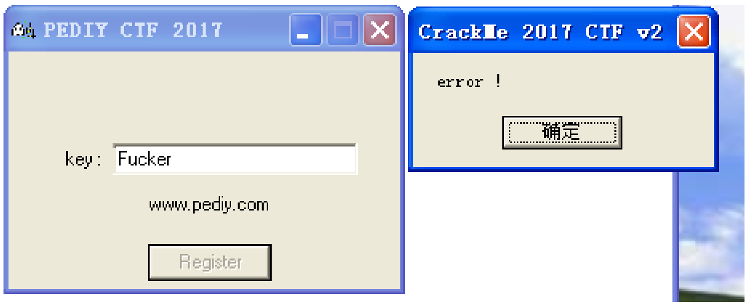
看雪WiFi万能钥匙CTF-第一题 WannaLOL

转载

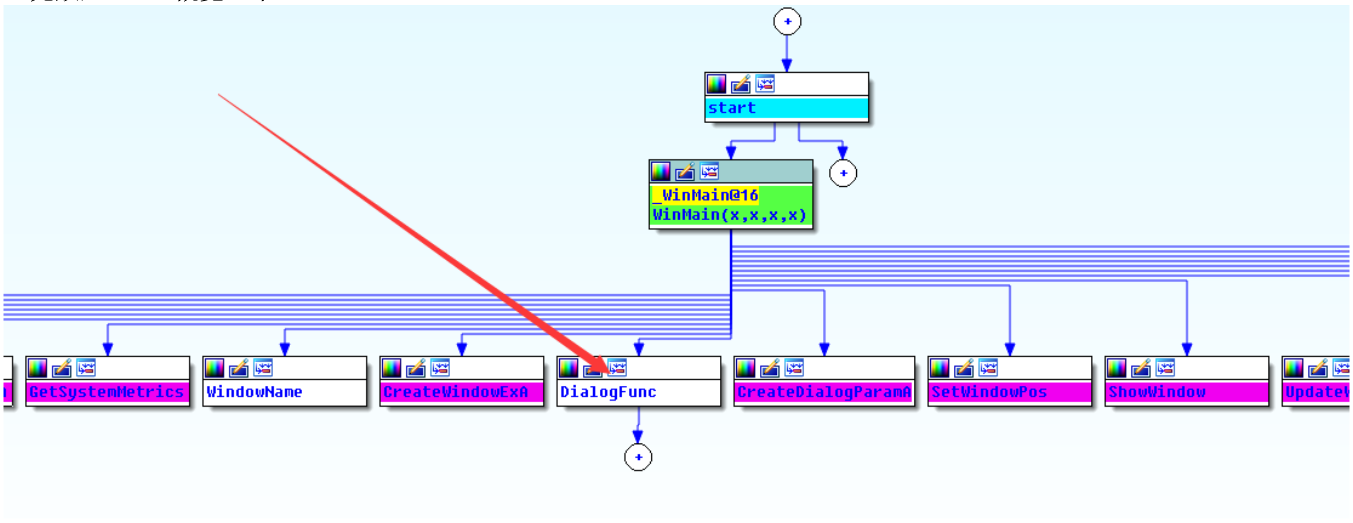
weixin_33951761 于 2017-12-27 22:22:00 发布 90 收藏
文章标签： python
原文链接：https://my.oschina.net/u/3281747/blog/1595845
版权

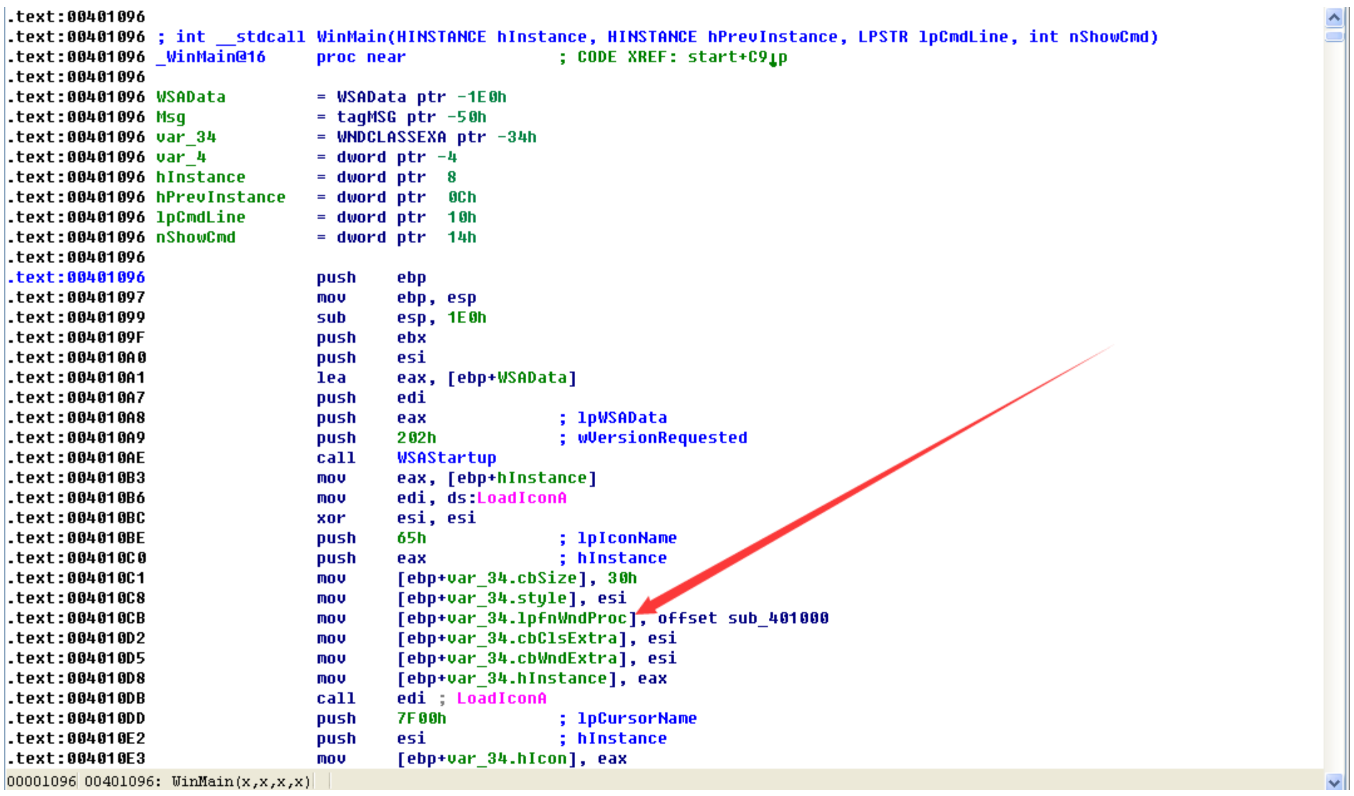
简单刷完韩国人的《逆向工程核心原理》之后，觉得自己需要投入到Crack-Me CTF中的淬炼当中，所以准备把看雪2017年中CFT的题目拿来锻炼一下，话不多说，先看看第一题WananLOL的面目吧。



