 最后，形成的结构如下所述：

```
.rdata:00545690 stru_545690 AFX_MSGMAP_ENTRY <0Fh, 0, 0, 0, 13h, offset sub_401DE0>
.rdata:00545690 ; DATA XREF: .rdata:stru_545708↓o
.rdata:005456A8 AFX_MSGMAP_ENTRY <37h, 0, 0, 0, 28h, offset sub_401E90>
.rdata:005456C0 AFX_MSGMAP_ENTRY <111h, 0, 1, 1, 39h, offset sub_401EA0>
.rdata:005456D8 AFX_MSGMAP_ENTRY <111h, 0, 3EAh, 3EAh, 39h, offset sub_402000>
```

```
.rdata:005456F0 AFX_MSGMAP_ENTRY <0>
.rdata:00545708 stru_545708 AFX_MSGMAP <offset sub_4055EC, offset stru_545690>
```

然后，我们比对一下刚才在 Resource_Hacker查找的Dialog内容，很容易定位到button的ID为1 ，即check对应的是.rdata:005456C0 AFX_MSGMAP_ENTRY <111h, 0, 1, 1, 39h, offset sub_401EA0>，然后我们双击 offset sub_401EA0 进入到该方法，按F5，很容易判断即为button check的处理方法。

```
int __thiscall btnCheck(CWnd *this)
{
CWnd *v1; // esi
int i; // eax
WCHAR inputStr; // [esp+Ch] [ebp-310h]
char v5; // [esp+Eh] [ebp-30Eh]
char newInputStr; // [esp+20Ch] [ebp-110h]
char v7; // [esp+20Dh] [ebp-10Fh]
DWORD v8; // [esp+30Ch] [ebp-10h]
CWnd *v9; // [esp+310h] [ebp-Ch]
int v10; // [esp+314h] [ebp-8h]
DWORD flOldProtect; // [esp+318h] [ebp-4h]

v1 = this;
v9 = this;
inputStr = 0;
memset(&v5, 0, 0x1FEu);
newInputStr = 0;
memset(&v7, 0, 0xFFu);
CWnd::GetDlgItemTextW(v1, 1000, &inputStr, 20);
if ( wcslen(&inputStr) == 16 ) // 输入字符串长度：16
{
i = 0;
while ( !(*(&inputStr + i) & 0xFF00) ) // 字符应该是 0001 至 00FF；不能为 FF00 和 0100
{
*(&newInputStr + i) = *((_BYTE *)&inputStr + 2 * i);
if ( ++i >= 16 )
{
v8 = 0x40;
flOldProtect = 0;
VirtualProtect(sub_4010E0, 0xD17u, 0x40u, &flOldProtect);// 0x40：PAGE_EXECUTE_READWRI
if ( GetLastError() )
return CWnd::MessageBoxW(v1, L"Wrong!", 0, 0);
qmemcpy(sub_4010E0, byte_5647B8, 0x330u);
VirtualProtect(sub_4010E0, 0xD17u, flOldProtect, &v8);
```

```
if ( !GetLastError() )
{
v10 = 0;
v10 = sub_4010E0(&newInputStr);
if ( v10 == 1 )
return CWnd::MessageBoxW(v9, L"Congratulations! You are right!", 0, 0);
}
v1 = v9;
return CWnd::MessageBoxW(v1, L"Wrong!", 0, 0);
}
}
}
return CWnd::MessageBoxW(v1, L"Wrong!", 0, 0);
}
```

2、分析主函数 sub_401EA0

里面有一个VirtualProtect(sub_4010E0, 0xD17u, 0x40u, &flOldProtect);这里有修改qmemcpy(sub_4010E0, byte_5647B8, 0x330u);

很明显，它将 byte_5647B8内容覆盖了 sub_4010E0，继续跟进 byte_5647B8

```
.data:005647B8 ; char byte_5647B8[816]
.data:005647B8 2A byte_5647B8 db 2Ah ; DATA XREF: btnCheck+E0↑o
.data:005647B9 47 db 47h ; G
.data:005647BA 3B db 3Bh ; ;
.data:005647BB AB db 0ABh
.data:005647BC AB db 0ABh
.data:005647BD AB db 0ABh
.data:005647BE FD db 0FDh
.data:005647BF 1B db 1Bh
.data:005647C0 2A db 2Ah ; *
.data:005647C1 FC db 0FCh
.data:005647C2 20 db 20h
.data:005647C3 17 db 17h
```

```
.data:005647B8 2A byte_5647B8 db 2Ah ; DATA XREF: sub_401D80:loc_401DC0↑w
.data:005647B9 47 db 47h ; G
.data:005647BA 3B db 3Bh ; ;
.data:005647BB AB db 0ABh
.data:005647BC AB db 0ABh
```

```
.data:005647BD AB db 0ABh
.data:005647BE FD db 0FDh
.data:005647BF 1B db 1Bh
.data:005647C0 2A db 2Ah ; *
.data:005647C1 FC db 0FCh
.data:005647C2 20 db 20h
.data:005647C3 17 db 17h
```

