看雪 FPC--reverse

原创

还木人 于 2018-10-05 12:20:59 发布 115 收藏

分类专栏： CTF 新操作

本文链接：https://blog.csdn.net/caobo_lq666/article/details/82943069

版权

CTF 同时被 2 个专栏收录

7 篇文章 0 订阅

订阅专栏

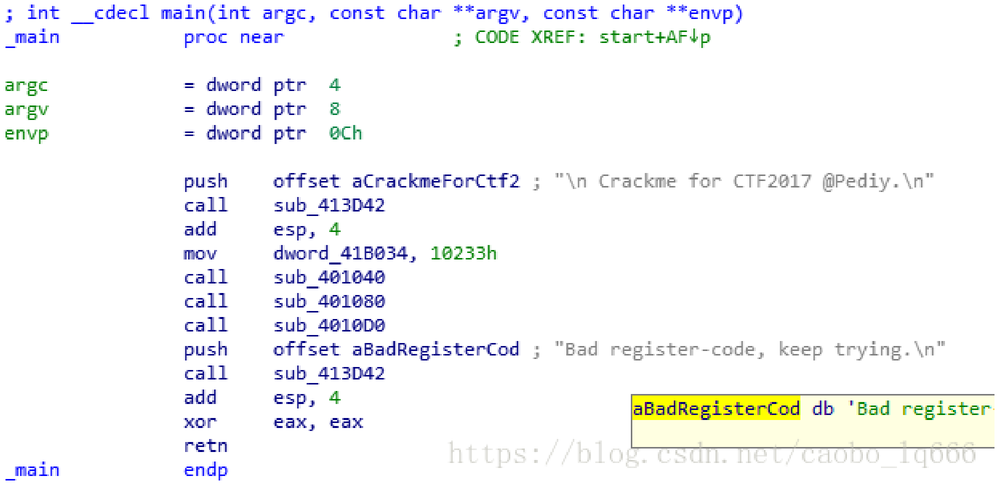
新操作

8 篇文章 0 订阅

订阅专栏

很长时间没有在这里记录Writeup了，这次这道题目实在太让我兴奋了。有感而记。

拖进IDA;

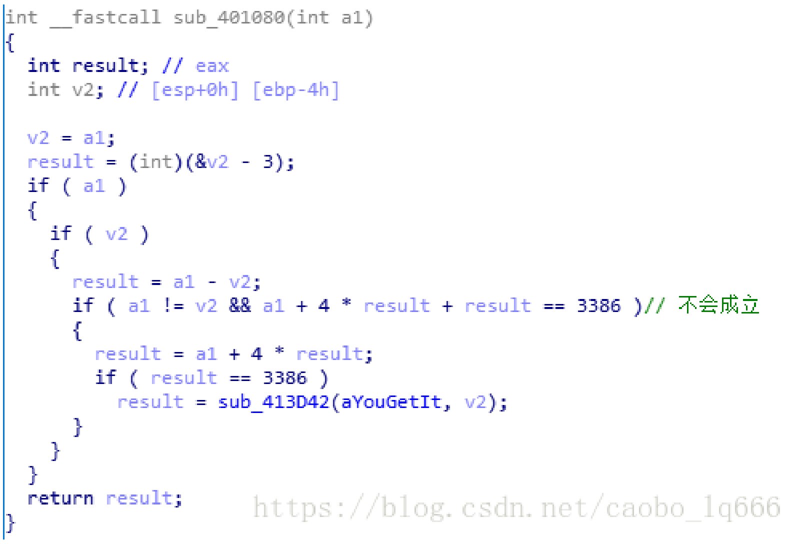


```
; int __cdecl main(int argc, const char **argv, const char **envp)
_main           proc near               ; CODE XREF: start+AF↓p

argc            = dword ptr  4
argv            = dword ptr  8
envp            = dword ptr  0Ch

                push    offset aCrackmeForCtf2 ; "\n Crackme for CTF2017 @Pediy.\n"
                call    sub_413D42
                add     esp, 4
                mov     dword_41B034, 10233h
                call    sub_401040
                call    sub_401080
                call    sub_4010D0
                push    offset aBadRegisterCod ; "Bad register-code, keep trying.\n"
                call    sub_413D42
                add     esp, 4
                xor     eax, eax
                retn
_main           endp
```

