

# xctf攻防世界csaw2013reversing2 writeup

原创

qq\_112419837 于 2020-11-05 17:15:18 发布 73 收藏  
版权声明：本文为博主原创文章，遵循CC 4.0 BY-SA 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/qq\\_42983283/article/details/109515484](https://blog.csdn.net/qq_42983283/article/details/109515484)  
版权

csaw2013reversing2 134 最佳Writeup由N1ce • Okam提供

难度系数: ★★★★★ 5.0

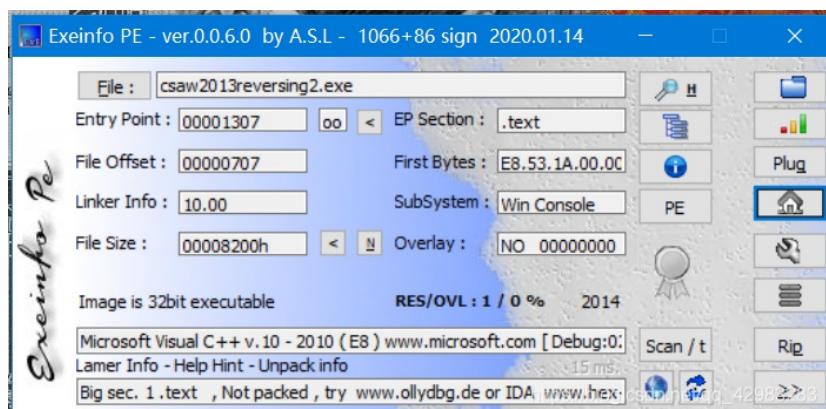
题目来源: Source: CSAW CTF 2014

题目描述: 听说运行就能拿到Flag, 不过菜鸟运行的结果不知道为什么是乱码

题目场景: 暂无

题目附件: 附件1 [https://blog.csdn.net/qq\\_42983283](https://blog.csdn.net/qq_42983283)

下载文件，查看：



ida打开，f5:

大意是，sub\_40102A或者IsDebuggerPresent这两个函数返回值有一个为1的话，程序貌似就要中断，但是最后的flag好像是固定的，也不需要输入啥的，直接setup大法：

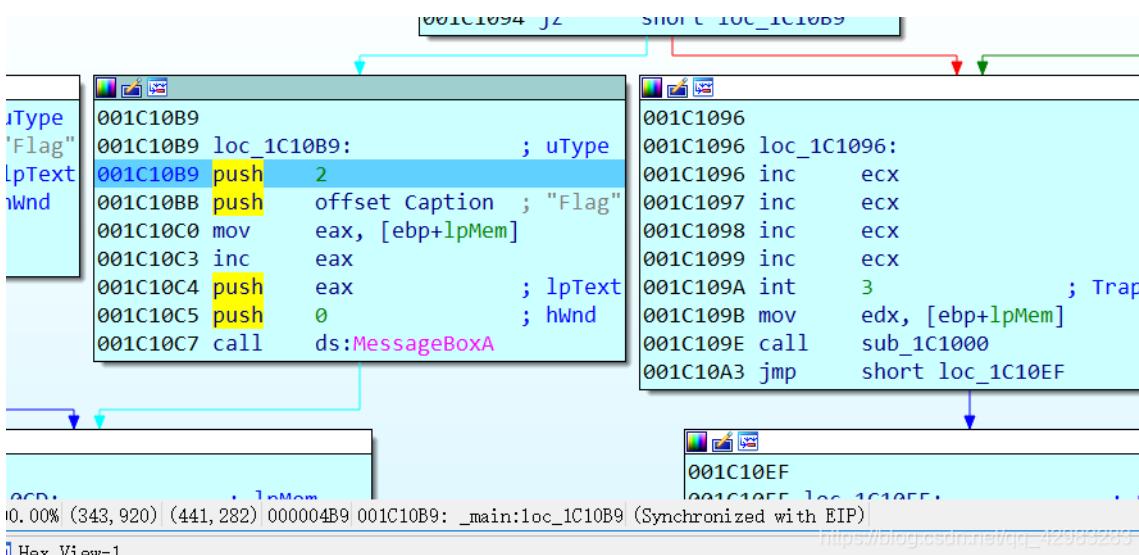
```

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]

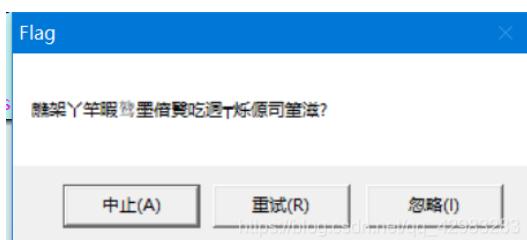
    hHeap = HeapCreate(0x400000u, 0, 0);
    lpMem = (CHAR *)HeapAlloc(hHeap, 8u, MaxCount + 1);
    memcpys(lpMem, MaxCount, &unk_409B10, MaxCount);
    if ( sub_40102A() || IsDebuggerPresent() )
    {
        __debugbreak();
        sub_401000(v3 + 4, lpMem);
        ExitProcess(0xFFFFFFFF);
    }
    MessageBoxA(0, lpMem + 1, "Flag", 2u);
    HeapFree(hHeap, 0, lpMem);
    HeapDestroy(hHeap);
    ExitProcess(0);
}