很明显，这个不是代码，我们看看有哪些代码访问了它，按x，找到条目（我在原代码做了Patch，其实该代码处有两处引用），一处是check处理；一处是修改 byte_5647B8，我们跟进sub_401D80，查看：

```
signed int __thiscall sub_401D80(CDialog *this)
{
CDialog *v1; // esi
unsigned int v2; // eax

v1 = this;
CDialog::OnInitDialog(this);
SendMessageW(*((HWND *)v1 + 8), 0x80u, 1u, *((_DWORD *)v1 + 29));
SendMessageW(*((HWND *)v1 + 8), 0x80u, 0, *((_DWORD *)v1 + 29));
v2 = 0;
do
{
byte_5647B8[v2] ^= 0xABu;
++v2;
}
while ( v2 < 0x330 );
return 1;
}
```

它相当于对 byte_5647B8做了异或处理后又保存了，相当于原来进行了加密，现在进行了解密。

3、Patch代码：以使其更容易静态分析

我们做两件事：一是通过异或解密覆盖sub_4010E0；二是nop掉赋值语句。

3.1、 异或解密覆盖sub_4010E0，使用IDC代码处理。

```
static main(void)
{
auto fp, begin, end, dexbyte, sourceAdd;
sourceAdd = 0x5647B8;
begin = 0x4010E0;
end = begin + 0x330;
for ( dexbyte = begin; dexbyte < end; dexbyte ++ , sourceAdd ++ )
{
PatchByte(dexbyte,Byte(sourceAdd) ^ 0xAB);
}
}
```

3.2、 nop掉赋值语句

.text:00401F8A F3 A5 ===> 90 90//也可以将 text:00401F80- text:00401F8B处均nop

```
.text:00401F48 8B 1D 10 D2 51 00 mov ebx, ds:VirtualProtect
.text:00401F4E 8D 45 FC lea eax, [ebp+flOldProtect]
.text:00401F51 50 push eax ; lpflOldProtect
.text:00401F52 6A 40 push 40h ; flNewProtect
.text:00401F54 68 17 0D 00 00 push 0D17h ; dwSize
.text:00401F59 68 E0 10 40 00 push offset sub_4010E0 ; lpAddress
.text:00401F5E C7 45 F0 40 00 00 00 mov [ebp+var_10], 40h
.text:00401F65 C7 45 FC 00 00 00 00 mov [ebp+flOldProtect], 0
.text:00401F6C FF D3 call ebx ; VirtualProtect
.text:00401F6E FF 15 08 D4 51 00 call ds:GetLastError
.text:00401F74 85 C0 test eax, eax
.text:00401F76 75 62 jnz short loc_401FDA
.text:00401F78 8B 55 FC mov edx, [ebp+flOldProtect]
.text:00401F7B B9 CC 00 00 00 mov ecx, 0CCh
.text:00401F80 90 nop
.text:00401F81 90 nop
.text:00401F82 90 nop
.text:00401F83 90 nop
.text:00401F84 90 nop
.text:00401F85 90 nop
.text:00401F86 90 nop
.text:00401F87 90 nop
.text:00401F88 90 nop
.text:00401F89 90 nop
.text:00401F8A 90 nop
.text:00401F8B 90 nop
```

```
.text:00401F8C 8D 4D F0 lea ecx, [ebp+var_10]
.text:00401F8F 51 push ecx ; lpflOldProtect
.text:00401F90 52 push edx ; flNewProtect
.text:00401F91 68 17 0D 00 00 push 0D17h ; dwSize
.text:00401F96 68 E0 10 40 00 push offset sub_4010E0 ; lpAddress
```

