REVERSE-PRACTICE-BUUCTF-15

原创

P1umH0 于 2021-02-26 23:12:12 发布 65 收藏

分类专栏： Reverse-BUUCTF

本文链接：https://blog.csdn.net/weixin_45582916/article/details/114155860

REVERSE-PRACTICE-BUUCTF-15

[2019红帽杯]xx

exe程序，运行后直接输入，无壳，ida分析

交叉引用字符串“You win！”来到sub_1400011A0函数

主要的逻辑为

读取输入，验证输入长度是否为19，且均为小写字母或0~9的数字

对输入进行TEA加密，加密密钥通过调试可知是输入的前4个字符，即“flag”

TEA加密的密文变换位置

变换位置后的密文循环异或

循环异或后的密文与已知字节比较

```
__int64 __fastcall sub_1400011A0(__int64 a1, __int64 a2)
{
  unsigned __int64 input_len_; // rbx
  signed __int64 input_len; // rax
  __int128 *v4; // rax
  __int64 Code; // r11
  __int128 *v6; // r14
  int v7; // edi
  __int128 *v8; // rsi
  char v9; // r10
  int v10; // edx
```

```
__int64 v11; // r8
unsigned __int64 Code_len; // rcx
signed __int64 Code_len_; // rcx
unsigned __int64 v14; // rax
unsigned __int64 i; // rax
_BYTE *cipher; // rax
size_t v17; // rsi
_BYTE *cipher_; // rbx
_BYTE *v19; // r9
signed int v20; // er11
char *v21; // r8
signed __int64 v22; // rcx
char v23; // al
signed __int64 v24; // r9
signed __int64 v25; // rdx
__int64 v26; // rax
size_t Size; // [rsp+20h] [rbp-48h]
__int128 v29; // [rsp+28h] [rbp-40h]
int v30; // [rsp+38h] [rbp-30h]
int v31; // [rsp+3Ch] [rbp-2Ch]
int input[4]; // [rsp+40h] [rbp-28h]
int v33; // [rsp+50h] [rbp-18h]

*(_OWORD *)input = 0i64;
v33 = 0;
sub_1400018C0(std::cin, a2, input);           // 读取输入
input_len_ = -1i64;
input_len = -1i64;
do                                            // do循环计算输入的长度
  ++input_len;
while ( *((_BYTE *)input + input_len) );
if ( input_len != 19 )                        // 验证输入的长度是否为19
{
  sub_140001620(std::cout, "error\n");
  _exit((unsigned __int64)input);
}
v4 = (__int128 *)sub_140001E5C(5ui64);
Code = *(_QWORD *)&::Code;                    // Code=="qwertyuiopasdfghjklzxcvbnm1234567890"
v6 = v4;
v7 = 0;
v8 = v4;                                      // v4==v6==v8
do
{
  v9 = *((_BYTE *)v8 + (char *)input - (char *)v4);
  v10 = 0;
  *(_BYTE *)v8 = v9;
  v11 = 0i64;
  Code_len = -1i64;
  do
    ++Code_len;
  while ( *(_BYTE *)(Code + Code_len) );
  if ( Code_len )
  {
    do
    {
      if ( v9 == *(_BYTE *)(Code + v11) )
        break;
      ++v10;
      ++v11;
```

```
    }
    while ( v10 < Code_len );
  }
  Code_len_ = -1i64;
  do
    ++Code_len_;
  while ( *(_BYTE *)(Code + Code_len_) );
  if ( v10 == Code_len_ )                       // 验证输入的内容是否在Code范围内，即输入的内容均为小写字母或0~9的数字
    _exit(Code);
  v8 = (__int128 *)((char *)v8 + 1);
}
while ( (char *)v8 - (char *)v4 < 4 );
*((_BYTE *)v4 + 4) = 0;
do
  ++input_len_;
while ( *((_BYTE *)input + input_len_) );
v14 = 0i64;
v29 = *v6;
while ( *((_BYTE *)&v29 + v14) )
{
  if ( !*((_BYTE *)&v29 + v14 + 1) )
  {
    ++v14;
    break;
  }
  if ( !*((_BYTE *)&v29 + v14 + 2) )
  {
    v14 += 2i64;
    break;
  }
  if ( !*((_BYTE *)&v29 + v14 + 3) )
  {
    v14 += 3i64;
    break;
  }
  v14 += 4i64;
  if ( v14 >= 0x10 )
    break;
}
for ( i = v14 + 1; i < 16; ++i )
  *((_BYTE *)&v29 + i) = 0;
cipher = sub_140001AB0((__int64)input, input_len_, (unsigned __int8 *)&v29, &Size);// 在sub_140001Ab0函数中发现一常数0x61C88647，判断为TEA加密算法
                                                // v29为密钥，调试可知为输入的前4个字符，即“flag”
v17 = Size;
cipher_ = cipher;
v19 = sub_140001E5C(Size);
v20 = 1;
*v19 = cipher_[2];                              // 输入经TEA加密后的密文cipher，变换位置，存储到v19
v21 = v19 + 1;
v19[1] = *cipher_;
v19[2] = cipher_[3];
v19[3] = cipher_[1];
v19[4] = cipher_[6];
v19[5] = cipher_[4];
v19[6] = cipher_[7];
v19[7] = cipher_[5];
v19[8] = cipher_[10];
v19[9] = cipher_[8];
v19[10] = cipher_[11];
```

```
  v19[11] = cipher_[9];
  v19[12] = cipher_[14];
  v19[13] = cipher_[12];
  v19[14] = cipher_[15];
  v19[15] = cipher_[13];
  v19[16] = cipher_[18];
  v19[17] = cipher_[16];
  v19[18] = cipher_[19];
  v19[19] = cipher_[17];
  v19[20] = cipher_[22];
  v19[21] = cipher_[20];
  v19[22] = cipher_[23];
  for ( v19[23] = cipher_[21]; v20 < v17; ++v21 )// v20初始值为1，v17==Size==24，v21初始值为&(v19[1])，v20和v21是同时加1的
  {
    v22 = 0i64;
    if ( v20 / 3 > 0 )                          // 决定下面异或运算的次数
    {
      v23 = *v21;                               // 取指针v21指向的值
      do
      {
        v23 ^= v19[v22++];                      // 实际上为 *v21^=v19[v22++]，而v21是指向v19的，意味着当v20大于等于3时，v19[v20]从v19[0]异或到v19[v20/3-1]
                                                // 即v19会变化，不再是简单的TEA加密的密文
        *v21 = v23;
      }
      while ( v22 < v20 / 3 );
    }
    ++v20;
  }
  *(_QWORD *)&v29 = 0xC0953A7C6B40BCCEi64;      // v29是最后要去比较的结果，提取时注意小端序
  v24 = v19 - (_BYTE *)&v29;
  *((_QWORD *)&v29 + 1) = 0x3502F79120209BEFi64;
  v25 = 0i64;
  v30 = 0xC8021823;
  v31 = 0xFA5656E7;
  do
  {
    if ( *((_BYTE *)&v29 + v25) != *((_BYTE *)&v29 + v25 + v24) )// v19和v29比较验证
      _exit(v7 * v7);
    ++v7;
    ++v25;
  }
  while ( v25 < 24 );
  v26 = sub_140001620(std::cout, "You win!");
  std::basic_ostream<char,std::char_traits<char>>::operator<<(v26, sub_1400017F0);
  return 0i64;
}
```

