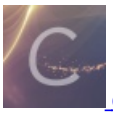
XCTF-Reverse-ExerciseArea-011-writeup

原创

y4ung 于 2019-08-10 15:49:51 发布 4008 收藏

分类专栏： ctf 文章标签： ctf

本文链接：https://blog.csdn.net/qq_35056292/article/details/99080927



0x00 介绍

本题是xctf攻防世界中Reverse的新手第十一题。题目来源：`CSAW CTF 2014`

给了一个二进制文件csaw2013reversing2.exe，需要对该二进制文件进行逆向分析，找到flag

实验环境：IDA Pro 7.0，ollydbg

本题考查的是反调试，IsDebuggerPresent函数，int 3中断

0x01 解题过程

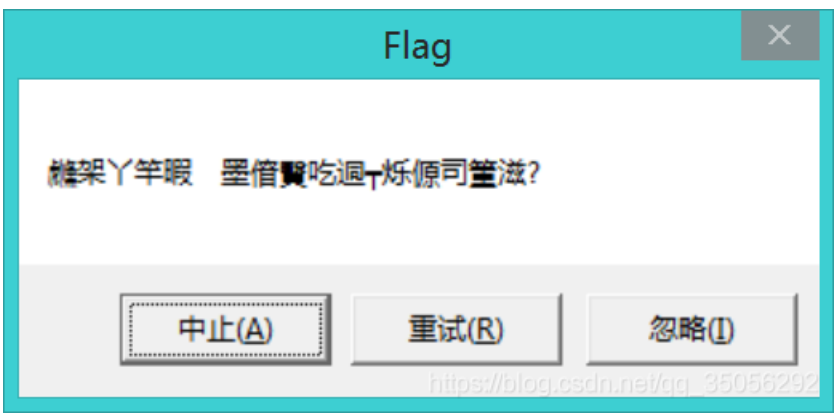
1.1 文件分析

windows下的可执行文件，32位

```
root@kali:~/hzy/ctf-learning# file csaw2013reversing2.exe
csaw2013reversing2.exe: PE32 executable (console) Intel 80386, for MS Windows
root@kali:~/hzy/ctf-learning#
```

1.2 逆向分析

1. 在windows下运行该文件，发现是一坨乱码， = =||



2. 能知道的就只有`Flag`、`中止`、`重试`和`忽略`四个字符串，在这里我先在IDA和Ollydbg里搜索字符串Flag，找到引用它的地方。[ebp-0xC]是乱码字符串的起始地址

并且在IDA中可以发现基本块0x40108C ⇒ 0x401094处，调用了数据段ds(Data Segment)的`IsDebuggerPresent`函数，后面判断了返回值是否为0。因此，猜测这是检查程序是否在调试的，可以看到，调试和不在调试，程序执行流是往不同的方向进行跳转的。