```

if那里下断点，动态运行，断下之后setip到flag这个程序段：



运行后是乱码：



感觉得执行一下sub\_1C1000，因为这个函数对第二个参数有一个赋值，也就是main中的lpMem：

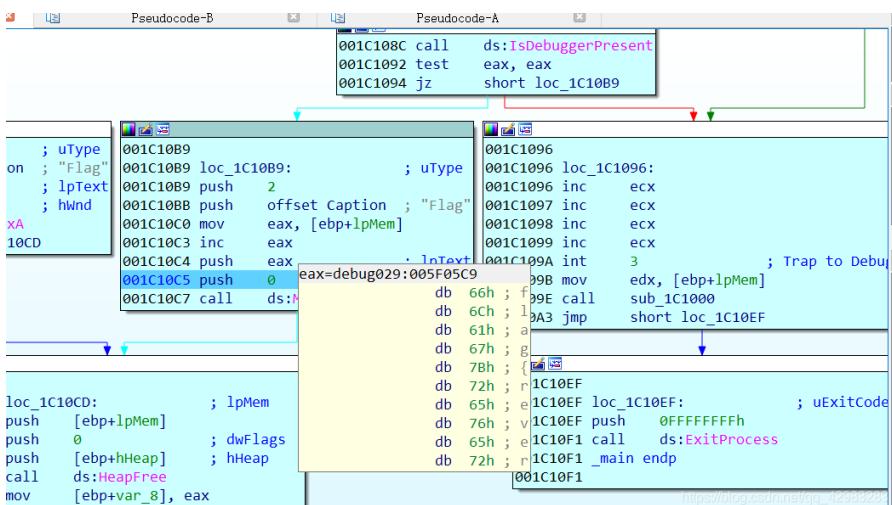
```

unsigned int __fastcall sub_1C1000(int a1, int a2)
{
    int v2; // esi
    unsigned int v3; // eax
    unsigned int v4; // ecx
    unsigned int result; // eax

    v2 = dword_1C9B38;
    v3 = a2 + 1 + strlen((const char *)(a2 + 1)) + 1;
    v4 = 0;
    result = ((v3 - (a2 + 2)) >> 2) + 1;
    if ( result )
    {
        do
            *(DWORD *)(a2 + 4 * v4++) ^= v2;
        while ( v4 < result );
    }
    return result;
}

```

那其实也好办，setip两次就行：



isdebuggerpresent那里不执行，直接到1c1000，然后就setip到1c10b9，eax里面就是flag。

双击lpMem也行，不过要+1：

```

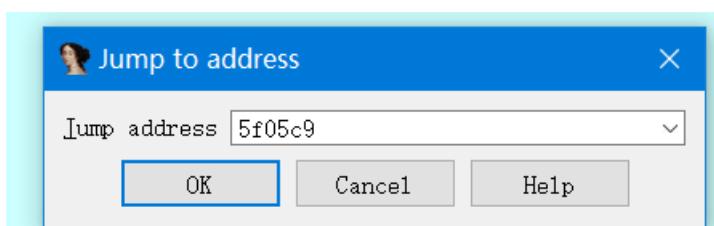
14     MessageBoxA(0, lpMem, "Flag", 2u);
15 }
16 MessageBoxA(0, lpMem + 1, "Flag", 2u);

```

```

J:008FF7F3 db 0
J:008FF7F4 db 0C8h
J:008FF7F5 db 5
J:008FF7F6 db 5Fh ; 
J:008FF7F7 db 0

```



• 5CB db 61h ; a  
• 5CC db 67h ; g  
• 5CD db 7Bh ; {  
• 5CE db 72h ; r  
• 5CF db 65h ; e  
• 5D0 db 76h ; v  
• 5D1 db 65h ; e  
• 5D2 db 72h ; r  
• 5D3 db 73h ; s  
• 5D4 db 69h ; i  
• 5D5 db 66h ; n  
• 5D6 db 67h ; g  
• 5D7 db 5Fh ; \_  
• 5D8 db 69h ; i  
• 5D9 db 73h ; s  
• 5DA db 5Fh ; \_  
• 5DB db 66h ; n  
• 5DC db 66h ; o  
• 5DD db 74h ; t

UNKNOWN 005F05CF: debug029:005F05CF (Synchronized with EIP, Hex View-1)  
<

UNKNOWN 005F05CF: debug029:005F05CF (Synchronized with EIP, Hex View-1)

Hex View-1

905F05B0	AB AB AB AB AB AB AB AB	00 00 00 00 00 00 00 00	骓..骓..骓..骓..
905F05C0	69 51 9F C0 14 93 00 18	00 66 6C 61 67 7B 72 65	IQ旗.....flag{re
905F05D0	76 65 72 73 69 6E 67 5F	69 73 5F 6E 74 5F 74	versing_is_not_t
905F05E0	68 61 74 5F 68 61 72 64	21 7D 00 00 00 AB AB AB	hat_hard!})..骓..
905F05F0	AB AB AB AB AB 00 00 00	00 00 00 00 00 00 00 00	骓..骓..
905F0600	5D 50 9C F6 3F 93 00 00	C0 00 5F 00 C0 00 5F 00	JP淹?..
905F0610	EE FE EE FE EE FE	EE FE EE FE EE FE EE FE	鉛..鉛..鉛..鉛..鉛..鉛..
905F0620	EE FE EE FE EE FE	EE FE EE FE EE FE EE FE	鉛..鉛..鉛..鉛..鉛..鉛..鉛..

UNKNOWN 005F05CF - dehu8029-005F05CF (Synchronized with RTP - TDA View-RTP)