PS：注意要点，它的标题以及消息框中的文字，还有Register按钮的短暂禁用。

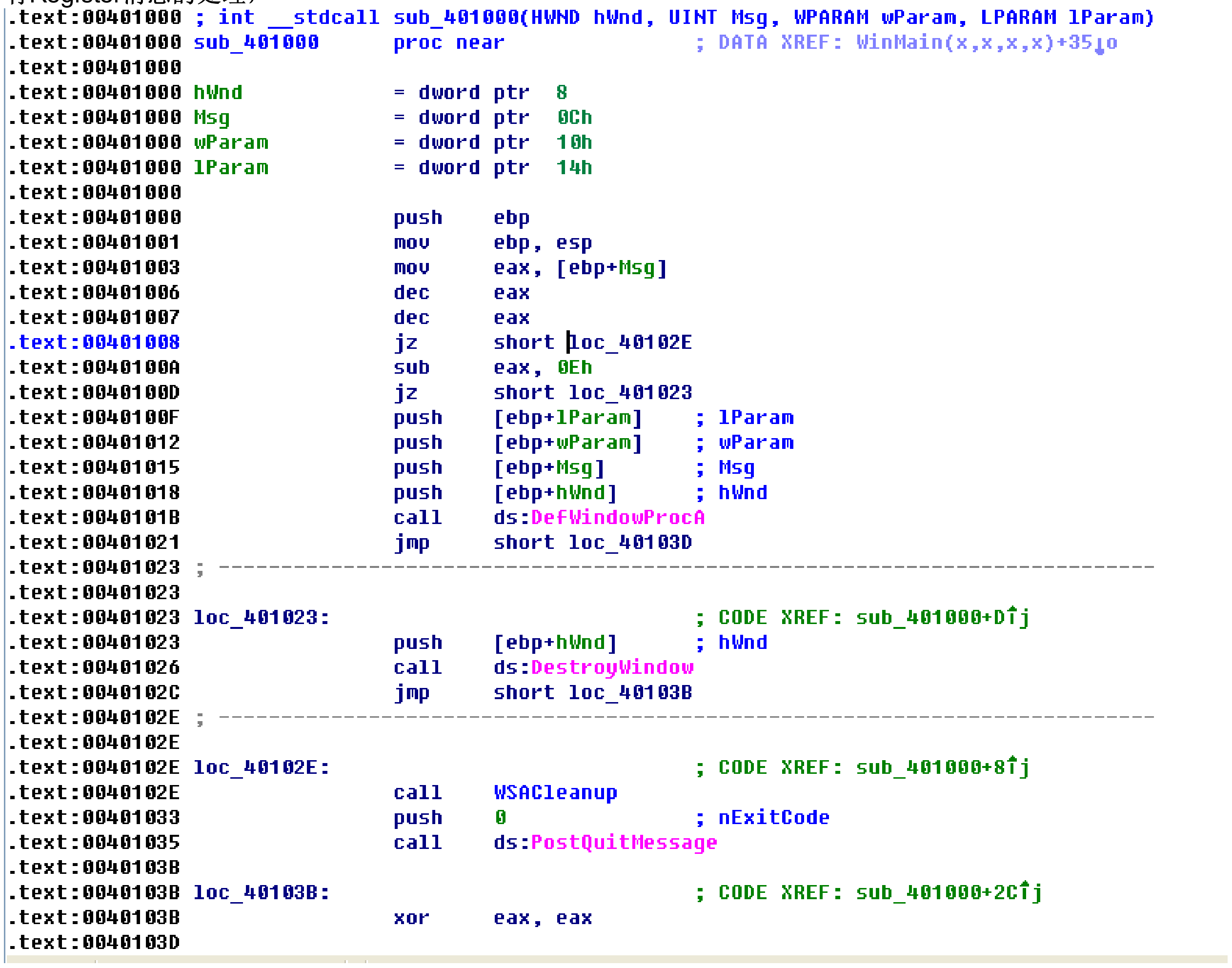
1. 先放入IDA，概览一下





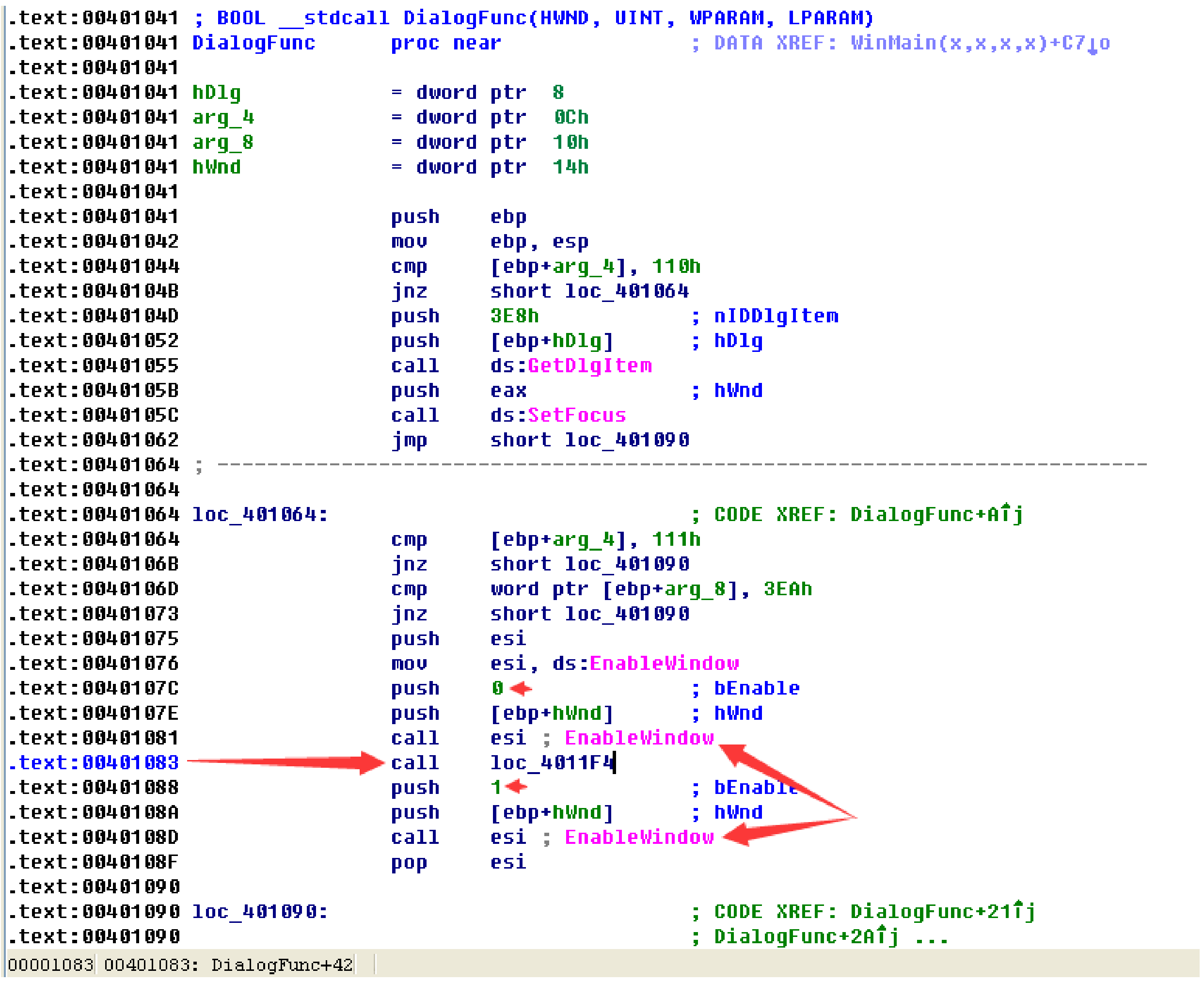
```
.text:00401096
.text:00401096 ; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
.text:00401096 _WinMain@16     proc near               ; CODE XREF: start+C9↓p
.text:00401096
.text:00401096 WSAData         = WSAData ptr -1E0h
.text:00401096 Msg             = tagMSG ptr -50h
.text:00401096 var_34          = WNDCLASSEXA ptr -34h
.text:00401096 var_4           = dword ptr -4
.text:00401096 hInstance       = dword ptr  8
.text:00401096 hPrevInstance   = dword ptr  0Ch
.text:00401096 lpCmdLine       = dword ptr  10h
.text:00401096 nShowCmd        = dword ptr  14h
.text:00401096
.text:00401096                 push    ebp
.text:00401097                 mov     ebp, esp
.text:00401099                 sub     esp, 1E0h
.text:0040109F                 push    ebx
.text:004010A0                 push    esi
.text:004010A1                 lea     eax, [ebp+WSAData]
.text:004010A7                 push    edi
.text:004010A8                 push    eax             ; lpWSAData
.text:004010A9                 push    202h            ; wVersionRequested
.text:004010AE                 call    WSAStartup
.text:004010B3                 mov     eax, [ebp+hInstance]
.text:004010B6                 mov     edi, ds:LoadIconA
.text:004010BC                 xor     esi, esi
.text:004010BE                 push    65h             ; lpIconName
.text:004010C0                 push    eax             ; hInstance
.text:004010C1                 mov     [ebp+var_34.cbSize], 30h
.text:004010C8                 mov     [ebp+var_34.style], esi
.text:004010CB                 mov     [ebp+var_34.lpfnWndProc], offset sub_401000
.text:004010D2                 mov     [ebp+var_34.cbClsExtra], esi
.text:004010D5                 mov     [ebp+var_34.cbWndExtra], esi
.text:004010D8                 mov     [ebp+var_34.hInstance], eax
.text:004010DB                 call    edi ; LoadIconA
.text:004010DD                 push    7F00h           ; lpCursorName
.text:004010E2                 push    esi             ; hInstance
.text:004010E3                 mov     [ebp+var_34.hIcon], eax
00001096 00401096: WinMain(x,x,x,x)
```