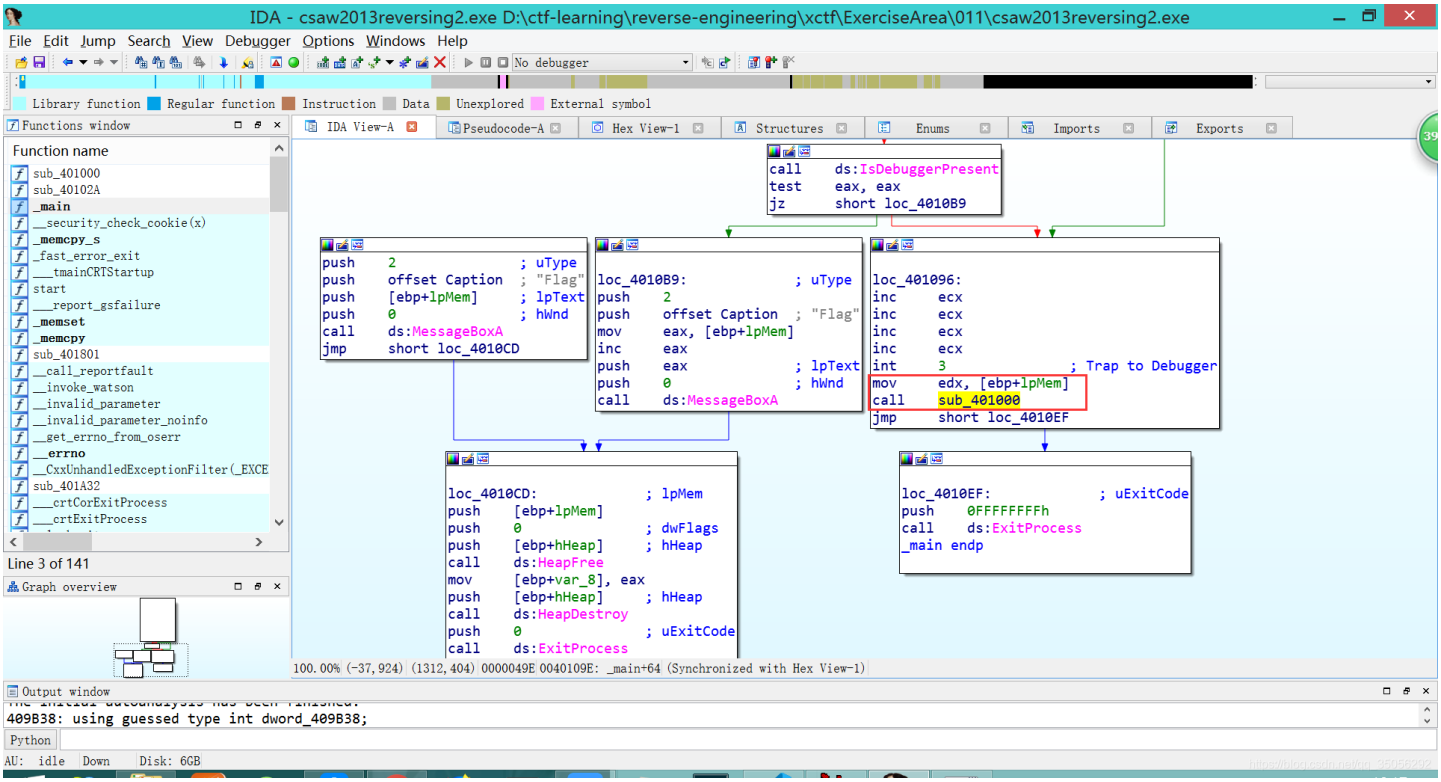
```
.text:0040108C                  call    ds:IsDebuggerPresent
.text:00401092                  test    eax, eax
.text:00401094                  jz      short loc_4010B9
```

3. 一开始我先修改了`IsDebuggerPresent`函数的返回值，将其改为0，发现显示的还是乱码；然后我发现基本块0x4010A5 ⇒ 0x4010B7是没有入口的，因此在修改了函数返回值的基础上又修改了`.text:00401094 jz short loc_4010B9`的跳转地址，跳转到基本块0x4010A5 ⇒ 0x4010B7

```
push    2               ; uType
push    offset Caption  ; "Flag"
push    [ebp+lpMem]     ; lpText
push    0               ; hWnd
call    ds:MessageBoxA
jmp     short loc_4010CD
```

发现还是乱码.....

