

# 攻防世界csaw2013reversing2Writeup

原创

bin\_cat 于 2022-01-08 14:20:56 发布 231 收藏

文章标签: 安全

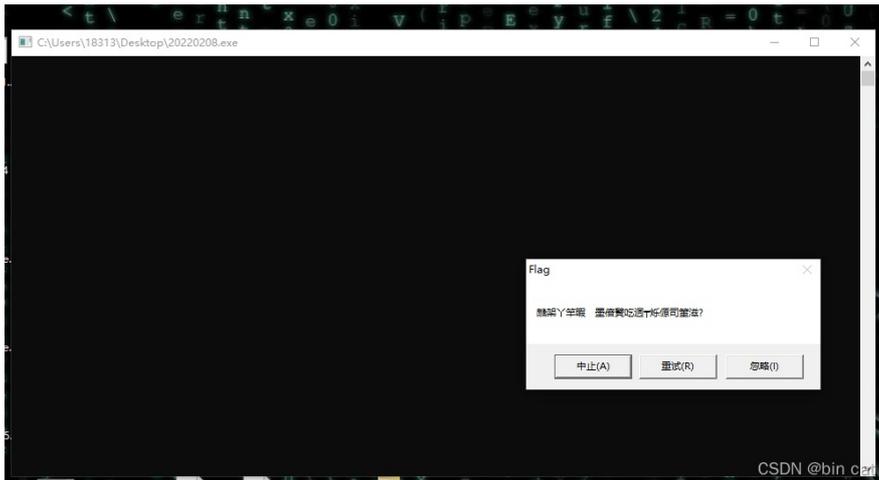
版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/m0\\_58348028/article/details/122378395](https://blog.csdn.net/m0_58348028/article/details/122378395)

版权

32位无壳

先试着运行exe文件, 得到如下结果符合题目 描述输出的玩意是乱码



用ida解析

```
IDA View-A Pseudocode-A Hex View-I Structures Enum
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
    int v3; // ecx
    CHAR *lpMem; // [esp+8h] [ebp-Ch]
    HANDLE hHeap; // [esp+10h] [ebp-4h]

    hHeap = HeapCreate(0x40000u, 0, 0);
    lpMem = (CHAR *)HeapAlloc(hHeap, 8u, SourceSize + 1);
    memcpy_s(lpMem, SourceSize, &unk_409B10, SourceSize);
    if ( !sub_40102A() && !IsDebuggerPresent() )
    {
        MessageBoxA(0, lpMem + 1, "Flag", 2u);
        HeapFree(hHeap, 0, lpMem);
        HeapDestroy(hHeap);
        ExitProcess(0);
    }
    _debugbreak();
    sub_401000(v3 + 4, lpMem);
    ExitProcess(0xFFFFFFFF);
}
```

CSDN @bin cat

主函数逻辑, 跟进sub\_40102A()函数中

```
IDA View-A Pseudocode-A
1 int sub_40102A()
2 {
3     return 0;
4 }
```

CSDN @bin cat

发现函数的返回值默认为0, 但是!0就意味着if所包含的语句会一直被默认执行, 联想到我们一开始看到程序运行起来的乱码, 我们就确定if所包含的语句就是导致乱码发生的原因



# flag{reversing\_is\_not\_that\_hard!}

```

00391099 41          inc ecx
0039109A 90          nop
0039109B 8B55 F4    mov  edx,dword ptr ss:[ebp-c]
0039109C E8 5DFFFFFF call 20220208.391000
003910A3 90          nop
003910A4 4A          dec  eax
003910A5 6A 02     push 2
003910A7 68 20783900 push 20220208.397820
003910AC FF75 F4    push dword ptr ss:[ebp-c]
003910AF 6A 00     push 0
003910B1 FF15 E4603900 call dword ptr ds:[&MessageBoxA]
003910B7 EB 14     jmp 20220208.3910CD
003910B9 6A 02     push 2
003910BB 68 20783900 push 20220208.397820
003910C0 8B45 F4    mov  eax,dword ptr ss:[ebp-c]
003910C3 40          inc  eax
003910C4 50          push eax
003910C5 6A 00     push 0
003910C7 FF15 E4603900 call dword ptr ds:[&MessageBoxA]
003910CD FF75 F4    push dword ptr ss:[ebp-c]
003910D0 6A 00     push 0
003910D2 FF75 FC    push dword ptr ss:[ebp-4]
003910D5 FF15 08603900 call dword ptr ds:[&HeapFree]
003910DB 8945 F8    mov  dword ptr ss:[ebp-8],eax
003910DE FF75 FC    push dword ptr ss:[ebp-4]
003910E1 FF15 0C603900 call dword ptr ds:[&HeapDestroy]
003910E7 6A 00     push 0
003910E9 FF15 00603900 call dword ptr ds:[&ExitProcess]
003910EF 6A FF     push FFFFFFFF
003910F1 FF15 00603900 call dword ptr ds:[&ExitProcess]
003910F7 C9          leave
003910F8 C3          ret
003910F9 3B0D 04903900 cmp  ecx,dword ptr ds:[399004]
003910FF 75 02     jne 20220208.391103
00391101 F3:C3     ret
00391103 E9 09020000 jmp 20220208.391311
00391108 8BFF     mov  edi,edi
    
```

CSDN @bin-cat

也可以将1处跳转nop掉单步走到003910C7去触发messageboxA

这样也能得flag

The screenshot shows a debugger interface with several panes. The main assembly pane highlights the instruction at address 003910C7: `call dword ptr ds:[&MessageBoxA]`. A red arrow points to this instruction. A dialog box titled "Flag" is displayed in the center, containing the text "flag{reversing\_is\_not\_that\_hard!}" and buttons for "中止(A)", "重试(R)", and "忽略(O)". The right-hand pane shows the CPU registers, with the instruction pointer (EIP) set to 003910C7. The bottom pane shows a list of memory addresses and their corresponding ASCII values, with the address 003910C7 highlighted.

CSDN @bin cat