其中有两处显示窗口的回调函数（WindowProc和DialogProc），在sub_401000（WindowProc）处没有Register消息的处理，



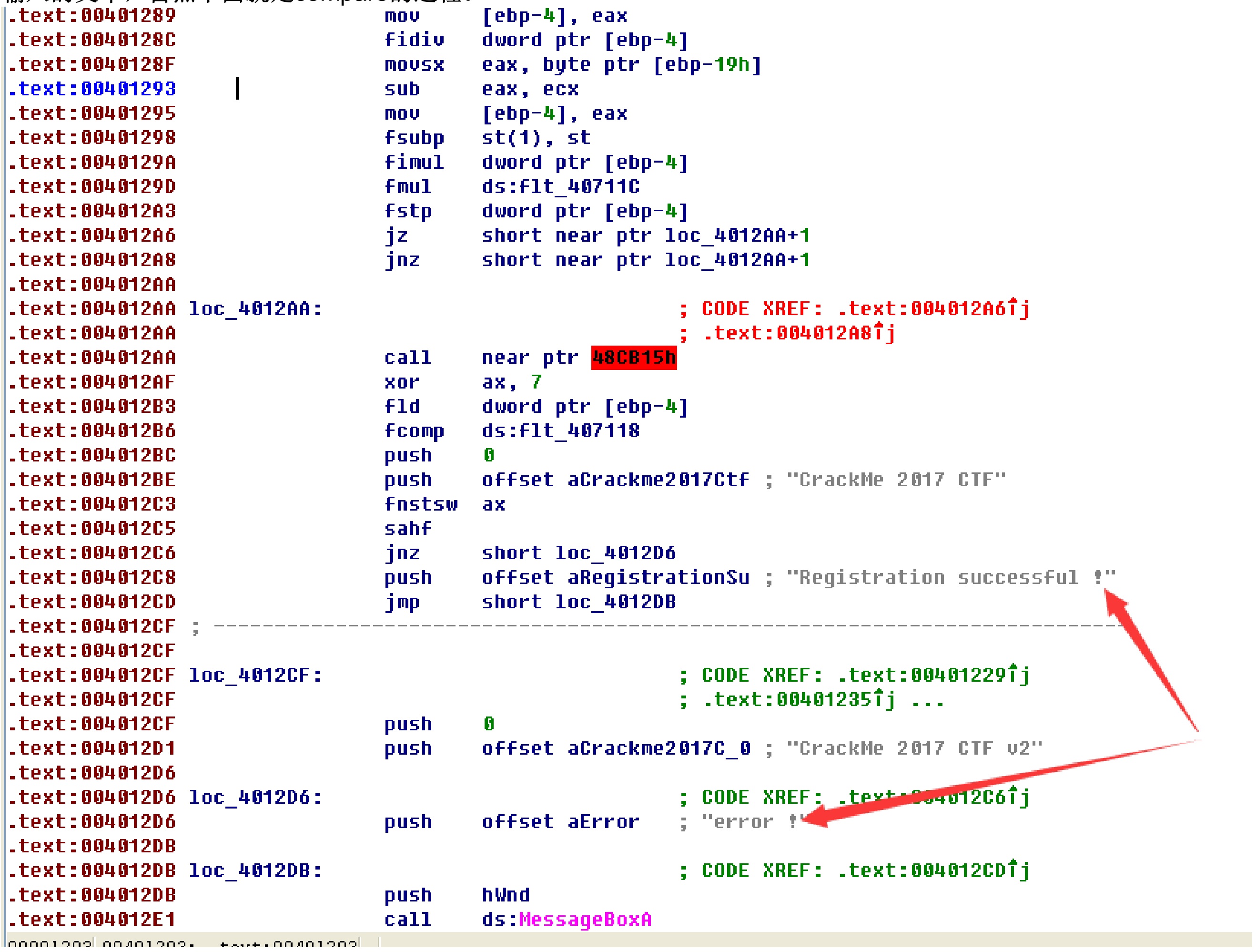
```
.text:00401000 ; int __stdcall sub_401000(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam)
.text:00401000 sub_401000      proc near               ; DATA XREF: WinMain(x,x,x,x)+35↓o
.text:00401000
.text:00401000 hWnd            = dword ptr  8
.text:00401000 Msg             = dword ptr  0Ch
.text:00401000 wParam          = dword ptr  10h
.text:00401000 lParam          = dword ptr  14h
.text:00401000
.text:00401000                 push    ebp
.text:00401001                 mov     ebp, esp
.text:00401003                 mov     eax, [ebp+Msg]
.text:00401006                 dec     eax
.text:00401007                 dec     eax
.text:00401008                 jz      short loc_40102E
.text:0040100A                 sub     eax, 0Eh
.text:0040100D                 jz      short loc_401023
.text:0040100F                 push    [ebp+lParam]    ; lParam
.text:00401012                 push    [ebp+wParam]    ; wParam
.text:00401015                 push    [ebp+Msg]       ; Msg
.text:00401018                 push    [ebp+hWnd]      ; hWnd
.text:0040101B                 call    ds:DefWindowProcA
.text:00401021                 jmp     short loc_40103D
.text:00401023 ; ---------------------------------------------------------------------------
.text:00401023
.text:00401023 loc_401023:                             ; CODE XREF: sub_401000+D↑j
.text:00401023                 push    [ebp+hWnd]      ; hWnd
.text:00401026                 call    ds:DestroyWindow
.text:0040102C                 jmp     short loc_40103B
.text:0040102E ; ---------------------------------------------------------------------------
.text:0040102E
.text:0040102E loc_40102E:                             ; CODE XREF: sub_401000+8↑j
.text:0040102E                 call    WSACleanup
.text:00401033                 push    0               ; nExitCode
.text:00401035                 call    ds:PostQuitMessage
.text:0040103B
.text:0040103B loc_40103B:                             ; CODE XREF: sub_401000+2C↑j
.text:0040103B                 xor     eax, eax
.text:0040103D
```

