reverse方向入门过程

原创

月阴荒 于 2020-08-07 09:14:35 发布 888 收藏 12

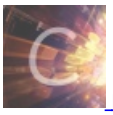
分类专栏： CTF re

本文链接：https://blog.csdn.net/weixin_43847969/article/details/107831619

CTF 同时被 2 个专栏收录

22 篇文章 1 订阅

订阅专栏



re

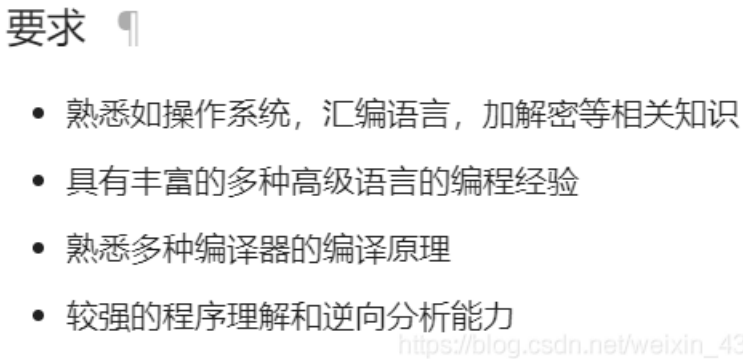
1 篇文章 0 订阅

订阅专栏

声明：因为懒，所以懒得从头开始做，基本都是看着wp复现的，其中大部分的wp都来源于

https://blog.csdn.net/palmer9/category_9607326.html

话不多说，直接从ctf-wiki上和做题开始



我们这里可以看到要求

熟悉操作系统（靠学校教的操作系统）

汇编语言（我看了王爽那本然后自己上网学了一点）

加解密（密码学）

具有丰富的多种高级语言编程经验（。。。。。多丰富？）

较强的程序理解和逆向分析能力（太主观了）

带着上述的基础，我们来进行逆向的学习

常规逆向流程

1. 使用 `strings/file/binwalk/IDA` 等静态分析工具收集信息，并根据这些静态信息进行 `google/github` 搜索
2. 研究程序的保护方法，如代码混淆，保护壳及反调试等技术，并设法破除或绕过保护
3. 反汇编目标软件，快速定位到关键代码进行分析
4. 结合动态调试，验证自己的初期猜想，在分析的过程中理清程序功能
5. 针对程序功能，写出对应脚本，求解出 flag

1.使用各种工具（对我来说主要是IDA）进行静态分析，收集信息
2.研究程序的保护方法，比如代码混淆，保护壳以及反调试等技术，并设法破除或者绕过保护。
3.反汇编目标软件，能够快速定位到关键代码进行分析
4.结合动态调试，验证自己的初期猜想，在分析的过程中理清程序功能
5.针对程序功能，写出对应脚本，求解出flag

代码混淆

比如使用`OLLVM`，`movfuscator`，`花指令`，`虚拟化`及`SMC`等工具技术对代码进行混淆，使得程序分析十分困难。

那么对应的也有反混淆技术，最主要的目的就是复原控制流。比如`模拟执行`和`符号执行`

保护壳

保护壳类型有许多，简单的压缩壳可以归类为如下几种

- unpack -> execute

  直接将程序代码全部解压到内存中再继续执行程序代码

- unpack -> execute -> unpack -> execute ...

  解压部分代码，再边解压边执行

- unpack -> [decoder | encoded code] -> decode -> execute

  程序代码有过编码，在解压后再运行函数将真正的程序代码解码执

对于脱壳也有相关的方法，比如`单步调试法`，`ESP定律`等等

反调试

反调试意在通过检测调试器等方法避免程序被调试分析。比如使用一些 API 函数如`IsDebuggerPresent`检测调试器，使用`SEH异常处理`，时间差检测等方法。也可以通过覆写调试端口、自调试等方法进行保护。

