2019看雪CTF 晋级赛Q2第四题wp

转载

deyou0823 于 2019-07-22 19:55:00 发布 165 收藏
原文链接：http://www.cnblogs.com/DirWang/p/11228038.html
版权

上次参加2019看雪CTF 晋级赛Q2卡在了这道题上，虽然逆出算法，但是方程不会解，哈哈哈哈，果然数学知识很重要呀，现在记录一下。

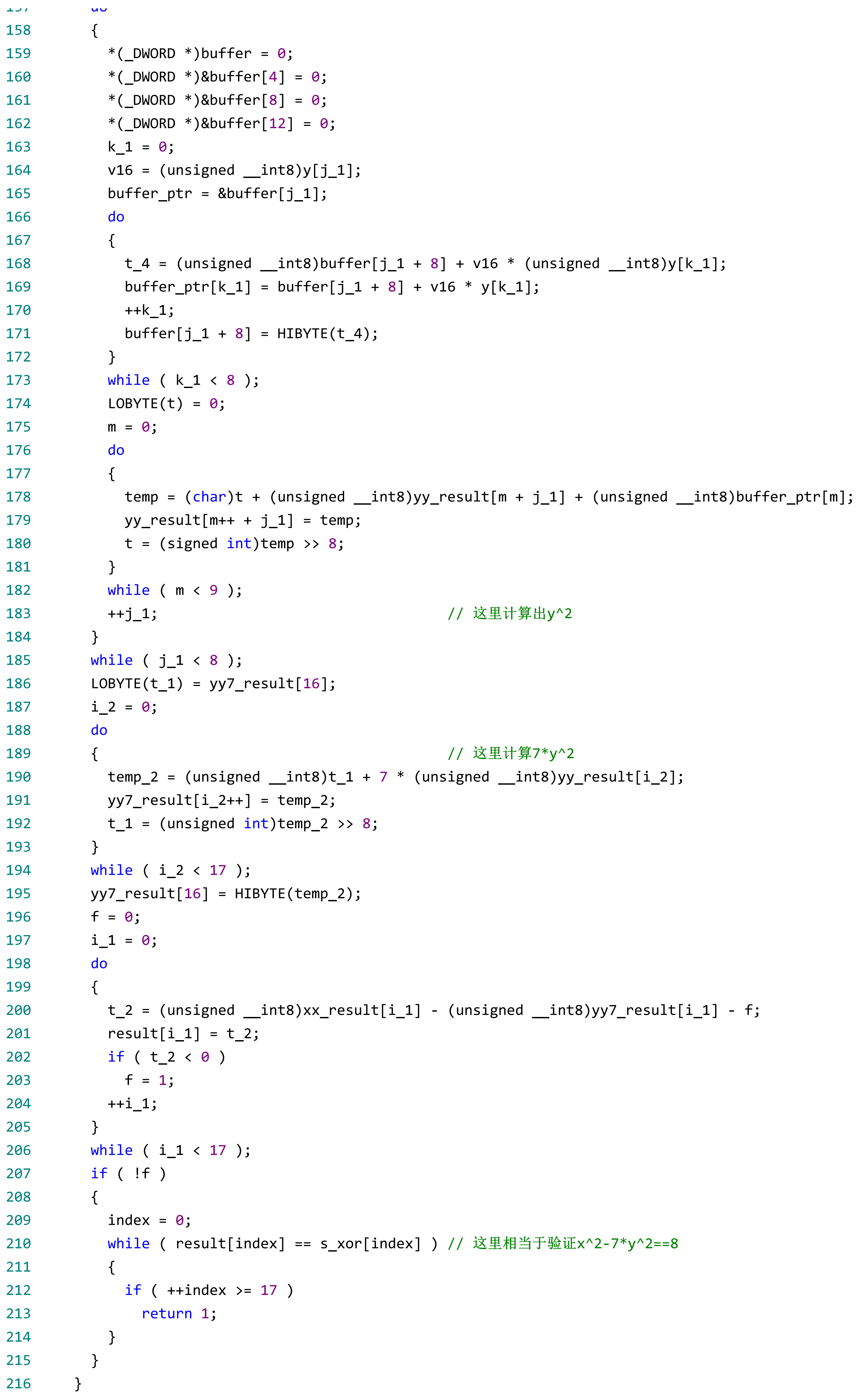
首先根据关键信息，根据错误提示字符串定位到这里：

```
int __thiscall guanjian_401EA0(CWnd *this)
{
  CWnd *v1; // esi
  int index; // eax
  WCHAR String; // [esp+Ch] [ebp-310h]
  char v5; // [esp+Eh] [ebp-30Eh]
  char ptr; // [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;
  String = 0;
  memset(&v5, 0, 0x1FEu);
  ptr = 0;
  memset(&v7, 0, 0xFFu);
  CWnd::GetDlgItemTextW(v1, 1000, &String, 20);
  if ( wcslen(&String) == 16 )
  {
    index = 0;
    while ( !(*(&String + index) & 0xFF00) )
    {
      *(&ptr + index) = *((_BYTE *)&String + 2 * index);
      if ( ++index >= 16 )
      {
        v8 = 64;
        flOldProtect = 0;
        VirtualProtect(sub_9D10E0, 0xD17u, 0x40u, &flOldProtect);// BOOL VirtualProtect(
                                                //   LPVOID lpAddress,
                                                //   DWORD dwSize,
                                                //   DWORD flNewProtect,
                                                //   PDWORD lpflOldProtect
                                                // lpAddress，要改变属性的内存起始地址。
                                                //
                                                // dwSize，要改变属性的内存区域大小。
                                                //
                                                // flNewProtect，内存新的属性类型，设置为PAGE_EXECUTE_READWRITE（0x40）时该内存页为可读可写可执行。
                                                //
                                                // pflOldProtect，内存原始属性类型保存地址。
                                                //
                                                //
        GetLastError();
        qmemcpy(sub_9D10E0, byte_B347B8, 0x330u);
        VirtualProtect(sub_9D10E0, 0xD17u, flOldProtect, &v8);
        if ( !GetLastError() )
        {
          v10 = 0;
          v10 = sub_9D10E0();
          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);
}
```