而在 DialogFunc处有EnableWindow的痕迹，明显能够看出来，EnableWindow的前后两次调用，分别是Register按钮的禁用和启用，那么有关key的compare函数也就在他们之间。



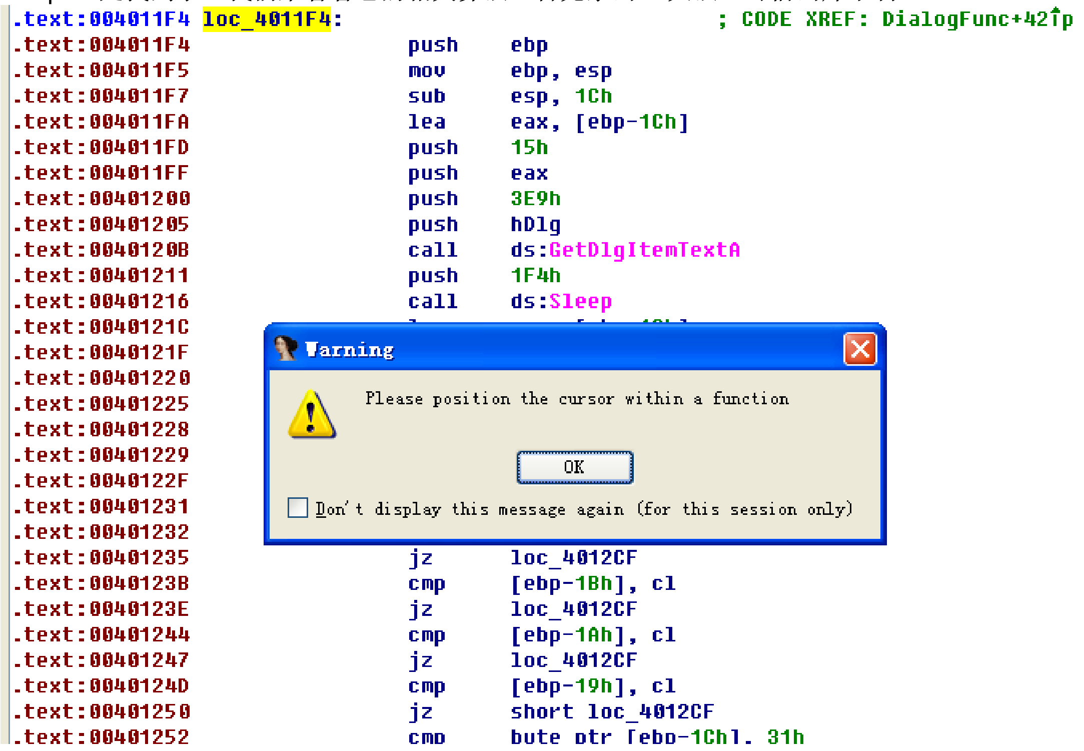
```
.text:00401041 ; BOOL __stdcall DialogFunc(HWND, UINT, WPARAM, LPARAM)
.text:00401041 DialogFunc      proc near               ; DATA XREF: WinMain(x,x,x,x)+C7↓o
.text:00401041
.text:00401041 hDlg            = dword ptr  8
.text:00401041 arg_4           = dword ptr  0Ch
.text:00401041 arg_8           = dword ptr  10h
.text:00401041 hWnd            = dword ptr  14h
.text:00401041
.text:00401041                 push    ebp
.text:00401042                 mov     ebp, esp
.text:00401044                 cmp     [ebp+arg_4], 110h
.text:0040104B                 jnz     short loc_401064
.text:0040104D                 push    3E8h            ; nIDDlgItem
.text:00401052                 push    [ebp+hDlg]      ; hDlg
.text:00401055                 call    ds:GetDlgItem
.text:0040105B                 push    eax             ; hWnd
.text:0040105C                 call    ds:SetFocus
.text:00401062                 jmp     short loc_401090
.text:00401064 ; ---------------------------------------------------------------------------
.text:00401064
.text:00401064 loc_401064:                             ; CODE XREF: DialogFunc+A↑j
.text:00401064                 cmp     [ebp+arg_4], 111h
.text:0040106B                 jnz     short loc_401090
.text:0040106D                 cmp     word ptr [ebp+arg_8], 3EAh
.text:00401073                 jnz     short loc_401090
.text:00401075                 push    esi
.text:00401076                 mov     esi, ds:EnableWindow
.text:0040107C                 push    0               ; bEnable
.text:0040107E                 push    [ebp+hWnd]      ; hWnd
.text:00401081                 call    esi ; EnableWindow
.text:00401083                 call    loc_4011F4
.text:00401088                 push    1               ; bEnable
.text:0040108A                 push    [ebp+hWnd]      ; hWnd
.text:0040108D                 call    esi ; EnableWindow
.text:0040108F                 pop     esi
.text:00401090
.text:00401090 loc_401090:                             ; CODE XREF: DialogFunc+21↑j
.text:00401090                                         ; DialogFunc+2A↑j ...
00001083 00401083: DialogFunc+42
```