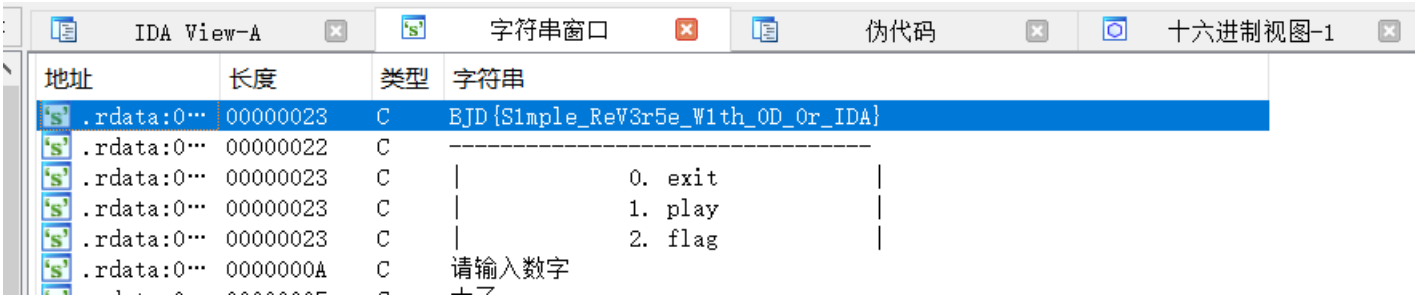
然后直接开始做题

BUU部分

最简单的题目（直接搜索字符串就能看到flag的）

1.[BJDCTF 2nd]guessgame

打开IDA，F12搜索字符串就能看到



xor

一个64位的文件，直接F5看main函数

```
memset(v6, 0, 0x100uLL);
v3 = (char *)256;
printf("Input your flag:\n", 0LL);
get_line(v6, 256LL);
if ( strlen(v6) != 33 )
  goto LABEL_12;
for ( i = 1; i < 33; ++i )
  v6[i] ^= v6[i - 1];
v3 = global;
if ( !strncmp(v6, global, 0x21uLL) )
  printf("Success", v3);
else
LABEL_12:
  printf("Failed", v3);
result = __stack_chk_guard;
if ( __stack_chk_guard == v7 )
  result = 0;
return result;
}
```


其中重要的语句为

if (strlen(v6) != 33) //判断v6的长度是否为33（从上面的程序中可以看出，v6就是我们的输入）

for (i = 1; i < 33; ++i) //进行循环异或，从v6的第二位开始将v6的每一位与前一位异或

v6[i] ^= v6[i - 1];

if (!strncmp(v6, global, 0x21uLL)) //比较v6与global段处存放的前33位（也就是0x21）是否相同，是如果相同的话输出success，

大致地看完代码之后我们发现进入global段里面写了什么很重要，我们要做的就是将异或后的v6与global段中的数据进行比对，比对完之后相同就能拿到flag（这里的sucess应该是提示你通过了这道题，并且前面有提示input your flag，也就是要我们输入一串字符（就是v6变量）做为答案。）那么我们只要进入global看到的global的内容就是异或后的v6，也就是异或后的flag

双击进入global看到内容。

```
0000100000F6E                 assume cs:__cstring
0000100000F6E                 ;org 100000F6Eh
0000100000F6E aFKWOXZUPFVMDGH db 'f',0Ah              ; DATA XREF: __data:_global↓o
0000100000F6E                 db 'k',0Ch,'w&O.@',11h,'x',0Dh,'Z;U',11h,'p',19h,'F',1Fh,'v"M#D',0Eh,'g'
0000100000F6E                 db 6,'h',0Fh,'G2O',0
0000100000F90 aInputYourFlag  db 'Input your flag:',0Ah,0
0000100000F90                                         ; DATA XREF: _main+B↑o
0000100000FA2 ; char aSuccess[]
```

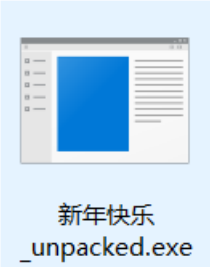
```python
str1 = ['f', 0x0A, 'k', 0x0C, 'w', '&', 'O', '.', '@', 0x11, 'x', 0x0D, 'Z', ';', 'U', 0x11, 'p', 0x19, 'F', 0x1F, 'v',
        '"', 'M', '#', 'D', 0x0E, 'g', 6, 'h', 0x0F, 'G', '2', 'O']

x = 'f'

for i in range(1, len(str1)):
    if (isinstance(str1[i], str)):
        if (isinstance(str1[i - 1], str)):
            x += chr(ord(str1[i]) ^ ord(str1[i - 1]))
        else:
            x += chr(ord(str1[i]) ^ str1[i - 1])
    else:
        x += chr(str1[i] ^ ord(str1[i - 1]))

print(x)
```