注意到运行时先修改sub_9D10E0()代码段，然后再运行sub_9D10E0()，我们动态调试，dump出修改完成的sub_9D10E0()代码段

```
int __cdecl sub_9D10E0(char *input)
{
  signed int i; // eax
  char v2; // cl
  signed int v3; // ecx
  signed int v4; // eax
  signed int low_index; // eax
  signed int j; // esi
  signed int k; // ecx
  __int16 v8; // dx
  char *buffer_ptr_; // edi
  __int16 v10; // ax
  signed int temp_1; // eax
  signed int m_1; // ecx
  unsigned __int16 t_3; // bx
  signed int j_1; // esi
  signed int k_1; // ecx
  __int16 v16; // dx
  char *buffer_ptr; // edi
  __int16 t_4; // ax
  signed int t; // eax
  signed int m; // ecx
  unsigned __int16 temp; // bx
  unsigned int t_1; // eax
  signed int i_2; // ecx
  unsigned __int16 temp_2; // dx
  char f; // dl
  signed int i_1; // eax
  __int16 t_2; // si
  int index; // eax
  char s_xor[17]; // [esp+8h] [ebp-90h]
  int v31; // [esp+1Ch] [ebp-7Ch]
  int v32; // [esp+20h] [ebp-78h]
  int v33; // [esp+24h] [ebp-74h]
  char x[8]; // [esp+28h] [ebp-70h]
  char buffer[16]; // [esp+30h] [ebp-68h]
  char y[8]; // [esp+40h] [ebp-58h]
  char yy7_result[17]; // [esp+48h] [ebp-50h]
char result[17]; // [esp+5Ch] [ebp-3Ch]
char xx_result[17]; // [esp+70h] [ebp-28h]
char yy_result[17]; // [esp+84h] [ebp-14h]

s_xor[4] = 0x81u;
s_xor[11] = 0x81u;
i = 0;
s_xor[0] = 0x16;
s_xor[1] = -106;
s_xor[2] = -116;
s_xor[3] = -29;
s_xor[5] = -104;
s_xor[6] = 110;
s_xor[7] = 100;
s_xor[8] = 0x84u;
s_xor[9] = 8;
s_xor[10] = -36;
s_xor[12] = 0xBEu;
s_xor[13] = 77;
s_xor[14] = 72;
s_xor[15] = 79;
*(_DWORD *)&s_xor[16] = 0;
v31 = 0;
v32 = 0;
v33 = 0;
*(_DWORD *)x = 0;
*(_DWORD *)&x[4] = 0;
*(_DWORD *)y = 0;
*(_DWORD *)&y[4] = 0;
do
{                                             // 先将输入分成两部分，记为x,y，然后进行异或操作，
  v2 = s_xor[i + 8] ^ input[i + 8];           // 从输入的第9位开始处理，索引8-15
                                              // 异或
  x[i] = s_xor[i] ^ s_xor[i + input - s_xor]; // 输入的前8位
                                              // 异或
  y[i++] = v2;
}
while ( i < 8 );
*(_DWORD *)s_xor = 0;
*(_DWORD *)xx_result = 0;
*(_DWORD *)&xx_result[4] = 0;
*(_DWORD *)&xx_result[8] = 0;
*(_DWORD *)&xx_result[12] = 0;
xx_result[16] = 0;
*(_DWORD *)yy_result = 0;
*(_DWORD *)&yy_result[4] = 0;
*(_DWORD *)&yy_result[8] = 0;
*(_DWORD *)&yy_result[12] = 0;
yy_result[16] = 0;
*(_DWORD *)yy7_result = 0;
*(_DWORD *)&yy7_result[4] = 0;
*(_DWORD *)&yy7_result[8] = 0;
*(_DWORD *)&yy7_result[12] = 0;
yy7_result[16] = 0;
*(_DWORD *)result = 0;
*(_DWORD *)&result[4] = 0;
*(_DWORD *)&result[8] = 0;
*(_DWORD *)&result[12] = 0;
result[16] = 0;
```

```
*(_DWORD *)&s_xor[4] = 0;
*(_DWORD *)&s_xor[8] = 0;
*(_DWORD *)&s_xor[12] = 0;
s_xor[16] = 0;
v3 = 8;
s_xor[0] = 8;
v4 = 7;
do
{
