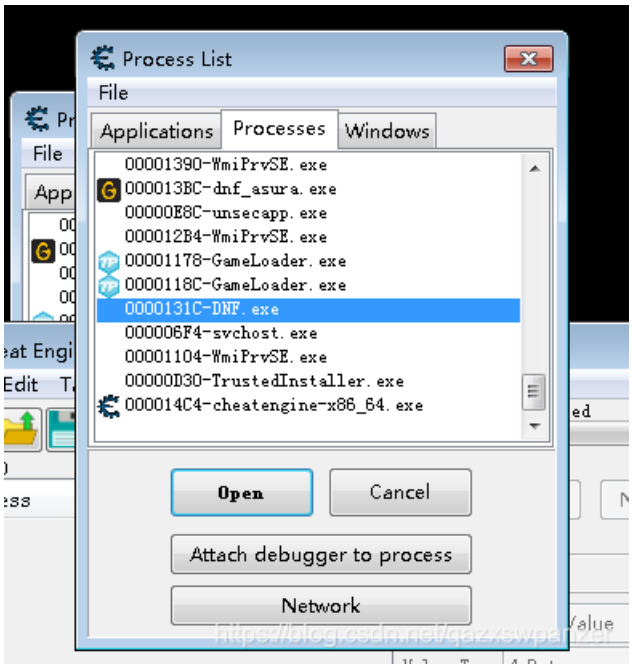
【驱动保护】某P的内存降权

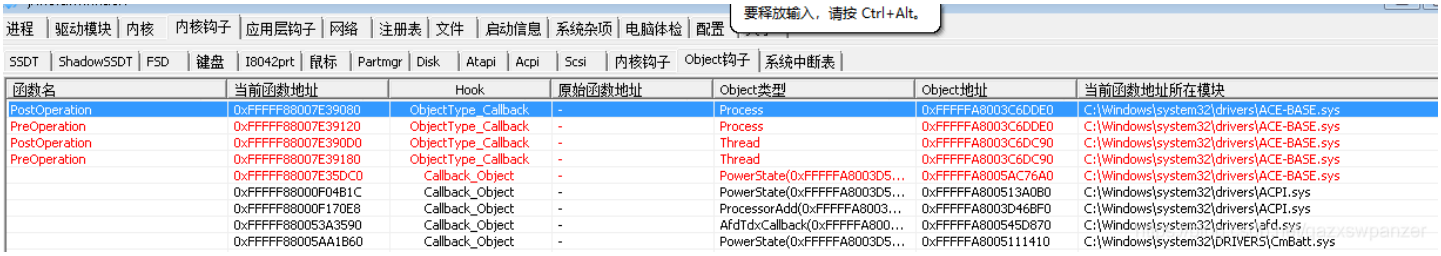
原创

Kasugano-Sora 于 2020-10-12 18:43:20 发布 782 收藏 9

上游戏打开CE发现在CE中对游戏没有内存操作权限（没有图标）



打开PChunter查看Object钩子 发现某P有对进程句柄权限回调的钩子



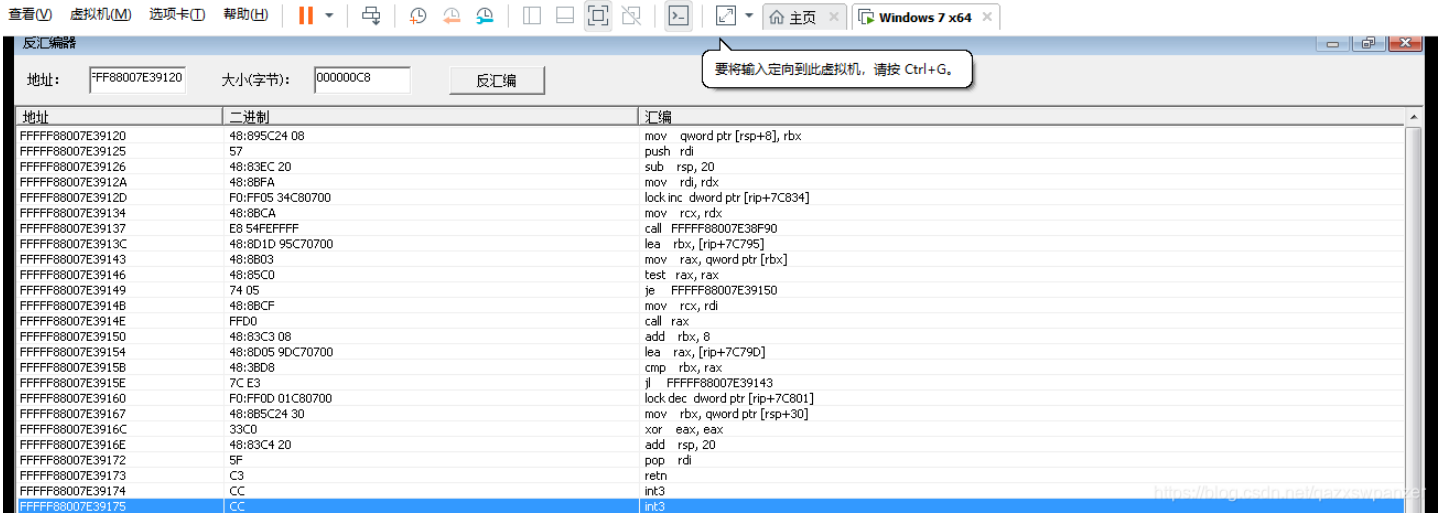
方式1：直接摘除回调

结果：黑屏重启

方式2：回调入口处写ret

结果：黑屏重启

看样子都有保护 普通的方法肯定是不行了 跟进去看一下回调函数的情况吧