新年快乐（最简单的脱壳）

发现放进IDA里有错误报告，应该是加壳了

先用PE查壳（步骤省略）

直接用万能工具查壳，然后点击脱壳就好了



然后得到一个这样的文件，然后就可以放进IDA了，然后看main函数，里面的程序还是相当简单的

IDA View-A　伪代码　十六进制视图-1

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    int result; // eax
4    char v4; // [esp+12h] [ebp-3Ah]
5    __int16 v5; // [esp+20h] [ebp-2Ch]
6    __int16 v6; // [esp+22h] [ebp-2Ah]
7
8    sub_401910();
9    strcpy(&v4, "HappyNewYear!");
10   v5 = 0;
11   memset(&v6, 0, 0x1Eu);
12   printf("please input the true flag:");
13   scanf("%s", &v5);
14   if ( !strncmp((const char *)&v5, &v4, strlen(&v4)) )
15     result = puts("this is true flag!");
16   else
17     result = puts("wrong!");
18   return result;
19 }
```


输入一个v5，与v4相等即可拿到flag（v4是HappyNewYear!，在上面可以看到）
v5来源于上面的scanf，也就是我们得到输入

```
scanf( "%s" , &v5);
if ( !strncmp((const char *)&v5, &v4, strlen(&v4)) )
  result = puts("this is true flag!");
else
```

emmmmm，其实也就是HappyNewYear!做为flag
所以flag是flag{HappyNewYear!}

SimpleRev

点进去看main函数，发现没什么关键的，下面代码的大致意思就是输入的不为d或者D就退出这次循环
和如果输入的为Q或者q就退出，然后看到有一个Decry函数，点进去

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  char v4; // [rsp+Fh] [rbp-1h]

  while ( 1 )
  {
    while ( 1 )
    {
      printf("Welcome to CTF game!\nPlease input d/D to start or input q/Q to quit this program: ", argv, envp);
      v4 = getchar();
      if ( v4 != 100 && v4 != 68 )
        break;
      Decry();
    }
    if ( v4 == 113 || v4 == 81 )
      Exit();
    puts("Input fault format!");
    v3 = getchar();
    putchar(v3);
  }
}
```


点进Decry函数

```
  v11 = 0;
  text = (char *)join(key3, &v9);
  strcpy(key, key1);
  strcat(key, src);
  v2 = 0;
  v3 = 0;
  getchar();
  v5 = strlen(key);
  for ( i = 0; i < v5; ++i )
  {
    if ( key[v3 % v5] > 64 && key[v3 % v5] <= 90 )
      key[i] = key[v3 % v5] + 32;
    ++v3;
  }
  printf("Please input your flag:", src);
  while ( 1 )
  {
    v1 = getchar();
    if ( v1 == 10 )
      break;
    if ( v1 == 32 )
    {
      ++v2;
    }
    else
    {
      if ( v1 <= 96 || v1 > 122 )
```

```
45      }
46      else
47      {
48        if ( v1 <= 96 || v1 > 122 )
49        {
50          if ( v1 > 64 && v1 <= 90 )
51            str2[v2] = (v1 - 39 - key[v3++ % v5] + 97) % 26 + 97;
52        }
53        else
54        {
55          str2[v2] = (v1 - 39 - key[v3++ % v5] + 97) % 26 + 97;
56        }
57        if ( !(v3 % v5) )
58          putchar(32);
59        ++v2;
60      }
61    }
62    if ( !strcmp(text, str2) )
63      puts("Congratulation!\n");
64    else
65      puts("Try again!\n");
66    return __readfsqword(0x28u) ^ v12;
67  }
```


其中v9注意按R转换成字符串格式

```
v9 = 'wodah';
```

同时发现有一个join函数，内容如下，大致意思为，

```
1  char *__fastcall join(const char *a1, const char *a2)
2  {
3    size_t v2; // rbx
4    size_t v3; // rax
5    char *dest; // [rsp+18h] [rbp-18h]
6
7    v2 = strlen(a1);
8    v3 = strlen(a2);
9    dest = (char *)malloc(v2 + v3 + 1);
10   if ( !dest )
11     exit(1);
12   strcpy(dest, a1);
13   strcat(dest, a2);
14   return dest;
15 }
```