  if ( x[v4] )                                // x[7]!=0
    break;
  --v3;
  --v4;
}
while ( v4 >= 0 );
if ( v3 == 8 )
{
  low_index = 7;
  do
  {
    if ( y[low_index] )                       // y[7]!=0
      break;
    --v3;
    --low_index;
  }
  while ( low_index >= 0 );                   // 输入为16位
  if ( v3 == 8 && !(x[7] & 0xF0) )            // 第8位<0x10
  {
    j = 0;
    do
    {
      *(_DWORD *)buffer = 0;
      *(_DWORD *)&buffer[4] = 0;
      *(_DWORD *)&buffer[8] = 0;
      *(_DWORD *)&buffer[12] = 0;
      k = 0;
      v8 = (unsigned __int8)x[j];
      buffer_ptr_ = &buffer[j];
      do
      {
        v10 = (unsigned __int8)buffer[j + 8] + v8 * (unsigned __int8)x[k];
        buffer_ptr_[k] = buffer[j + 8] + v8 * x[k];
        ++k;
        buffer[j + 8] = HIBYTE(v10);
      }
      while ( k < 8 );
      LOBYTE(temp_1) = 0;
      m_1 = 0;
      do
      {
        t_3 = (char)temp_1 + (unsigned __int8)xx_result[m_1 + j] + (unsigned __int8)buffer_ptr_[m_1];
        xx_result[m_1++ + j] = t_3;
        temp_1 = (signed int)t_3 >> 8;
      }
      while ( m_1 < 9 );
      ++j;                                    // 先按字节乘，再加进位，和手算乘法原理一致
    }
    while ( j < 8 );                          // 通过两层循环其实是为了计算大数相乘，这里算出x^2
    j_1 = 0;
    do
```



```
  do
  {
    *(_DWORD *)buffer = 0;
    *(_DWORD *)&buffer[4] = 0;
    *(_DWORD *)&buffer[8] = 0;
    *(_DWORD *)&buffer[12] = 0;
    k_1 = 0;
    v16 = (unsigned __int8)y[j_1];
    buffer_ptr = &buffer[j_1];
    do
    {
      t_4 = (unsigned __int8)buffer[j_1 + 8] + v16 * (unsigned __int8)y[k_1];
      buffer_ptr[k_1] = buffer[j_1 + 8] + v16 * y[k_1];
      ++k_1;
      buffer[j_1 + 8] = HIBYTE(t_4);
    }
    while ( k_1 < 8 );
    LOBYTE(t) = 0;
    m = 0;
    do
    {
      temp = (char)t + (unsigned __int8)yy_result[m + j_1] + (unsigned __int8)buffer_ptr[m];
      yy_result[m++ + j_1] = temp;
      t = (signed int)temp >> 8;
    }
    while ( m < 9 );
    ++j_1;                                  // 这里计算出y^2
  }
  while ( j_1 < 8 );
  LOBYTE(t_1) = yy7_result[16];
  i_2 = 0;
  do
  {                                         // 这里计算7*y^2
    temp_2 = (unsigned __int8)t_1 + 7 * (unsigned __int8)yy_result[i_2];
    yy7_result[i_2++] = temp_2;
    t_1 = (unsigned int)temp_2 >> 8;
  }
  while ( i_2 < 17 );
  yy7_result[16] = HIBYTE(temp_2);
  f = 0;
  i_1 = 0;
  do
  {
    t_2 = (unsigned __int8)xx_result[i_1] - (unsigned __int8)yy7_result[i_1] - f;
    result[i_1] = t_2;
    if ( t_2 < 0 )
      f = 1;
    ++i_1;
  }
  while ( i_1 < 17 );
  if ( !f )
  {
    index = 0;
    while ( result[index] == s_xor[index] ) // 这里相当于验证x^2-7*y^2==8
    {
      if ( ++index >= 17 )
        return 1;
    }
  }
}
```

```
  }
  return 0;
}
```