惊奇的发现，除了scanf之外，只需要两个函数就执行到了"Bad.....";

依次查看两个验证函数;

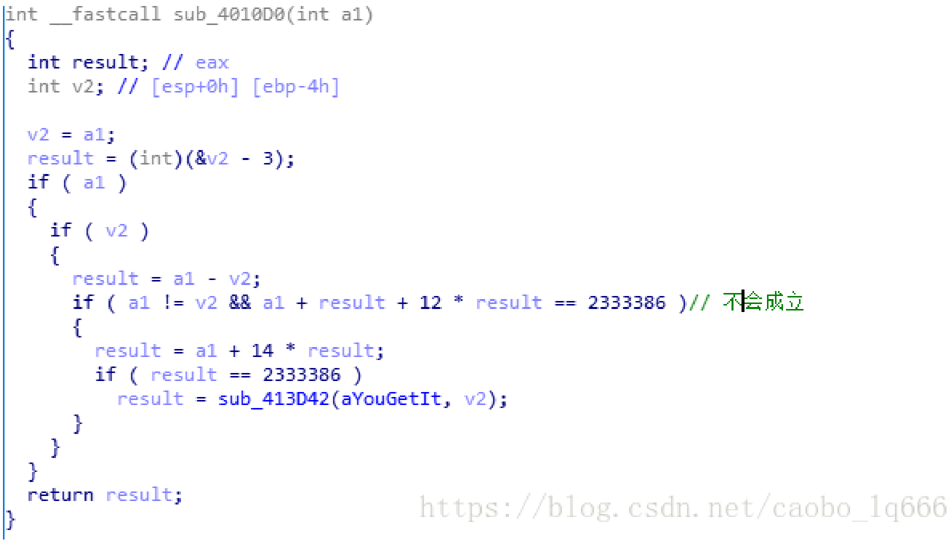
1.sub_401080



```
int __fastcall sub_401080(int a1)
{
  int result; // eax
  int v2; // [esp+0h] [ebp-4h]

  v2 = a1;
  result = (int)(&v2 - 3);
  if ( a1 )
  {
    if ( v2 )
    {
      result = a1 - v2;
      if ( a1 != v2 && a1 + 4 * result + result == 3386 )// 不会成立
      {
        result = a1 + 4 * result;
        if ( result == 3386 )
          result = sub_413D42(aYouGetIt, v2);
      }
    }
  }
  return result;
}
```

会发现 a1与v2是恒等的，if 不会成立，也就是永远不会“YouGetIt”

2.sub_4010D0



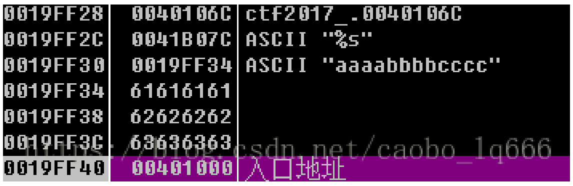
```
int __fastcall sub_4010D0(int a1)
{
  int result; // eax
  int v2; // [esp+0h] [ebp-4h]

  v2 = a1;
  result = (int)(&v2 - 3);
  if ( a1 )
  {
    if ( v2 )
    {
      result = a1 - v2;
      if ( a1 != v2 && a1 + result + 12 * result == 2333386 )// 不会成立
      {
        result = a1 + 14 * result;
        if ( result == 2333386 )
          result = sub_413D42(aYouGetIt, v2);
      }
    }
  }
  return result;
}
```

同样的道理，这里也永远不会"Yougetit"