3.3、保存

IDA：Edit→Patch Program→Assembly

4、静态分析 sub_4010E0

F5分析伪代码

```
signed int __cdecl sub_4010E0(int a1)//a1为输入字符串，16字节
{
signed int v1; // eax
char v2; // cl
signed int v3; // ecx
signed int v4; // eax
signed int v5; // eax
signed int v6; // esi
signed int v7; // ecx
__int16 v8; // dx
char *v9; // edi
__int16 v10; // ax
signed int v11; // eax
signed int v12; // ecx
unsigned __int16 v13; // bx
signed int v14; // esi
signed int v15; // ecx
__int16 v16; // dx
char *v17; // edi
__int16 v18; // ax
signed int v19; // eax
signed int v20; // ecx
unsigned __int16 v21; // bx
unsigned int v22; // eax
signed int v23; // ecx
unsigned __int16 v24; // dx
char v25; // dl
signed int v26; // eax
__int16 v27; // si
```

```
int v28; // eax
struc_1 v30; // [esp+8h] [ebp-90h]
int v31; // [esp+1Ch] [ebp-7Ch]
int v32; // [esp+20h] [ebp-78h]
int v33; // [esp+24h] [ebp-74h]
int v34[2]; // [esp+28h] [ebp-70h]
int v35[4]; // [esp+30h] [ebp-68h]
int v36[2]; // [esp+40h] [ebp-58h]
struc_1 v37; // [esp+48h] [ebp-50h]
struc_1 v38; // [esp+5Ch] [ebp-3Ch]
struc_1 v39; // [esp+70h] [ebp-28h]
struc_1 v40; // [esp+84h] [ebp-14h]

v30.intArray[1] = 0x646E9881;
v1 = 0;
v30.intArray[0] = 0xE38C9616;
v30.intArray[2] = 0x81DC0884;
v30.intArray[3] = 0x4F484DBE;
*(_DWORD *)&v30.charPtr = 0;
v31 = 0;
v32 = 0;
v33 = 0;
v34[0] = 0;
v34[1] = 0;
v36[0] = 0;
v36[1] = 0;
do
{
v2 = *((_BYTE *)&v30.intArray[2] + v1) ^ *(_BYTE *)(a1 + v1 + 8);
*((_BYTE *)v34 + v1) = *((_BYTE *)v30.intArray + v1) ^ *((_BYTE *)v30.intArray + v1 +
*((_BYTE *)v36 + v1++) = v2;
}//v34存放前8个字节异或结果；//v36存放后8个字节异或结果
while ( v1 < 8 );
v30.intArray[0] = 0;
v39.intArray[0] = 0;
v39.intArray[1] = 0;
v39.intArray[2] = 0;
v39.intArray[3] = 0;
v39.charPtr = 0;
v40.intArray[0] = 0;
v40.intArray[1] = 0;
v40.intArray[2] = 0;
v40.intArray[3] = 0;
v40.charPtr = 0;
v37.intArray[0] = 0;
```

```
v37.intArray[1] = 0;
v37.intArray[2] = 0;
v37.intArray[3] = 0;
v37.charPtr = 0;
v38.intArray[0] = 0;
v38.intArray[1] = 0;
v38.intArray[2] = 0;
v38.intArray[3] = 0;
v38.charPtr = 0;
v30.intArray[1] = 0;
v30.intArray[2] = 0;
v30.intArray[3] = 0;
v30.charPtr = 0;
v3 = 8;
LOBYTE(v30.intArray[0]) = 8;//v30=8
v4 = 7;
do
{
if ( *((_BYTE *)v34 + v4) )//异或后字节不能为00，避免多解
break;
--v3;
--v4;
}
while ( v4 >= 0 );
if ( v3 == 8 )
{
v5 = 7;
do
{
if ( *((_BYTE *)v36 + v5) )//异或后字节不能为00，避免多解
break;
--v3;
--v5;
}
while ( v5 >= 0 );
if ( v3 == 8 && !(v34[1] & 0xF0000000) )//第7个字节（0开始计数）智能为a-o；貌似没有意义
{
v6 = 0;
do
{
v35[0] = 0;
v35[1] = 0;
v35[2] = 0;
v35[3] = 0;
v7 = 0;
```

```
v8 = *((unsigned __int8 *)v34 + v6);
v9 = (char *)v35 + v6;
do
{
v10 = *((unsigned __int8 *)&v35[2] + v6) + v8 * *((unsigned __int8 *)v34 + v7);
v9[v7] = *((_BYTE *)&v35[2] + v6) + v8 * *((_BYTE *)v34 + v7);
++v7;
*((_BYTE *)&v35[2] + v6) = HIBYTE(v10);
}
while ( v7 < 8 );
LOBYTE(v11) = 0;
v12 = 0;
do
{
v13 = (char)v11 + *((unsigned __int8 *)v39.intArray + v12 + v6) + (unsigned __int8)v9[
*((_BYTE *)v39.intArray + v12++ + v6) = v13;
v11 = (signed int)v13 >> 8;
}
while ( v12 < 9 );
++v6;
}
while ( v6 < 8 );//执行完了后，v39=X^2（X、Y为分别为前、后8个字节异或后的整数）
v14 = 0;
do
{
v35[0] = 0;
v35[1] = 0;
v35[2] = 0;
v35[3] = 0;
v15 = 0;
v16 = *((unsigned __int8 *)v36 + v14);
v17 = (char *)v35 + v14;
do
{
v18 = *((unsigned __int8 *)&v35[2] + v14) + v16 * *((unsigned __int8 *)v36 + v15);
v17[v15] = *((_BYTE *)&v35[2] + v14) + v16 * *((_BYTE *)v36 + v15);
++v15;
*((_BYTE *)&v35[2] + v14) = HIBYTE(v18);
}
while ( v15 < 8 );
LOBYTE(v19) = 0;
v20 = 0;
do
{
v21 = (char)v19 + *((unsigned __int8 *)v40.intArray + v20 + v14) + (unsigned __int8)v1
```

```
*((_BYTE *)v40.intArray + v20++ + v14) = v21;
v19 = (signed int)v21 >> 8;
}
while ( v20 < 9 );
++v14;
}
while ( v14 < 8 );//执行完了后，v40=Y^2（X、Y为分别为前、后8个字节异或后的整数）
LOBYTE(v22) = v37.charPtr;
v23 = 0;
do
{
v24 = (unsigned __int8)v22 + 7 * *((unsigned __int8 *)v40.intArray + v23);
*((_BYTE *)v37.intArray + v23++) = v24;
v22 = (unsigned int)v24 >> 8;
}
while ( v23 < 17 );////执行完了后，v37=7*Y^2（X、Y为分别为前、后8个字节异或后的整数）
v37.charPtr = HIBYTE(v24);
v25 = 0;
v26 = 0;
do
{
v27 = *((unsigned __int8 *)v39.intArray + v26) - *((unsigned __int8 *)v37.intArray + v
*((_BYTE *)v38.intArray + v26) = v27;
if ( v27 < 0 )
v25 = 1;
++v26;
}
while ( v26 < 17 );///执行完了后，v38=X^2 - 7*Y^2（X、Y为分别为前、后8个字节异或后的整数
if ( !v25 )
{
v28 = 0;
while ( *((_BYTE *)v38.intArray + v28) == *((_BYTE *)v30.intArray + v28) )
{
if ( ++v28 >= 17 )
return 1;
}//判断方程X^2 - 7*Y^2 = 8，那么就要求解 X 和 Y
}
}
}
return 0;
}
```