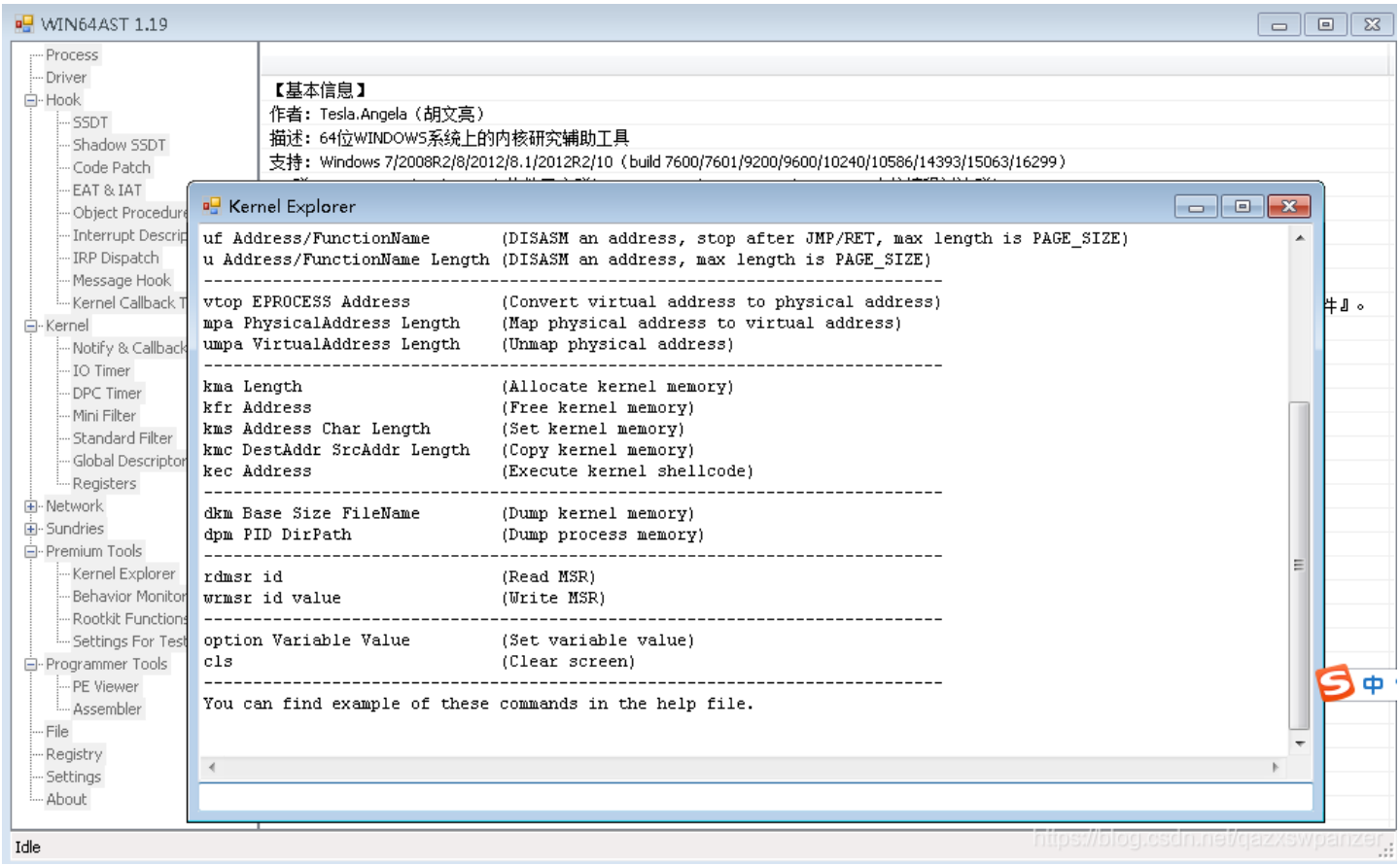
函数非常简单 观察它的汇编代码我们发现 它其实是循环调用某一个表里面的函数

```
表头
FFFFF88007E3913C  48:8D1D 95C70700  lea    rbx, [rip+7C795]
FFFFF88007E3913C+7C795=FFFFF88007EB58D1
表尾
FFFFF88007E39154  48:8D05 9DC70700  lea    rax, [rip+7C79D]
FFFFF88007E39154+7C79D=FFFFF88007EB58F1
这个表最多能保存(FFFFF88007EB58F1-FFFFF88007EB58D1)/8=4 个函数
```