困惑：这就很不正常了，就是说程序的正常执行流程下，永远不会pass；

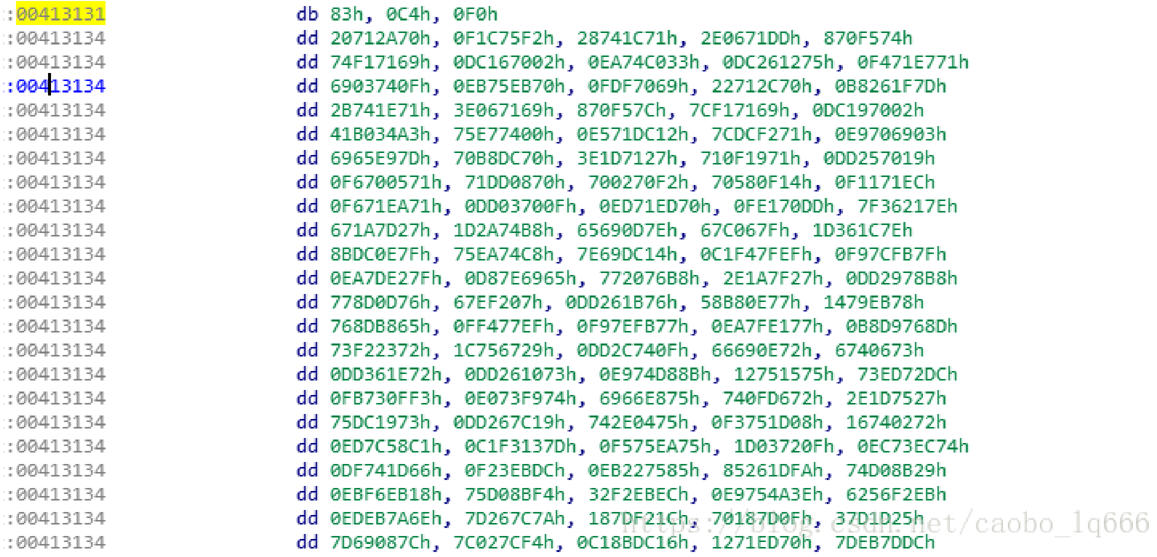
暗示，可能需要特殊的输入，改变程序的执行流程（栈溢出）

拖进OD，在scanf入flag后通过观察栈，得知，12个字符恰好覆盖返回地址以上部分。



```
0019FF28  0040106C ctf2017_.0040106C
0019FF2C  0041B07C ASCII "%s"
0019FF30  0019FF34 ASCII "aaaabbbbcccc"
0019FF34  61616161
0019FF38  62626262
0019FF3C  63636363
0019FF40  00401000 入口地址
```