5、求解X^2 - 7*Y^2 = 8，并反推字符串序列号。

```
# -*- coding: UTF-8 -*-
import re
import array

#丢番图方程 X^2 - D * Y^2 = M
#丢番图方程 X^2 - 7 * Y^2 = 8
#通解( 6 + 2 * sqrt(7) )( 8 + 3 * sqrt(7) )^n

#丢番图初始化参数:
A = 6;
B = 2;
C = 8;
D = 3;
#-----------------------------------------------------------------------------------
# 计算丢番图方程参数:关键代码
def getArg(arg):
return [arg[0] * C + arg[1] * D * 7,arg[0] * D +arg[1] * C];
#-----------------------------------------------------------------------------------
# 字符串转十六进制
def str_to_hex(s):
return ' '.join([hex(ord(c)).replace('0x', '') for c in s])
#-----------------------------------------------------------------------------------
def str_to_hexArr(s):
return re.findall(r'.{2}', s).reverse();
#-----------------------------------------------------------------------------------
#十六进制数(字符串) 转换成 十六进制数组(十六进制数)(按小端存放)
def hex_to_hexArr(s):
tmp = ("%s"%(s))[2:len(s)-1]
#print tmp;
if len(tmp)%2==1 :
tmp="0"+tmp;
tmpArr = re.findall(r'.{2}', tmp);
tmpArr.reverse();
#print tmpArr;
argAHexArr = [];
for item in tmpArr:
argAHexArr.append(int(item,16));
return argAHexArr;
#-----------------------------------------------------------------------------------
#输出十六进制数组
def print_hexArr_1D(arr):
hex_array=[];
for item in arr:
hex_array.append('0x%02X'%item);
```

```
return hex_array ;
def print_hexArr_2D(arr):
hex_array=[];
for item in arr:
hex_array.append( print_hexArr_1D(item));
return hex_array;
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
#关键子程序，对找到的解，进行校核判断
def Solution(arg):
print "\n进一步验证是否满足要求";
argHex = [hex(arg[0]),hex(arg[1])];
argHexArr = [hex_to_hexArr(hex(arg[0])),hex_to_hexArr(hex(arg[1]))];
print "Result for argHexArr_Solution:\n %s"%(argHexArr);
print "Result for argHexArr_Solution(Hex):\n %s\n"%(print_hexArr_2D(argHexArr));
#print "-"*80;
if len(argHexArr)!=2 or len(argHexArr[0])<8 or len(argHexArr[1])<8 :
print '不满足要求：长度不够（不然高位将为00）';
print "-"*80;
return;
for item in argHexArr:
for itm in item:
if itm == 0 :
print '不满足要求：不能为00';
print "-"*80;
return;
#异或运算
strHexArr = [];
i=0;
while i<2:
j=0;
tmpHexArr =[];
while j<8:
tmpHexArr.append(argHexArr[i][j] ^ keyArr[i][j]);
j=j+1;
strHexArr.append(tmpHexArr);
i=i+1;
print "Result for strHexArr:\n %s"%(strHexArr);
print "Result for strHexArr(Hex):\n %s"%(print_hexArr_2D(strHexArr));
#验证所有字符：应为可输入字符：0x20-0x7E。
for item in strHexArr:
for itm in item:
if itm < 0x20 or itm > 0x7E :
print '不满足要求：应为可输入字符：0x20-0x7E';
print "-"*80;
```

```
return;
print "找到序列号：";
#sn = str.decode(strHexArr[0]);
#sn='---'.join(strHexArr[0])
sn="";
for item in strHexArr:
for itm in item:
sn=sn+chr(itm);
print sn;
sn_result.append(sn);
print "-"*80;
return;
#------------------------------------------------------------------------------
#------------------------------------------------------------------------------
#------------------------------------------------------------------------------
# 求解 大整数:Main主函数，找到潜在的解，判断解，输出解

#key按小端存放
key0 = 0xE38C9616;
key1 = 0x646E9881;
key2 = 0x81DC0884;
key3 = 0x4F484DBE;
sn_result=[];
key = [hex(key0 ^ (key1<<32)),hex(key2 ^ (key3<<32))];
keyArr = [hex_to_hexArr(key[0]),hex_to_hexArr(key[1])];
print "Result for keyArr:\n %s\n"%(keyArr);
print "Result for keyArr(Hex):\n %s"%(print_hexArr_2D(keyArr));
print "-"*80;
arg = [A,B];
argHex =[0x00,0x00];
while arg[0] < 0x1000000000000000 :#第8个字节应为01至0F，才能满足要求
arg =getArg(arg);
print "Result for Solution(Hex):\n %s"%(print_hexArr_1D(arg));
if arg[0] <0x0100000000000000 :
print "长度不满足要求";
print "-"*80;
continue;
print "长度初步满足要求";
Solution(arg);
print "-"*80;
print "最终找的结果是：%d个"%(len(sn_result));
print sn_result;
#------------------------------------------------------------------------------
#------------------------------------------------------------------------------
```