大致意思为将a1的长度赋值给v2
将a2的长度赋值给v3，然后malloc动态分配空间dest给a1和a2 (大小为a1+a2+1)
然后将a1赋值给dest(strcpy函数)，将a2拼接到dest后（strcat函数）（。。。。其实整个函数就是将a1和a2，也就是下面的key3和v9连接起来的意思）

```
000202030                public key3
000202030 key3           db 'kills',0
000202036                db    0
```

```
.data:0000000000202010                public key1
.data:0000000000202010 ; char key1[]
.data:0000000000202010 key1           db 'ADSFK',0
.data:0000000000202016                db    0
```

```
unsigned __int64 Decry()
{
  char v1; // [rsp+Fh] [rbp-51h]
  int v2; // [rsp+10h] [rbp-50h]
  int v3; // [rsp+14h] [rbp-4Ch]
  int i; // [rsp+18h] [rbp-48h]
  int v5; // [rsp+1Ch] [rbp-44h]
  char src[8]; // [rsp+20h] [rbp-40h]
  __int64 v7; // [rsp+28h] [rbp-38h]
  int v8; // [rsp+30h] [rbp-30h]
  __int64 v9; // [rsp+40h] [rbp-20h]
  __int64 v10; // [rsp+48h] [rbp-18h]
  int v11; // [rsp+50h] [rbp-10h]
  unsigned __int64 v12; // [rsp+58h] [rbp-8h]

  v12 = __readfsqword(0x28u);
  *(_QWORD *)src = 357761762382LL;      //（按R变成SLCDN）
  v7 = 0LL;
  v8 = 0;
  v9 = 512969957736LL;
  v10 = 0LL;
  v11 = 0;
  text = (char *)join(key3, &v9);
               //令text等于key3+v9
               //key3="kills"（双击点进key3可知）
               //v9="hadow"  //因为小端序存储
               //则 text= killshadow
  strcpy(key, key1);        //将key1赋值给key ， key = "ADSFK"(同样双击可知)
  strcat(key, src);         //将src处的字符拼接到key后
               //key = "ADSFKNDCLS"（小端序）
  v2 = 0;
  v3 = 0;
  getchar();
  v5 = strlen(key);         // v5 = key的长度   v5 = 10
  for ( i = 0; i < v5; ++i )
  {
    if ( key[v3 % v5] > 64 && key[v3 % v5] <= 90 )   //如果key中存在大写字母，将其变成小写（asc码加32）（这里的v5是10，对10取余等于个位数，而v3最大只到9，所以可以忽略v5）
      key[i] = key[v3 % v5] + 32;       //key = "adsfkndcls"
    ++v3;
  }
  printf("Please input your flag:", src);
  while ( 1 )
```

```
  {
    v1 = getchar();         //接受用户的输入赋值给v1
    if ( v1 == 10 )         // 如果输入的为换行符，则退出（都可以按R转换看到是\n，以下省略省略）
      break;
    if ( v1 == 32 )            // 如果输入的为空格，则v2加一
    {
      ++v2;
    }
    else
    {
      if ( v1 <= 96 || v1 > 122 )          // 如果输入的v1不为小写字母
      {
        if ( v1 > 64 && v1 <= 90 )          // 如果v1为大写字母
          str2[v2] = (v1 - 39 - key[v3++ % v5] + 97) % 26 + 97;   // 将v1减去39再减去key中下表为[v3++]位置的字母然后加上97之后求除以26的余数再加上97，然后赋值给str2
                          // str1[v2] = (v1-key[v3]+58)%26 + 97
                         // 变换后str1[v2]存放小写字母
      }
      else
      {
        str2[v2] = (v1 - 39 - key[v3++ % v5] + 97) % 26 + 97;   //如果v1为小写，则同样处理
      }
      if ( !(v3 % v5) )              //如果循环到key的最后一位
        putchar(32);             //打印一个空格
      ++v2;
    }
  }
  if ( !strcmp(text, str2) )         // 如果text和str2存储的相同，则成功
    puts("Congratulation!\n");            // text = "killshadow"
  else
    puts("Try again!\n");
  return __readfsqword(0x28u) ^ v12;
}
```