先写逆循环异或和逆位置变换得到TEA加密的密文脚本

```
v29=[0xce, 0xbc, 0x40, 0x6b, 0x7c, 0x3a, 0x95, 0xc0,
     0xef, 0x9b, 0x20, 0x20, 0x91, 0xf7, 0x02, 0x35,
     0x23, 0x18, 0x02, 0xc8, 0xe7, 0x56, 0x56, 0xfa]
v19=v29
for v20 in range(len(v19)-1,-1,-1):
    tmp=v20//3
    if tmp>0:
        for i in range(tmp):
            v19[v20]^=v19[i]
cipher=[0]*len(v19)
cipher_index=[2, 0, 3, 1, 6, 4, 7, 5, 10, 8, 11, 9, 14, 12, 15, 13, 18, 16, 19, 17, 22, 20, 23, 21]
for i in range(len(cipher)):
    cipher[cipher_index[i]]=v19[i]
for i in range(len(cipher)):
    print(hex(cipher[i])),
```

test1

```
D:\python27-x64\python2.exe D:/Python/pycharm/pycfile/test1.py
0xbc 0xa5 0xce 0x40 0xf4 0xb2 0xb2 0xe7 0xa9 0x12 0x9d 0x12 0xae 0x10 0xc8 0x5b 0x3d 0xd7 0x6 0x1d 0xdc 0x70 0xf8 0xdc
```