点进loc_4011F4处，果然看见MessageBox的调用，在loc_4011F4处的靠前部分还有GetDlgItemText，用于获得输入的文本，自然下面就是compare的过程。



```
.text:00401289                 mov     [ebp-4], eax
.text:0040128C                 fidiv   dword ptr [ebp-4]
.text:0040128F                 movsx   eax, byte ptr [ebp-19h]
.text:00401293                 sub     eax, ecx
.text:00401295                 mov     [ebp-4], eax
.text:00401298                 fsubp   st(1), st
.text:0040129A                 fimul   dword ptr [ebp-4]
.text:0040129D                 fmul    ds:flt_40711C
.text:004012A3                 fstp    dword ptr [ebp-4]
.text:004012A6                 jz      short near ptr loc_4012AA+1
.text:004012A8                 jnz     short near ptr loc_4012AA+1
.text:004012AA
.text:004012AA loc_4012AA:                             ; CODE XREF: .text:004012A6↑j
.text:004012AA                                         ; .text:004012A8↑j
.text:004012AA                 call    near ptr 48CB15h
.text:004012AF                 xor     ax, 7
.text:004012B3                 fld     dword ptr [ebp-4]
.text:004012B6                 fcomp   ds:flt_407118
.text:004012BC                 push    0
.text:004012BE                 push    offset aCrackme2017Ctf ; "CrackMe 2017 CTF"
.text:004012C3                 fnstsw  ax
.text:004012C5                 sahf
.text:004012C6                 jnz     short loc_4012D6
.text:004012C8                 push    offset aRegistrationSu ; "Registration successful !"
.text:004012CD                 jmp     short loc_4012DB
.text:004012CF ; ---------------------------------------------------------------------------
.text:004012CF
.text:004012CF loc_4012CF:                             ; CODE XREF: .text:00401229↑j
.text:004012CF                                         ; .text:00401235↑j ...
.text:004012CF                 push    0
.text:004012D1                 push    offset aCrackme2017C_0 ; "CrackMe 2017 CTF v2"
.text:004012D6
.text:004012D6 loc_4012D6:                             ; CODE XREF: .text:004012C6↑j
.text:004012D6                 push    offset aError   ; "error !"
.text:004012DB
.text:004012DB loc_4012DB:                             ; CODE XREF: .text:004012CD↑j
.text:004012DB                 push    hWnd
.text:004012E1                 call    ds:MessageBoxA
00001293 00401293: .text:00401293
```

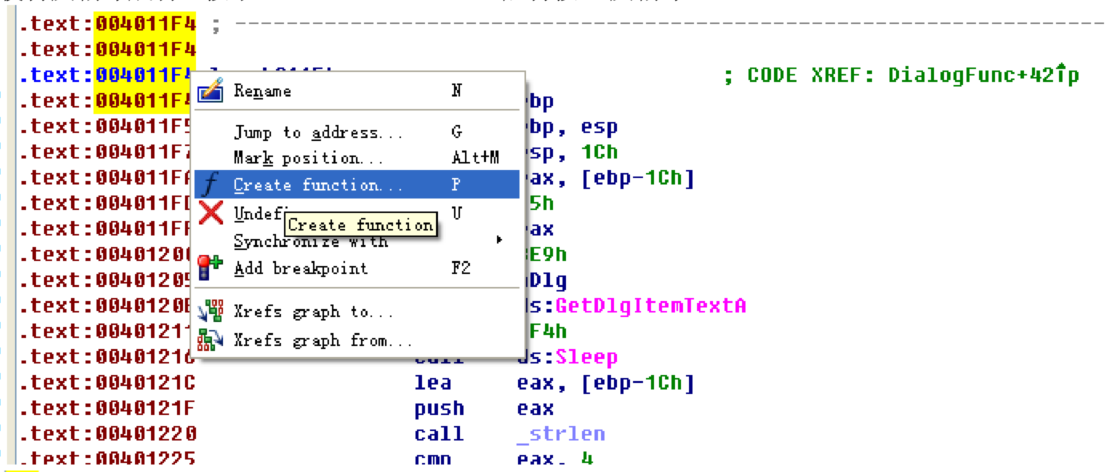
2. compare处找到了，我们来看看它的相关算法。首先祭出F5大法，可惜出师不利