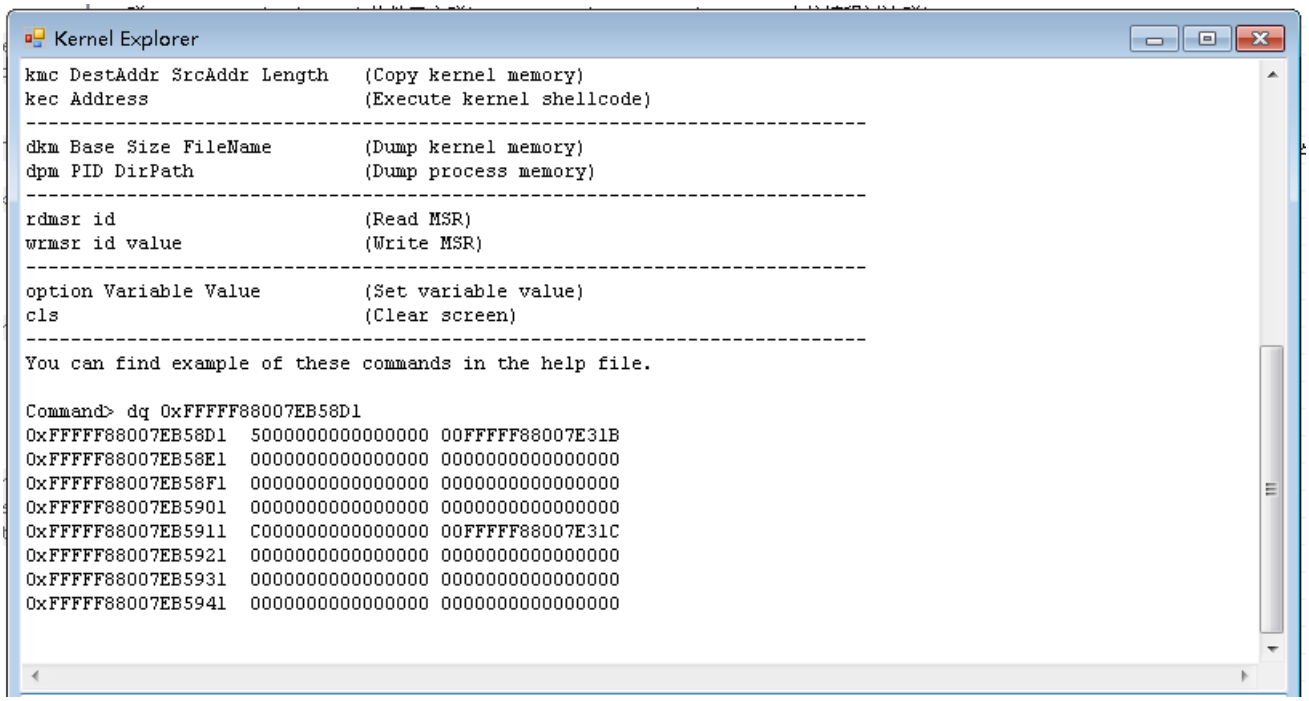
好了 现在第三种思路来了 我们来清空这个表

打开Win64AST

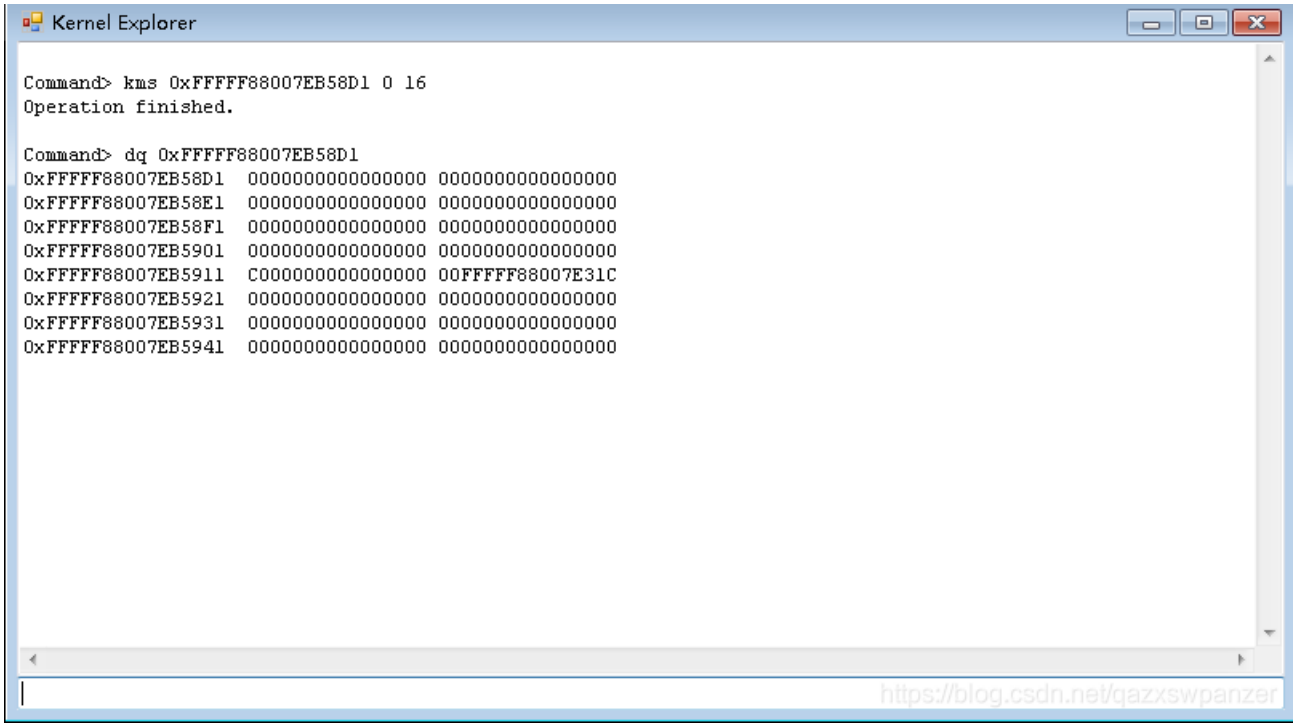