网上找的xxtea的python代码，参考：XXTEA 加解密 as3 和 Python 分别实现

密钥为“flag”需要加长到16，写解xxtea脚本即可得到flag

```
# encoding: utf-8
import struct

_DELTA = 0x9E3779B9

def _long2str(v, w):
    n = (len(v) - 1) << 2
    if w:
        m = v[-1]
        if (m < n - 3) or (m > n): return ''
        n = m
    s = struct.pack('<%iL' % len(v), *v)
    return s[0:n] if w else s

def _str2long(s, w):
    n = len(s)
    m = (4 - (n & 3) & 3) + n
    s = s.ljust(m, "\0")
    v = list(struct.unpack('<%iL' % (m >> 2), s))
    if w: v.append(n)
    return v

def encrypt(str, key):
    if str == '': return str
    v = _str2long(str, True)
    k = _str2long(key.ljust(16, "\0"), False)
    n = len(v) - 1
    z = v[n]
    y = v[0]
    sum = 0
    q = 6 + 52 // (n + 1)
    while q > 0:
        sum = (sum + _DELTA) & 0xffffffff
        e = sum >> 2 & 3
        for p in range(n):
```

```
            for p in range(n):
                y = v[p + 1]
                v[p] = (v[p] + ((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4) ^ (sum ^ y) + (k[p & 3 ^ e] ^ z))) & 0xffffffff
                z = v[p]
            y = v[0]
            v[n] = (v[n] + ((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4) ^ (sum ^ y) + (k[n & 3 ^ e] ^ z))) & 0xffffffff
            z = v[n]
            q -= 1
        return _long2str(v, False)

def decrypt(str, key):
    if str == '': return str
    v = _str2long(str, False)
    k = _str2long(key.ljust(16, "\0"), False)
    n = len(v) - 1
    z = v[n]
    y = v[0]
    q = 6 + 52 // (n + 1)
    sum = (q * _DELTA) & 0xffffffff
    while (sum != 0):
        e = sum >> 2 & 3
        for p in range(n, 0, -1):
            z = v[p - 1]
            v[p] = (v[p] - ((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4) ^ (sum ^ y) + (k[p & 3 ^ e] ^ z))) & 0xffffffff
            y = v[p]
        z = v[n]
        v[0] = (v[0] - ((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4) ^ (sum ^ y) + (k[0 & 3 ^ e] ^ z))) & 0xffffffff
        y = v[0]
        sum = (sum - _DELTA) & 0xffffffff
    return _long2str(v, True)

def xor(x ,y):
    return ord(x) ^ ord(y)

key="flag"+'\x00'*12
cipher_data=['\xbc','\xa5','\xce','\x40','\xf4','\xb2','\xb2','\xe7','\xa9','\x12','\x9d','\x12','\xae','\x10','
\xc8','\x5b','\x3d','\xd7','\x06','\x1d','\xdc','\x70','\xf8','\xdc']
cipher=''.join(cipher_data)
print(decrypt(cipher,key))
#flag{CXX_and_++tea}
```

[ACTF新生赛2020]Universe_final_answer

elf文件，无壳，ida分析

main函数，读取输入，check函数验证输入的字符，输入验证成功，程序调用sub_C50，最后输出flag

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  __int64 v4; // [rsp+0h] [rbp-A8h]
  char input; // [rsp+20h] [rbp-88h]
  unsigned __int64 v6; // [rsp+88h] [rbp-20h]

  v6 = __readfsqword(0x28u);
  __printf_chk(1LL, "Please give me the key string:", a3);
  scanf("%s", &input);
  if ( check(&input) )                          // 验证输入input的字符
  {
    sub_C50((__int64)&input, &input, &v4);      // 输入验证成功，程序调用sub_C50
    __printf_chk(1LL, "Judgement pass! flag is actf{%s_%s}\n", &input);
  }
  else
  {
    puts("False key!");
  }
  return 0LL;
}
```


重要的是check函数，分析可知，验证input的字符，就是验证方程组是否成立