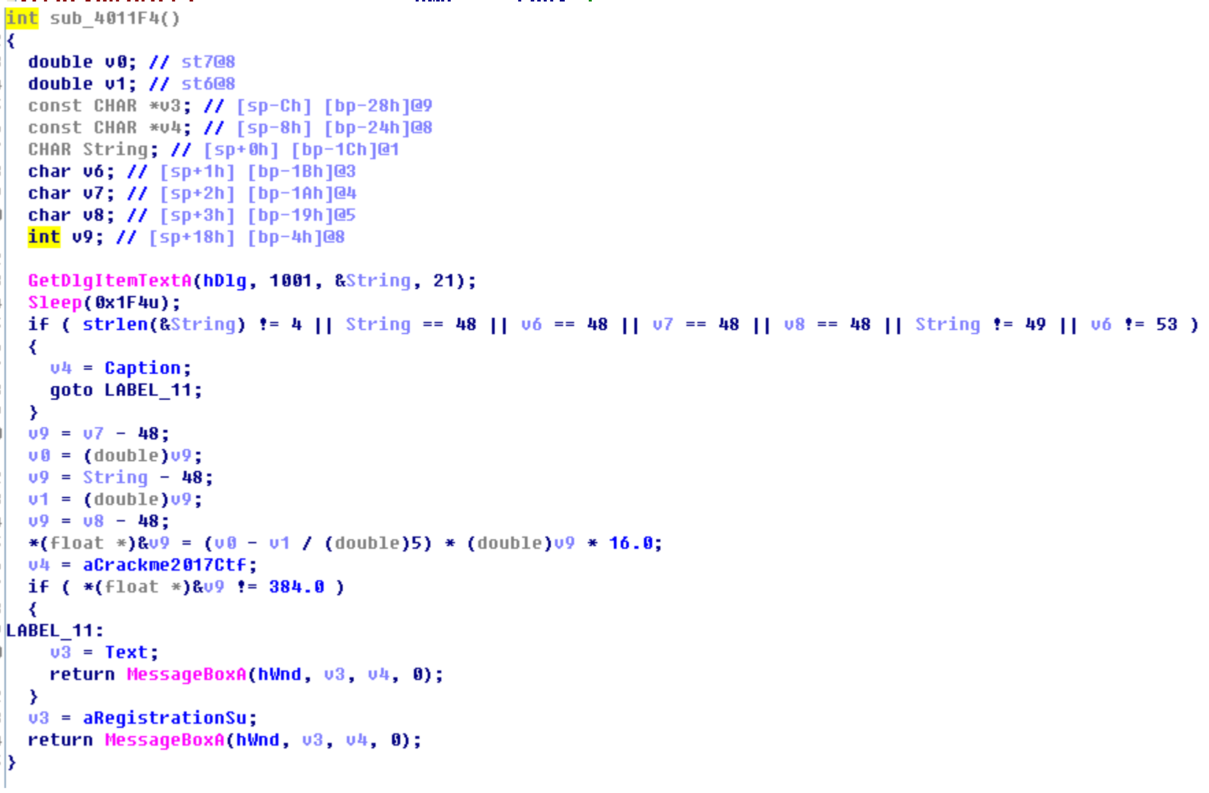
```
.text:004011F4 loc_4011F4:                             ; CODE XREF: DialogFunc+42↑p
.text:004011F4                 push    ebp
.text:004011F5                 mov     ebp, esp
.text:004011F7                 sub     esp, 1Ch
.text:004011FA                 lea     eax, [ebp-1Ch]
.text:004011FD                 push    15h
.text:004011FF                 push    eax
.text:00401200                 push    3E9h
.text:00401205                 push    hDlg
.text:0040120B                 call    ds:GetDlgItemTextA
.text:00401211                 push    1F4h
.text:00401216                 call    ds:Sleep
.text:0040121C
.text:0040121F
.text:00401220
.text:00401225
.text:00401228
.text:00401229
.text:0040122F
.text:00401231
.text:00401232
.text:00401235                 jz      loc_4012CF
.text:0040123B                 cmp     [ebp-1Bh], cl
.text:0040123E                 jz      loc_4012CF
.text:00401244                 cmp     [ebp-1Ah], cl
.text:00401247                 jz      loc_4012CF
.text:0040124D                 cmp     [ebp-19h], cl
.text:00401250                 jz      short loc_4012CF
.text:00401252                 cmp     byte ptr [ebp-1Ch], 31h
```