使用它自带的Kernel Explorer来编辑内核内存



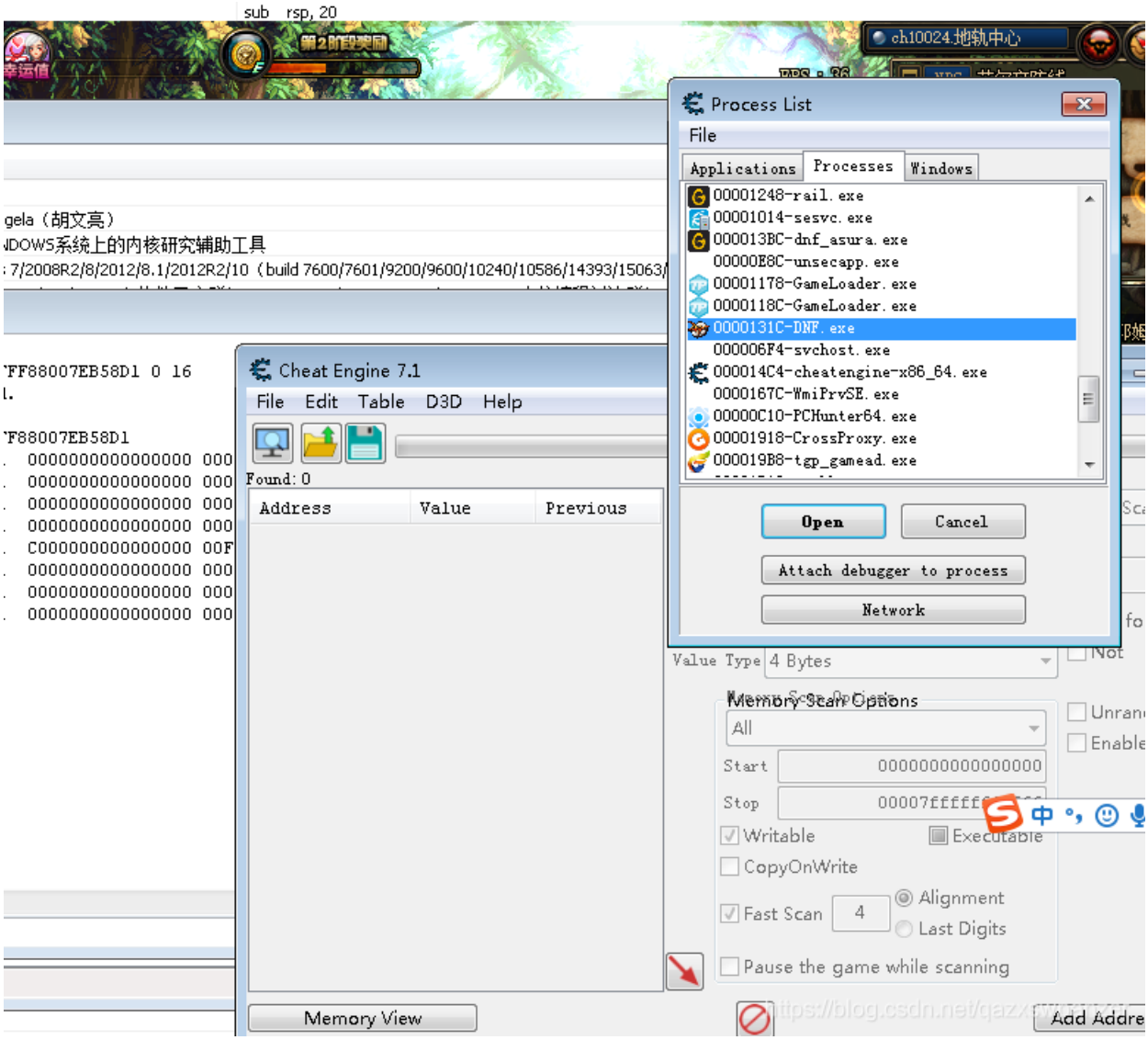
输入dq FFFFF88007EB58D1查看表头情况



有16字节的数据 我们输入kms指令对其清空



表已经成功清空 虚拟机也没有黑屏重启 我们看一下CE的情况



CE已经能正常看到dnf的ico图标
尝试读取一下内存 但是 ce内的数据仅仅存在一段时间 然后数据变为“??”
打开YDArk
经过研究发现CE仅仅在打开进程几秒内有权限 几秒后权限被降权