```
  v1 = input[1];
  v2 = *input;
  v3 = input[2];
  v4 = input[3];
  v5 = input[4];
  v6 = input[6];
  v7 = input[5];
  v8 = input[7];
  v9 = input[8];
  result = 0;
  if ( -85 * v9 + 58 * v8 + 97 * v6 + v7 + -45 * v5 + 84 * v4 + 95 * v2 - 20 * v1 + 12 * v3 == 12613 )
  {
    v11 = input[9];
    if ( 30 * v11 + -70 * v9 + -122 * v6 + -81 * v7 + -66 * v5 + -115 * v4 + -41 * v3 + -86 * v1 - 15 * v2 - 30 * v8 == -54400
      && -103 * v11 + 120 * v8 + 108 * v7 + 48 * v4 + -89 * v3 + 78 * v1 - 41 * v2 + 31 * v5 - (v6 << 6) - 120 * v9 == -10283
      && 71 * v6 + (v7 << 7) + 99 * v5 + -111 * v3 + 85 * v1 + 79 * v2 - 30 * v4 - 119 * v8 + 48 * v9 - 16 * v11 == 22855
      && 5 * v11 + 23 * v9 + 122 * v8 + -19 * v6 + 99 * v7 + -117 * v5 + -69 * v3 + 22 * v1 - 98 * v2 + 10 * v4 == -2944
      && -54 * v11 + -23 * v8 + -82 * v3 + -85 * v2 + 124 * v1 - 11 * v4 - 8 * v5 - 60 * v7 + 95 * v6 + 100 * v9 == -2222
      && -83 * v11 + -111 * v7 + -57 * v2 + 41 * v1 + 73 * v3 - 18 * v4 + 26 * v5 + 16 * v6 + 77 * v8 - 63 * v9 == -13258
      && 81 * v11 + -48 * v9 + 66 * v8 + -104 * v6 + -121 * v7 + 95 * v5 + 85 * v4 + 60 * v3 + -85 * v2 + 80 * v1 == -1559
      && 101 * v11 + -85 * v9 + 7 * v6 + 117 * v7 + -83 * v5 + -101 * v4 + 90 * v3 + -28 * v1 + 18 * v2 - v8 == 6308 )
    {
      result = 99 * v11 + -28 * v9 + 5 * v8 + 93 * v6 + -18 * v7 + -127 * v5 + 6 * v4 + -9 * v3 + -93 * v1 + 58 * v2 == -1697;
    }
  }
  return result;
}
```


利用python的z3库写脚本得到input

```
from z3 import *
flag=[Int('flag[%d]' %i ) for i in range(10)]
s=Solver()
s.add(-85 * flag[8] + 58 * flag[7] + 97 * flag[6] + flag[5] + -45 * flag[4] + 84 * flag[3] + 95 * flag[0] - 20 *
 flag[1] + 12 * flag[2] == 12613)
s.add(30 * flag[9] + -70 * flag[8] + -122 * flag[6] + -81 * flag[5] + -66 * flag[4] + -115 * flag[3] + -41 * fla
g[2] + -86 * flag[1] - 15 * flag[0] - 30 * flag[7] == -54400)
s.add(-103 * flag[9] + 120 * flag[7] + 108 * flag[5] + 48 * flag[3] + -89 * flag[2] + 78 * flag[1] - 41 * flag[0
] + 31 * flag[4] - (flag[6]*64) - 120 * flag[8] == -10283)
s.add(71 * flag[6] + (flag[5] *128) + 99 * flag[4] + -111 * flag[2] + 85 * flag[1] + 79 * flag[0] - 30 * flag[3]
 - 119 * flag[7] + 48 * flag[8] - 16 * flag[9] == 22855)
s.add(5 * flag[9] + 23 * flag[8] + 122 * flag[7] + -19 * flag[6] + 99 * flag[5] + -117 * flag[4] + -69 * flag[2]
 + 22 * flag[1] - 98 * flag[0] + 10 * flag[3] == -2944)
s.add(-54 * flag[9] + -23 * flag[7] + -82 * flag[2] + -85 * flag[0] + 124 * flag[1] - 11 * flag[3] - 8 * flag[4]
 - 60 * flag[5] + 95 * flag[6] + 100 * flag[8] == -2222)
s.add(-83 * flag[9] + -111 * flag[5] + -57 * flag[0] + 41 * flag[1] + 73 * flag[2] - 18 * flag[3] + 26 * flag[4]
 + 16 * flag[6] + 77 * flag[7] - 63 * flag[8] == -13258)
s.add(81 * flag[9] + -48 * flag[8] + 66 * flag[7] + -104 * flag[6] + -121 * flag[5] + 95 * flag[4] + 85 * flag[3
] + 60 * flag[2] + -85 * flag[0] + 80 * flag[1] == -1559)
s.add(101 * flag[9] + -85 * flag[8] + 7 * flag[6] + 117 * flag[5] + -83 * flag[4] + -101 * flag[3] + 90 * flag[2
] + -28 * flag[1] + 18 * flag[0] - flag[7] == 6308 )
s.add(99 * flag[9] + -28 * flag[8] + 5 * flag[7] + 93 * flag[6] + -18 * flag[5] + -127 * flag[4] + 6 * flag[3] +
 -9 * flag[2] + -93 * flag[1] + 58 * flag[0] == -1697)
if s.check()==sat:
    print(s.model())
```

运行结果