Warning
Please position the cursor within a function
OK
Don't display this message again (for this session only)

程序通过一些jmp方式跳过一些特殊字节，起到了模糊静态反编译的效果。通过将0x401262和0x4012AA处的指令nop掉，使得反编译成功。按下P（Create Function）后再按F5反编译。

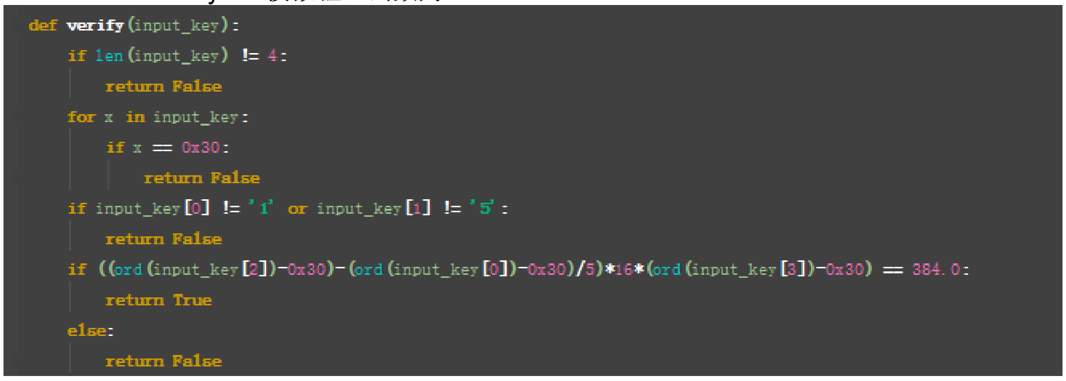


```
.text:004011F4 ; ---------------------------------------------------------------------
.text:004011F4
.text:004011F4                                         ; CODE XREF: DialogFunc+42↑p
.text:004011F4                                    bp
.text:004011F5                                    bp, esp
.text:004011F7                                    sp, 1Ch
.text:004011F                                     ax, [ebp-1Ch]
.text:004011F                                     5h
.text:004011F                                     ax
.text:0040120                                     E9h
.text:0040120                                     Dlg
.text:0040120                                     s:GetDlgItemTextA
.text:0040121                                     F4h
.text:0040121                                     s:Sleep
.text:0040121C                 lea     eax, [ebp-1Ch]
.text:0040121F                 push    eax
.text:00401220                 call    _strlen
.text:00401225                 cmp     eax, 4
```



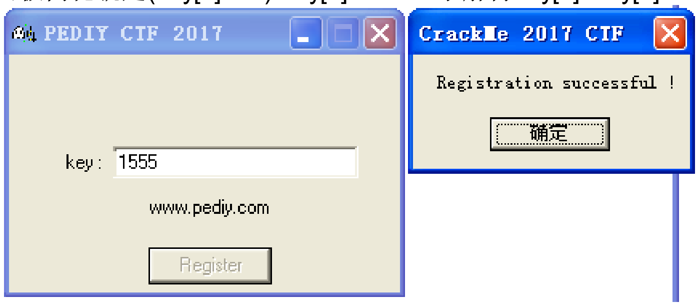
```
int sub_4011F4()
{
  double v0; // st7@8
  double v1; // st6@8
  const CHAR *v3; // [sp-Ch] [bp-28h]@9
  const CHAR *v4; // [sp-8h] [bp-24h]@8
  CHAR String; // [sp+0h] [bp-1Ch]@1
  char v6; // [sp+1h] [bp-1Bh]@3
  char v7; // [sp+2h] [bp-1Ah]@4
  char v8; // [sp+3h] [bp-19h]@5
  int v9; // [sp+18h] [bp-4h]@8

  GetDlgItemTextA(hDlg, 1001, &String, 21);
  Sleep(0x1F4u);
  if ( strlen(&String) != 4 || String == 48 || v6 == 48 || v7 == 48 || v8 == 48 || String != 49 || v6 != 53 )
  {
    v4 = Caption;
    goto LABEL_11;
  }
  v9 = v7 - 48;
  v0 = (double)v9;
  v9 = String - 48;
  v1 = (double)v9;
  v9 = v8 - 48;
  *(float *)&v9 = (v0 - v1 / (double)5) * (double)v9 * 16.0;
  v4 = aCrackme2017Ctf;
  if ( *(float *)&v9 != 384.0 )
  {
LABEL_11:
    v3 = Text;
    return MessageBoxA(hWnd, v3, v4, 0);
  }
  v3 = aRegistrationSu;
  return MessageBoxA(hWnd, v3, v4, 0);
}
```

Python模拟验证函数为



```python
def verify(input_key):
    if len(input_key) != 4:
        return False
    for x in input_key:
        if x == 0x30:
            return False
    if input_key[0] != '1' or input_key[1] != '5':
        return False
    if ((ord(input_key[2])-0x30)-(ord(input_key[0])-0x30)/5)*16*(ord(input_key[3])-0x30) == 384.0:
        return True
    else:
        return False
```

3. 终极简化就是(key[2]-0.2)*key[3]=24.0，求解得key[2]=key[3]=5，即key="1555"



Registration successful !

（如有错误，敬请指出~）

转载于:https://my.oschina.net/u/3281747/blog/1595845