6、运算结果

```
Result for keyArr:
[[22, 150, 140, 227, 129, 152, 110, 100], [132, 8, 220, 129, 190, 77, 72, 79]]

Result for keyArr(Hex):
[['0x16', '0x96', '0x8C', '0xE3', '0x81', '0x98', '0x6E', '0x64'], ['0x84', '0x
08', '0xDC', '0x81', '0xBE', '0x4D', '0x48', '0x4F']]
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x5A', '0x22']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x59A', '0x21E']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x5946', '0x21BE']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x58EC6', '0x219C2']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x58931A', '0x217A62']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x583A2DA', '0x2158C5E']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x57E19A86', '0x21374B7E']
长度不满足要求
```

```
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x578960586', '0x2115F2B82']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x57317EBDDA', '0x20F4BB6CA2']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x56D9F55D81A', '0x20D3A579E9E']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x5682C3DEC3C6', '0x20B2B0BE7D3E']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x562BE9E966446', '0x2091DD1903542']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x55D5672587809A', '0x20712A6844D6E2']
长度不满足要求
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x557F3B3B9E1A55A', '0x2050988B2BD38DE']
长度初步满足要求

进一步验证是否满足要求
Result for argHexArr_Solution:
[[90, 165, 225, 185, 179, 243, 87, 5], [222, 56, 189, 178, 136, 9, 5, 2]]
Result for argHexArr_Solution(Hex):
[['0x5A', '0xA5', '0xE1', '0xB9', '0xB3', '0xF3', '0x57', '0x05'], ['0xDE', '0x
38', '0xBD', '0xB2', '0x88', '0x09', '0x05', '0x02']]

Result for strHexArr:
[[76, 51, 109, 90, 50, 107, 57, 97], [90, 48, 97, 51, 54, 68, 77, 77]]
```

```
Result for strHexArr(Hex):
[['0x4C', '0x33', '0x6D', '0x5A', '0x32', '0x6B', '0x39', '0x61'], ['0x5A', '0x
30', '0x61', '0x33', '0x36', '0x44', '0x4D', '0x4D']]
找到序列号：
L3mZ2k9aZ0a36DMM
--------------------------------------------------------------------------------

Result for Solution(Hex):
['0x552965D47892D506', '0x20302760C38EB6FE']
长度初步满足要求

进一步验证是否满足要求
Result for argHexArr_Solution:
[[6, 213, 146, 120, 212, 101, 41, 85], [254, 182, 142, 195, 96, 39, 48, 32]]
Result for argHexArr_Solution(Hex):
[['0x06', '0xD5', '0x92', '0x78', '0xD4', '0x65', '0x29', '0x55'], ['0xFE', '0x
B6', '0x8E', '0xC3', '0x60', '0x27', '0x30', '0x20']]

Result for strHexArr:
[[16, 67, 30, 155, 85, 253, 71, 49], [122, 190, 82, 66, 222, 106, 120, 111]]
Result for strHexArr(Hex):
[['0x10', '0x43', '0x1E', '0x9B', '0x55', '0xFD', '0x47', '0x31'], ['0x7A', '0x
BE', '0x52', '0x42', '0xDE', '0x6A', '0x78', '0x6F']]
不满足要求：应为可输入字符：0x20-0x7E
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

最终找的结果是：1个
['L3mZ2k9aZ0a36DMM']

请按任意键继续. . .
```