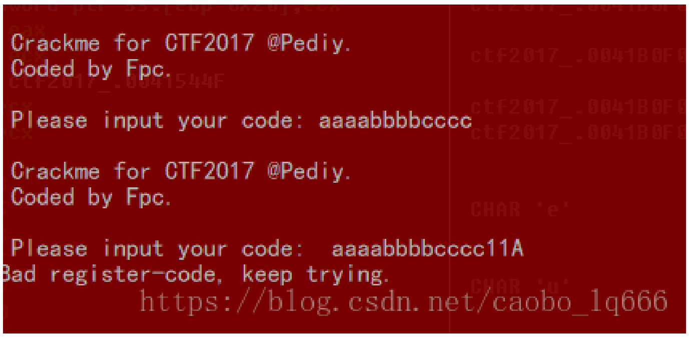
那么，要把程序带到什么地方呢（用哪个地址来覆盖原来的返回地址）；
由于输入的全部是可打印字符，可以在IDA中手动查看可疑的地址字段。发现00413131，是一段很像花指令的东西。



```
:00413131                  db 83h, 0C4h, 0F0h
:00413134                  dd 20712A70h, 0F1C75F2h, 28741C71h, 2E0671DDh, 870F574h
:00413134                  dd 74F17169h, 0DC167002h, 0EA74C033h, 0DC261275h, 0F471E771h
:00413134                  dd 6903740Fh, 0EB75EB70h, 0FDF7069h, 22712C70h, 0B8261F7Dh
:00413134                  dd 2B741E71h, 3E067169h, 870F57Ch, 7CF17169h, 0DC197002h
:00413134                  dd 41B034A3h, 75E77400h, 0E571DC12h, 7CDCF271h, 0E9706903h
:00413134                  dd 6965E97Dh, 70B8DC70h, 3E1D7127h, 710F1971h, 0DD257019h
:00413134                  dd 0F6700571h, 71DD0870h, 700270F2h, 70580F14h, 0F1171ECh
:00413134                  dd 0F671EA71h, 0DD03700Fh, 0ED71ED70h, 0FE170DDh, 7F36217Eh
:00413134                  dd 671A7D27h, 1D2A74B8h, 65690D7Eh, 67C067Fh, 1D361C7Eh
:00413134                  dd 8BDC0E7Fh, 75EA74C8h, 7E69DC14h, 0C1F47FEFh, 0F97CFB7Fh
:00413134                  dd 0EA7DE27Fh, 0D87E6965h, 772076B8h, 2E1A7F27h, 0DD2978B8h
:00413134                  dd 778D0D76h, 67EF207h, 0DD261B76h, 58B80E77h, 1479EB78h
:00413134                  dd 768DB865h, 0FF477EFh, 0F97EFB77h, 0EA7FE177h, 0B8D9768Dh
:00413134                  dd 73F22372h, 1C756729h, 0DD2C740Fh, 66690E72h, 6740673h
:00413134                  dd 0DD361E72h, 0DD261073h, 0E974D88Bh, 12751575h, 73ED72DCh
:00413134                  dd 0FB730FF3h, 0E073F974h, 6966E875h, 740FD672h, 2E1D7527h
:00413134                  dd 75DC1973h, 0DD267C19h, 742E0475h, 0F3751D08h, 16740272h
:00413134                  dd 0ED7C58C1h, 0C1F3137Dh, 0F575EA75h, 1D03720Fh, 0EC73EC74h
:00413134                  dd 0DF741D66h, 0F23EBDCh, 0EB227585h, 85261DFAh, 74D08B29h
:00413134                  dd 0EBF6EB18h, 75D08BF4h, 32F2EBECh, 0E9754A3Eh, 6256F2EBh
:00413134                  dd 0EDEB7A6Eh, 7D267C7Ah, 187DF21Ch, 70187D0Fh, 37D1D25h
:00413134                  dd 7D69087Ch, 7C027CF4h, 0C18BDC16h, 1271ED70h, 7DEB7DDCh
```

尝试一下，输入‘aaaabbbbcccc11A’;