所以我们的目标就是像str2中存入killshadow，其中要满足输入的每个字符减去39再减去key中下标为[v3++]位置的字母然后加上97对26取余然后再加上97，变为killshadow
根据这个要求，我们来编写脚本，首先赋值出一个代表killshadow的变量和一个代表adsfkndcls的变量，然后从大写字母（因为我们的只有我们的大写字母在经过一系列运算之后加上97还能得到的答案是字母的,虽然在字母表中把小写字母写上也没事）中找到一个字母减去39再减去key中下标为v3的字母的asc值再加上97后对26取余再加上97会得到killshadow
（其中97就是a）

```
key = "adsfkndcls"          //赋值给key
text = "killshadow"            //赋值给text
flag = ""
_dict = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"     //做字母表
v5 = len(text)            //赋值给v5 text的长度
for i in range(v5):
    for v1 in _dict:
        if ord(text[i]) == (ord(v1) - 39 - ord(key[i % v5]) + 97) % 26 + 97:     //例：
            flag += v1
print(flag)
```

//例：将text中的第一个k取出并且转换成asc值107，然后判断是否等于大写字母表中的一个值减去39和字母a的asc码值（也就是97），然后加上97，再对26取余，再加上97那么就是flag，我们时候诸葛亮的知道第一个正确字母是K，也就是75，减去39再减去97再加上97，得36，对26取余得到10，然后再加上97=107（后续答案同样如此得来）
答案KLDQCUDFZO
//ord，返回其asc码值

[ACTF新生赛2020]easyre

事先脱壳(UPX脱壳，用之前的万能脱壳工具即可)

```
sub_401A10();
v4 = '*';
v5 = 'F';
v6 = '\'';
v7 = '"';
v8 = 'N';
v9 = ',';
v10 = '"';
v11 = '(';
v12 = 'I';
v13 = '?';
v14 = '+';
v15 = '@';
printf("Please input:");
scanf("%s", &v19);
if ( (_BYTE)v19 != 'A' || HIBYTE(v19) != 'C' || v20 != 'T' || v21 != 'F' || v22 != '{' || v26 != '}' )
  return 0;
v16 = v23;
v17 = v24;
v18 = v25;
for ( i = 0; i <= 11; ++i )
{
  if ( *(&v4 + i) != byte_402000[*((char *)&v16 + i) - 1] )
    return 0;
}
printf("You are correct!");
return 0;
}
```


前面都是一些定义变量的东西，然后到scanf，输入变量给v19，并且v19如果不等于ACTF{}这些字符，那么return 0
然后定义v16 17 18(?意义不明)
然后解题的重点就在这个循环，那就是判断*（&v4+i）是否等于byte_402000这个数组中下标为的&v16+i-1的数

```
UPX0:00402000 ; char byte_402000[]
UPX0:00402000 byte_402000     db 7Eh                  ; DATA XREF: _main+EC↑r
UPX0:00402001 aZyxwvutsrqponm db '}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>='
UPX0:00402001                 db '<;:9876543210/.-,+*)(',27h,'&%$# !"',0
UPX0:00402060                 align 40h
UPX0:00402080 dword_402080    dd 0FFFFFFFFh           ; DATA XREF: sub_401000+4A↑r
```

```
for ( i = 0; i <= 11; ++i )
```

我们点进去看到了byte_402000有这些，并且在上面的
循环中我们可以知道flag大概是12位

并且*（&v4+i），意为每次v4的地址加1，也就是v4,v5，v6…这样子，直到v15然后判断是否等于byte-402000
所以如果写脚本的话我们要这样写
先将byte-402000中的值赋值给一个变量，然后将v4到v15的值赋值给另外一个变量，
然后将v4中的值用chr转换成字符，然后检索在byte_402000中的位置，然后把这个位置转换成字符添加到flag变量里