```
D:\python27-x64\python2.exe D:/Python/pycharm/pycfile/test1.py
[flag[8] = 119,
 flag[3] = 82,
 flag[1] = 48,
 flag[0] = 70,
 flag[4] = 84,
 flag[2] = 117,
 flag[9] = 64,
 flag[5] = 121,
 flag[6] = 95,
 flag[7] = 55]
```


转成字符串，运行elf文件，输入，得到flag

```
zihaoli@zihaoliworld: ~/CTF
文件(F)  编辑(E)  查看(V)  搜索(S)  终端(T)  帮助(H)
zihaoli@zihaoliworld:~/CTF$ ./UniverseFinalAnswer
Please give me the key string:F0uRTy_7w@
Judgement pass! flag is actf{F0uRTy_7w@_42}
zihaoli@zihaoliworld:~/CTF$
```

[WUSTCTF2020]level4

elf文件，运行后输出了几段字符串，可知是二叉树的遍历，无壳，ida分析

main函数，程序输出的type1为中序遍历，type2为后序遍历，type3被注释掉，应该是先序遍历

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  puts("Practice my Data Structure code.....");
  puts("Typing....Struct.....char....*left....*right.............emmmmm...OK!");
  init();
  puts("Traversal!");
  printf("Traversal type 1:", argv);
  type1((char *)&unk_601290);                   // 中序遍历
  printf("\nTraversal type 2:");
  type2((char *)&unk_601290);                   // 后序遍历
  printf("\nTraversal type 3:");
  puts("    //type3(&x[22]);   No way!");      // 先序遍历
  puts(&byte_400A37);
  return 0;
}
```


中序遍历：2f0t02T{hcsiI_SwA__r7Ee}

后序遍历：20f0Th{2tsIS_icArE}e7__w

由中序遍历和后序遍历的字符串确定二叉树，再先序遍历二叉树，即为flag

代码参考：已知后序序遍历序列和中序遍历序列建立二叉树

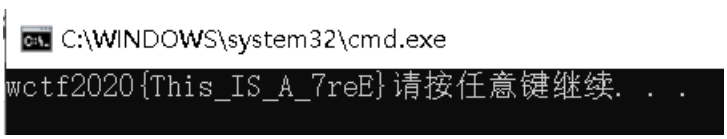
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef char ElementType;
typedef struct BiTNode {
 ElementType data;
 struct BiTNode* lchild;
 struct BiTNode* rchild;
}BiTNode, *BiTree;

BiTree CreatBinTree(char* post, char* in, int n);
void preorder(BiTree T);

int main()
{
 BiTree T;
 char postlist[100] = "20f0Th{2tsIS_icArE}e7__w";
 char inlist[100] = "2f0t02T{hcsiI_SwA__r7Ee}";
 int length;
 length = strlen(postlist);
 T = CreatBinTree(postlist, inlist, length);
 preorder(T);
 return 0;
}
void  preorder(BiTree T)
{
 if (T)
 {
  printf("%c", T->data);
  preorder(T->lchild);
  preorder(T->rchild);