后记：

关于如何找到 丢番图 X^2 - 7*Y^2 = 8的最小正整数解，我的方法是 用Excel来搞定。因为这个方程不复杂，很快就能找到，如果作者整个复杂的角色，估计就麻烦了。

比如X^2 - 1141 * Y^2 =1，它的基本解Xn+Yn*sqrt(1141)中的Yn=30693385322765657197397208，这就麻烦了。如果好好设计系数D和M，将增强解密的难度。

▼▼▼
▼▼
▼

1、看雪.纽盾 KCTF 2019 Q2 | 第一题点评及解题思路
2、看雪.纽盾 KCTF 2019 Q2 | 第三题点评及解题思路
3、看雪.纽盾 KCTF 2019 Q2 | 第二题点评及解题思路
4、终曲 · 看雪.纽盾 KCTF 2019 Q2 圆满落幕，精彩回顾！

主办方

看雪

看雪学院（www.kanxue.com）是一个专注于PC、移动、智能设备安全研究及逆向工程的开发者社区！创建于2000年，历经19年的发展，受到业内的广泛认同，在行业中树立了令人尊敬的专业形象。平台为会员提供安全知识的在线课程教学，同时为企业提供智能设备安全相关产品和服务。

合作伙伴

NEWDON secures IT

上海纽盾科技股份有限公司（www.newdon.net）成立于2009年，是一家以“网络安全”为主轴，以“科技源自生活，纽盾服务社会”为核心经营理念，以网络安全产品的研发、生产、销售、售后服务与相关安全服务为一体的专业安全公司，致力于为数字化时代背景下的用户提供安全产品、安全服务以及等级保护等安全解决方案。

小手一戳，了解更多

公众号ID：ikanxue

官方微博：看雪安全

商务合作：wsc@kanxue.com

戳原文，查看更多精彩writeup！

阅读原文