惊奇的发现，程序执行了验证算法，（证实了思路）

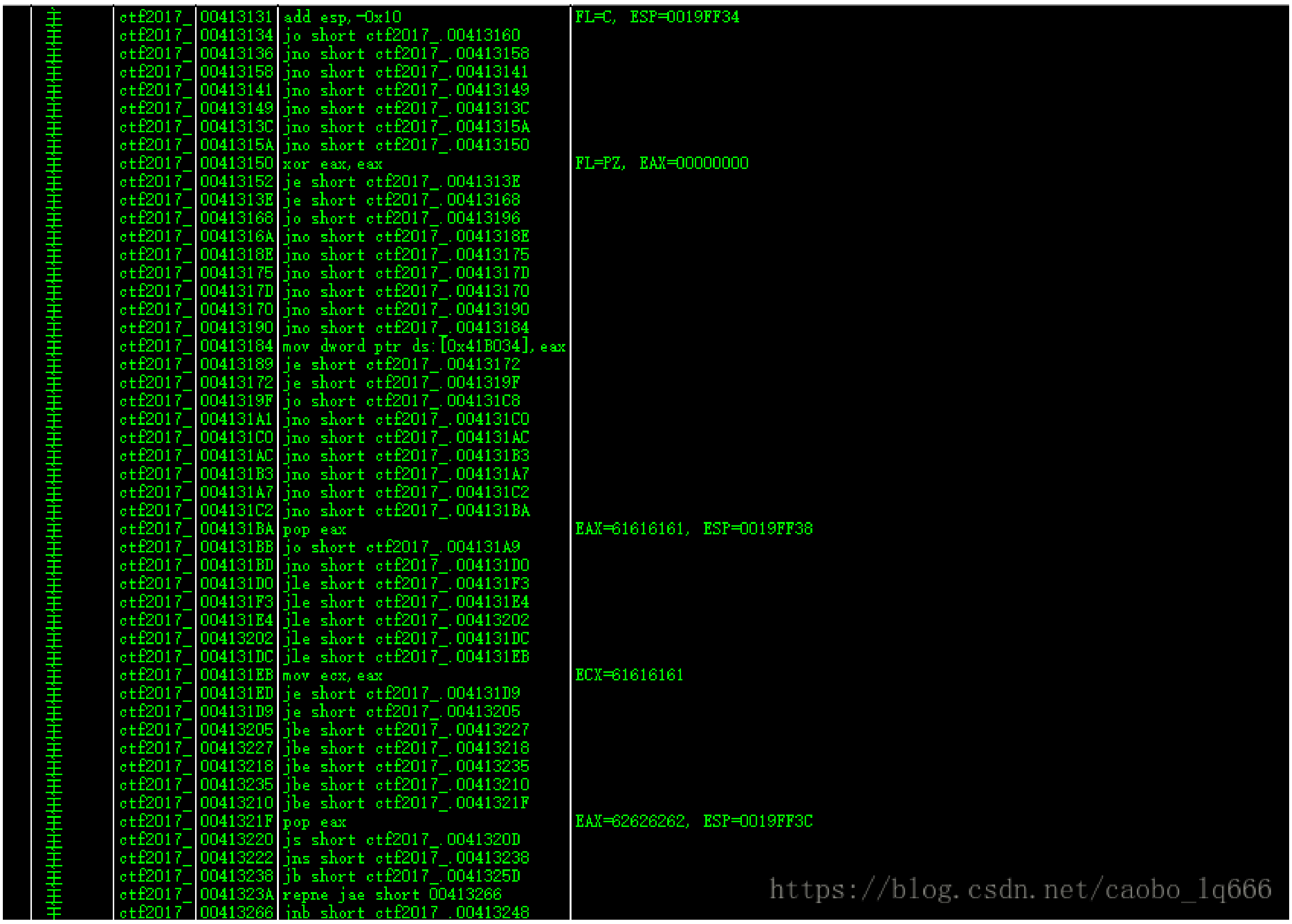


```
Crackme for CTF2017 @Pediy.
Coded by Fpc.

Please input your code: aaaabbbbcccc

Crackme for CTF2017 @Pediy.
Coded by Fpc.

Please input your code:  aaaabbbbcccc11A
Bad register-code, keep trying.
```