 }
}
BiTree CreatBinTree(char* post, char* in, int n)
{
 BiTree T;
 int i;
 if (n <= 0) return NULL;
 T = (BiTree)malloc(sizeof(BiTNode));
 T->data = post[n - 1];
 for (i = 0; in[i] != post[n - 1]; i++);
 T->lchild = CreatBinTree(post, in, i);
 T->rchild = CreatBinTree(post + i, in + i + 1, n - 1 - i);
 return T;
}
```

运行结果即为flag



findKey

exe程序，无壳，ida分析

交叉引用字符串“flag{}”，发现从地址0x00401640开始，代码就没有被ida识别，应该是加了花指令

在地址0x00401918和地址0x0040191D处，两条一样的指令，放在了一起，nop掉第二条push指令

```
.text:0040190F                  push    eax
.text:00401910                  call    _memcpy
.text:00401915                  add     esp, 0Ch
.text:00401918                  push    offset byte_428C54
.text:0040191D
.text:0040191D loc_40191D:                              ; CODE XREF: .text:0040193D↓j
.text:0040191D                  push    offset byte_428C54
.text:00401922                  call    _strlen
.text:00401927                  add     esp, 4
.text:0040192A                  push    eax
.text:0040192B                  push    offset byte_428C54
```


在地址0x00401640处右键->create function，或者选中全部的红色地址代码，按p创建函数，F5反汇编

主要的逻辑为

v20的字符串和v16的字符串循环异或，结果和pbData的md5散列值比较

比较相等时，md5散列前的pbData和unk_423030循环异或，结果即为flag

```
LRESULT __stdcall sub_401640(HWND hWndParent, UINT Msg, WPARAM wParam, LPARAM lParam)
{
  int v5; // eax
  size_t v6; // eax
  DWORD v7; // eax
  int v8; // eax
  int v9; // eax
  const char *v10; // [esp-4h] [ebp-450h]
  CHAR *v11; // [esp+0h] [ebp-44Ch]
  int v12; // [esp+4h] [ebp-448h]
  int v13; // [esp+4Ch] [ebp-400h]
  UINT v14; // [esp+50h] [ebp-3FCh]
  CHAR v15; // [esp+54h] [ebp-3F8h]
  CHAR v16[2]; // [esp+154h] [ebp-2F8h]
  int v17; // [esp+157h] [ebp-2F5h]
  __int16 v18; // [esp+15Bh] [ebp-2F1h]
  char v19; // [esp+15Dh] [ebp-2EFh]
  char v20; // [esp+160h] [ebp-2ECh]
  char v21; // [esp+181h] [ebp-2CBh]
  __int16 v22; // [esp+25Dh] [ebp-1EFh]
  char v23; // [esp+25Fh] [ebp-1EDh]
  CHAR v24; // [esp+260h] [ebp-1ECh]
  CHAR String[4]; // [esp+360h] [ebp-ECh]
  int v26; // [esp+364h] [ebp-E8h]
  __int16 v27; // [esp+368h] [ebp-E4h]
  CHAR Text; // [esp+36Ch] [ebp-E0h]
  struct tagRECT Rect; // [esp+38Ch] [ebp-C0h]
  CHAR Buffer; // [esp+39Ch] [ebp-B0h]
  HDC hdc; // [esp+400h] [ebp-4Ch]
  struct tagPAINTSTRUCT Paint; // [esp+404h] [ebp-48h]
  WPARAM v33; // [esp+444h] [ebp-8h]
  int v34; // [esp+448h] [ebp-4h]

  LoadStringA(hInstance, 0x6Au, &Buffer, 100);
  v14 = Msg;
  if ( Msg > 0x111 )
  {
    if ( v14 == 517 )
```