到这里思路已经很明确了，16位输入，分成两部分进行异或操作，验证x^2-7*y^2=8

限定条件为

0x100000000000000<x < 0x1000000000000000,（x为8字节，且第8字节<0x10）

0x100000000000000<y<x （y为8字节，）

在进行一步异或即可得到原输入

x^2-7*y^2=8为非标准佩尔方程，求解使用了wolframalpha

https://www.wolframalpha.com/input/?i=x%5E2-7y%5E2%3D8,72057594037927936%3Cx+%3C+1152921504606846976,72057594037927936%3Cy%3Cx

```
x=385044246406735194
y=145533045678356702

s_xor1=0x646e9881e38c9616
s_xor2=0x4f484dbe81dc0884

t1=hex(x^s_xor1)[2:]
t2=hex(y^s_xor2)[2:]

m1=[]
m2=[]
for i in range(0,16,2):
    m1.append(int(t1[i:i+2],16))
    m2.append(int(t2[i:i+2],16))
a=''.join(map(chr,m1))
b=''.join(map(chr,m2))
print(a[-1::-1]+b[-1::-1])
# L3mZ2k9aZ0a36DMM
```

链接：https://pan.baidu.com/s/1ZpzCus2BdlSujkVRBQZZxQ
提取码：qrlz
复制这段内容后打开百度网盘手机App，操作更方便哦

转载于:https://www.cnblogs.com/DirWang/p/11228038.html