这里要介绍两个函数方法，一个叫做append，作用是将对象添加到末尾
例如
aList = [123, ‘xyz’, ‘zara’, ‘abc’];
aList.append(2009);
print "Updated List : ", aList;
结果将会输出Updated List : [123, ‘xyz’, ‘zara’, ‘abc’, 2009]，2009被添加到了列表的末尾。
一个叫做find ，作用是检索一个字符串在指定字符串中的位置
例如
str1 = “this is string example…wow!!!”;
str2 = “exam”;

print str1.find(str2);
将会输出15，意为exam这个str2在 str1中的第15个位置开始

其实下面这张图里的这句话这样看，我们会顺眼很多
v4[i]!=byte_402000[flag[i]-1]

```
if ( *(&v4 + i) != byte_402000[*((char *)&v16 + i) - 1] )
```

```
# -*- coding:utf-8 -*-

v4 = [42,70,39,34,78,44,34,40,73,63,43,64]

model = "}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)(" + chr(0x27) + r
'&%$# !"'

pos = []

for i in v4:
    pos.append(model.find(chr(i))+1)      //意为将v4中的数字转换成字符+1然后检索在model中的位置，并且把这个位置+1，然后把这个位置（一个数字）添加到pos后面
s = [chr(x + 1) for x in pos]        //然后将pos中的每个数字+1转换成字符，然后一个个组成flag
flag = ''.join(s)
print ('flag{'+flag+'}')
```

举个例子大概就是这样，先从v4中取出第一个字符42，然后将转换成字符，也就是*（星号），然后，然后检索*在model中的位置，是83，加1是84，再加1是85，再转换成字符就是85转换成U

*********为什么要两次加1？
答：第一个chr(i)+1是因为位置加1才是下标（下标从0开始）
第二个chr(x+1)是因为原式子中byte_402000[&v16+i]-1,所以要加1补回去

攻防世界部分

insanity（直接搜索字符串就能得到flag）

Mysterious

```
int __stdcall sub_401090(HWND hWnd, int a2, int a3, int a4)
{
  char v5; // [esp+50h] [ebp-310h]
  CHAR Text[4]; // [esp+154h] [ebp-20Ch]
  char v7; // [esp+159h] [ebp-207h]
  __int16 v8; // [esp+255h] [ebp-10Bh]
  char v9; // [esp+257h] [ebp-109h]
  int v10; // [esp+258h] [ebp-108h]
  CHAR String; // [esp+25Ch] [ebp-104h]
  char v12; // [esp+25Fh] [ebp-101h]
  char v13; // [esp+260h] [ebp-100h]
  char v14; // [esp+261h] [ebp-FFh]

  memset(&String, 0, 0x104u);
  v10 = 0;
  if ( a2 == 16 )
  {
    DestroyWindow(hWnd);
    PostQuitMessage(0);
  }
  else if ( a2 == 273 )
  {
    if ( a3 == 1000 )
    {
      GetDlgItemTextA(hWnd, 1002, &String, 260);
      strlen(&String);
      if ( strlen(&String) > 6 )
        ExitProcess(0);
      v10 = atoi(&String) + 1;
      if ( v10 == 123 && v12 == 120 && v14 == 122 && v13 == 121 )
      {
        strcpy(Text, "flag");
        memset(&v7, 0, 0xFCu);
        v8 = 0;
        v9 = 0;
        _itoa(v10, &v5, 10);
```



```
memset(&String, 0, 0x104u);
v10 = 0;
if ( a2 == 16 )
{
  DestroyWindow(hWnd);
  PostQuitMessage(0);
}
else if ( a2 == 273 )
{
  if ( a3 == 1000 )
  {
    GetDlgItemTextA(hWnd, 1002, &String, 260);
    strlen(&String);
    if ( strlen(&String) > 6 )
      ExitProcess(0);
    v10 = atoi(&String) + 1;
    if ( v10 == 123 && v12 == 120 && v14 == 122 && v13 == 121 )
    {
      strcpy(Text, "flag");
      memset(&v7, 0, 0xFCu);
      v8 = 0;
      v9 = 0;
      _itoa(v10, &v5, 10);
      strcat(Text, "{");
      strcat(Text, &v5);
      strcat(Text, "_");
      strcat(Text, "Buff3r_0v3rf|0w");
      strcat(Text, "}");
      MessageBoxA(0, Text, "well done", 0);
    }
    SetTimer(hWnd, 1u, 0x3E8u, TimerFunc);
  }
  if ( a3 == 1001 )
    KillTimer(hWnd, 1u);
```


满足v10=123,v12=120,v14=122,v13=121