```
  {
    if ( strlen((const char *)&pbData) > 6 )
      ExitProcess(0);
    if ( strlen((const char *)&pbData) )
    {
      memset(&v24, 0, 0x100u);
      v6 = strlen((const char *)&pbData);
      memcpy(&v24, &pbData, v6);              // pbData拷贝到v24
      v10 = (const char *)&pbData;
      do
      {
        v7 = strlen(v10);
        sub_40101E(&pbData, v7, v11);         // CryptCreateHash函数，第二个参数为0x8003u，对pbData进行md5散列
      }
      while ( &v12 && !&v12 );
      strcpy(&v20, "0kk`d1a`55k222k2a776jbfgd`06cjjb");// v20="0kk`d1a`55k222k2a776jbfgd`06cjjb"
      memset(&v21, 0, 0xDCu);
      v22 = 0;
      v23 = 0;
      strcpy(v16, "SS");                      // v16="SS"
      v17 = 0;
      v18 = 0;
      v19 = 0;
      v8 = strlen(&v20);
      sub_401005(v16, (int)&v20, v8);         // v20^=v16
      if ( _strcmpi((const char *)&pbData, &v20) )// v20和md5散列后的pbData比较，求出v20，解md5散列，可得到散列前的pbData
      {
        SetWindowTextA(hWndParent, "flag{}");
        MessageBoxA(hWndParent, "Are you kidding me?", "^_^", 0);
        ExitProcess(0);
      }
      memcpy(&v15, &unk_423030, 0x32u);       // unk_423030已知，拷贝到v15
      v9 = strlen(&v15);
      sub_401005(&v24, (int)&v15, v9);        // v24和md5散列前的pbData相同，v24^=v15
      MessageBoxA(hWndParent, &v15, 0, 0x32u);
    }
    ++dword_428D54;
  }
  else
  {
    if ( v14 != 520 )
      return DefWindowProcA(hWndParent, Msg, wParam, lParam);
    if ( dword_428D54 == 16 )
    {
      strcpy(String, "ctf");
      v26 = 0;
      v27 = 0;
      SetWindowTextA(hWndParent, String);
      strcpy(&Text, "Are you kidding me?");
      MessageBoxA(hWndParent, &Text, &Buffer, 0);
    }
    ++dword_428D54;
  }
}
else
{
  switch ( v14 )
  {
    case 0x111u:
```

```
    case 0x111u:
      v34 = (unsigned __int16)wParam;
      v33 = wParam >> 16;
      v13 = (unsigned __int16)wParam;
      if ( (unsigned __int16)wParam == 104 )
      {
        DialogBoxParamA(hInstance, (LPCSTR)0x67, hWndParent, (DLGPROC)DialogFunc, 0);
      }
      else
      {
        if ( v13 != 105 )
          return DefWindowProcA(hWndParent, Msg, wParam, lParam);
        DestroyWindow(hWndParent);
      }
      break;
    case 2u:
      PostQuitMessage(0);
      break;
    case 0xFu:
      hdc = BeginPaint(hWndParent, &Paint);
      GetClientRect(hWndParent, &Rect);
      v5 = strlen(&Buffer);
      DrawTextA(hdc, &Buffer, v5, &Rect, 1u);
      EndPaint(hWndParent, &Paint);
      break;
    default:
      return DefWindowProcA(hWndParent, Msg, wParam, lParam);
  }
 }
 return 0;
}
```

先写脚本得到pbData的md5散列值，通过在线网站求逆

```
v20="0kk`d1a`55k222k2a776jbfgd`06cjjb"
v16="SS"
pbData=""
for i in range(len(v20)):
    pbData+=chr(ord(v20[i])^ord(v16[i%len(v16)]))
print(pbData)
```

```
test1
D:\python27-x64\python2.exe D:/Python/pycharm/pycfile/
c8837b23ff8aaa8a2dde915473ce0991
```


再写md5散列前的pbData（v24）和unk_unk_423030（v15）循环异或的脚本，得到flag

```
v24="123321"
v15=[0x57, 0x5E, 0x52, 0x54, 0x49, 0x5F, 0x01, 0x6D, 0x69, 0x46,
  0x02, 0x6E, 0x5F, 0x02, 0x6C, 0x57, 0x5B, 0x54, 0x4C]
flag=""
for i in range(len(v15)):
    flag+=chr(ord(v24[i%len(v24)])^v15[i])
print(flag)
```

```
test2
D:\python27-x64\python2.exe D:/Python/pycharm/pycfile/test2.py
flag{n0_Zu0_n0_die}
```