由于花指令的存在，我选择了动态调试

在OD中开启RUN TRACE,单步跟入调试(F8或者f7跟踪，遇到跳转，jz,jl　可以改变z标志位改变流程，继续跟踪）

最后查看执行过的指令。



```
ctf2017_ 00413131 add esp,-0x10                          FL=C, ESP=0019FF34
ctf2017_ 00413134 jo short ctf2017_.00413160
ctf2017_ 00413136 jno short ctf2017_.00413158
ctf2017_ 00413158 jno short ctf2017_.00413141
ctf2017_ 00413141 jno short ctf2017_.00413149
ctf2017_ 00413149 jno short ctf2017_.0041313C
ctf2017_ 0041313C jno short ctf2017_.0041315A
ctf2017_ 0041315A jno short ctf2017_.00413150
ctf2017_ 00413150 xor eax,eax                            FL=PZ, EAX=00000000
ctf2017_ 00413152 je short ctf2017_.0041313E
ctf2017_ 0041313E je short ctf2017_.00413168
ctf2017_ 00413168 jo short ctf2017_.00413196
ctf2017_ 0041316A jno short ctf2017_.0041318E
ctf2017_ 0041318E jno short ctf2017_.00413175
ctf2017_ 00413175 jno short ctf2017_.0041317D
ctf2017_ 0041317D jno short ctf2017_.00413170
ctf2017_ 00413170 jno short ctf2017_.00413190
ctf2017_ 00413190 jno short ctf2017_.00413184
ctf2017_ 00413184 mov dword ptr ds:[0x41B034],eax
ctf2017_ 00413189 je short ctf2017_.00413172
ctf2017_ 00413172 je short ctf2017_.0041319F
ctf2017_ 0041319F jo short ctf2017_.004131C8
ctf2017_ 004131A1 jno short ctf2017_.004131C0
ctf2017_ 004131C0 jno short ctf2017_.004131AC
ctf2017_ 004131AC jno short ctf2017_.004131B3
ctf2017_ 004131B3 jno short ctf2017_.004131A7
ctf2017_ 004131A7 jno short ctf2017_.004131C2
ctf2017_ 004131C2 jno short ctf2017_.004131BA
ctf2017_ 004131BA pop eax                                EAX=61616161, ESP=0019FF38
ctf2017_ 004131BB jo short ctf2017_.004131A9
ctf2017_ 004131BD jno short ctf2017_.004131D0
ctf2017_ 004131D0 jle short ctf2017_.004131F3
ctf2017_ 004131F3 jle short ctf2017_.004131E4
ctf2017_ 004131E4 jle short ctf2017_.00413202
ctf2017_ 00413202 jle short ctf2017_.004131DC
ctf2017_ 004131DC jle short ctf2017_.004131EB
ctf2017_ 004131EB mov ecx,eax                            ECX=61616161
ctf2017_ 004131ED je short ctf2017_.004131D9
ctf2017_ 004131D9 je short ctf2017_.00413205
ctf2017_ 00413205 jbe short ctf2017_.00413227
ctf2017_ 00413227 jbe short ctf2017_.00413218
ctf2017_ 00413218 jbe short ctf2017_.00413235
ctf2017_ 00413235 jbe short ctf2017_.00413210
ctf2017_ 00413210 jbe short ctf2017_.0041321F
ctf2017_ 0041321F pop eax                                EAX=62626262, ESP=0019FF3C
ctf2017_ 00413220 js short ctf2017_.0041320D
ctf2017_ 00413222 jns short ctf2017_.00413238
ctf2017_ 00413238 jb short ctf2017_.0041325D
ctf2017_ 0041323A repne jae short 00413266
ctf2017_ 00413266 jnb short ctf2017_.00413248
```

拷贝下来，过滤得到有效的指令

分析汇编代码：

```
add esp,-0x10
xor eax,eax
mov dword ptr ds:[0x41B034],eax
pop eax
mov ecx,eax
pop eax
repne jae short 00413266
mov ebx,eax
pop eax
mov edx,eax
mov edx,eax
mov eax,ecx
sub eax,ebx
shl eax,0x2
add eax,ecx
add eax,edx
sub eax,0xEAF917E2
```

对应的方程：

(x-y0<<2 + x + z == 0xEAF917E2

```
add eax,ecx
sub eax,ebx
mov ebx,eax
shl eax,1
add eax,ebx
add eax,ecx
mov ecx,eax
add eax,edx
sub eax,0xE8F508C8
```

对应的方程：

(x-y) << 1 + (x-y) + x + z ==0XE8F508C8

```
mov eax,ecx
mov eax,ecx
sub eax,edx
sub eax,0xC0A3C68
```

对应的方程：

(x-y) << 1 + (x-y) + x - z == 0xC0A3C68

其实就是解三元方程组
用Z3求解器，或者在线求解
得到x,y,z,在加上'11A'就是flag

```
Just0For0Fun11A
```