4. 最后决定在ollydbg中测试运行调试的分支会有什么结果

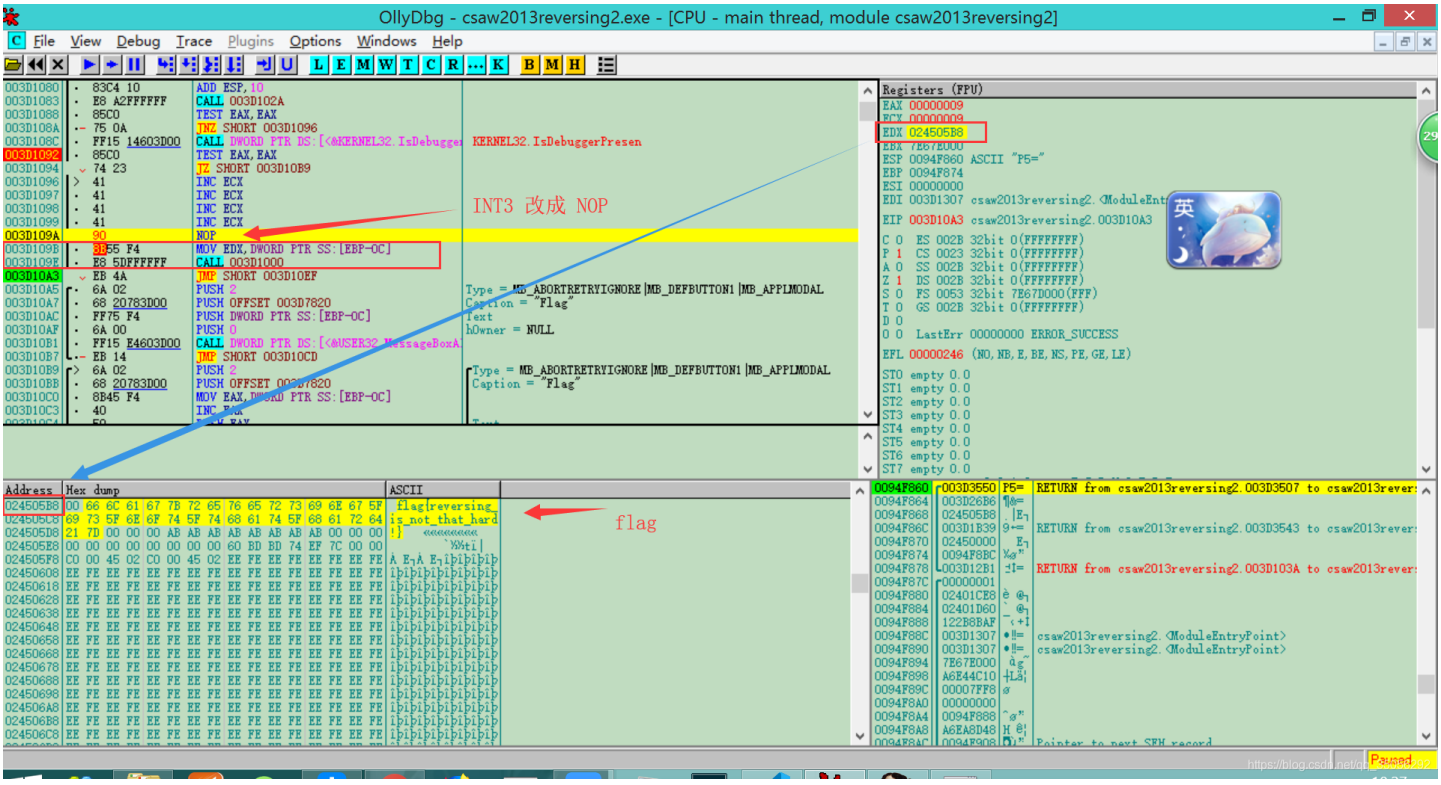


可以看到基本块0x401096 ⇒ 0x4010A3中

```
.text:00401096 loc_401096:                             ; CODE XREF: _main+50↑j
.text:00401096                 inc     ecx
.text:00401097                 inc     ecx
.text:00401098                 inc     ecx
.text:00401099                 inc     ecx
.text:0040109A                 int     3               ; Trap to Debugger
.text:0040109B                 mov     edx, [ebp+lpMem]
.text:0040109E                 call    sub_401000
.text:004010A3                 jmp     short loc_4010EF
```

有个`int 3`中断，为调试断点指令。然后将字符串起始地址赋值给了edx寄存器，调用了函数`sub_401000`。最后直接跳转到进程结束的函数调用

在ollydbg中调试，可以看到当执行完调用函数`sub_401000`以后，edx寄存器中的乱码内容被解密出来了，拿到flag。。并且可以看到该函数`sub_401000`是解密用的函数



flag为：`flag{reversing_is_not_that_hard!}`

ps: 如果还要显示在messagebox中的话，就在ollydbg中，把下面指令中的跳转地址0x003D10EF修改为003D10B9即可

```
003D10A3    JMP SHORT 003D10EF
```