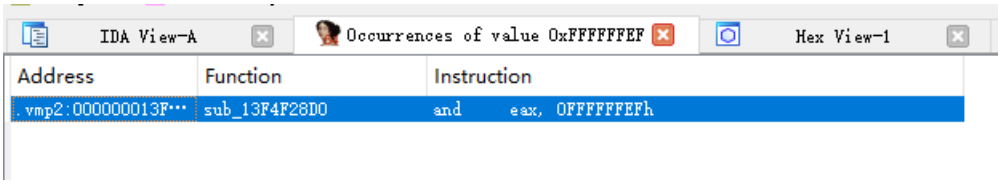
1FFFFF	Process	Pid:5316 C:\Program Files\Cheat Engine 7.1\cheatengine-...	000340	0xFFFFFA8008073650	7
1FFFFF	Process	Pid:4892 E:\WeGameApps\地下城与勇士\DNF.exe	000530	0xFFFFFA8006553B00	7
1F0001	Mutant		000054	0xFFFFFA800807F100	14

1FFFC5	Process	Pid:4892 E:\WeGameApps\地下城与勇士\DNF.exe	000530	0xFFFFFA8006553B00	7

方式3：清零回调表
结果：读写权限仅存在几秒 几秒后被降权
曾在看雪论坛看到过一篇帖子
[原创]某南极动物厂新版XX分析——系统线程部分（新瓶旧酒）
里面有这样一段内容

> AllProcessCommonCheckThread：
>
> ```
> if ( a9 )
> {
>   if ( UtilGetObjectType(a5) == MEMORY[0xFFFFF8000A04390] )
>   {
>     Irp = PsLookupByPID(pid);
>     if ( Irp )
>     {
>       ObfDereferenceObject(Irp);
>       if ( !IsGameProcess((__int64)Irp) )
>       {
>         if ( IsGameProcess(a5) )
>         {
>           granted_access = 0x12;
>           if ( !IsPrivTokenProcess(pid) )
>             granted_access = 0x3A;
>           if ( pid == (void *)get_lsass_pid() )
>             granted_access &= 0xFFFFFFEF;
>           *(_DWORD *)(v13 + 8) &= ~granted_access;
>         }
>       }
> ```
>
> 对PID为16~0x40000的所有进程枚举句柄表
> 如果句柄ObjectType==*PsProcessType且 拥有该句柄的进程不是受保护的游戏进程 且 进程句柄指向的进程是受保护的游戏进程 则进行句柄降权

看样子就是某P的线程搞的鬼 我们直接dump下来驱动内存 拉进IDA里面分析
根据帖子的内容我们搜索立即数0xFFFFFFEF



得到一个结果

```
.vmp2:000000013F4F29C0                call    sub_13F4FB860
.vmp2:000000013F4F29C5                cmp     [rsp+48h+arg_0], rax
.vmp2:000000013F4F29CA                jnz     short loc_13F4F29D7
.vmp2:000000013F4F29CC                mov     eax, [rsp+48h+var_28]
.vmp2:000000013F4F29D0                and     eax, 0FFFFFFEFh
.vmp2:000000013F4F29D3                mov     [rsp+48h+var_28], eax
.vmp2:000000013F4F29D7
.vmp2:000000013F4F29D7 loc_13F4F29D7:                         ; CODE XREF: sub_13F4F28D0+FA↑j
.vmp2:000000013F4F29D7                mov     eax, [rsp+48h+var_28]
.vmp2:000000013F4F29DB                not     eax
.vmp2:000000013F4F29DD                mov     rcx, [rsp+48h+arg_10]
.vmp2:000000013F4F29E2                mov     ecx, [rcx+8]
.vmp2:000000013F4F29E5                and     ecx, eax
.vmp2:000000013F4F29E7                mov     eax, ecx
.vmp2:000000013F4F29E9                mov     rcx, [rsp+48h+arg_10]
.vmp2:000000013F4F29EE                mov     [rcx+8], eax
.vmp2:000000013F4F29F1
.vmp2:000000013F4F29F1 loc_13F4F29F1:                         ; CODE XREF: sub_13F4F28D0+CA↑j
.vmp2:000000013F4F29F1                xor     eax, eax
.vmp2:000000013F4F29F3                test    eax, eax
.vmp2:000000013F4F29F5                jnz     loc_13F4F28E8
.vmp2:000000013F4F29FB
.vmp2:000000013F4F29FB loc_13F4F29FB:                         ; CODE XREF: sub_13F4F28D0:loc_13F4F291E↑j
.vmp2:000000013F4F29FB                                        ; sub_13F4F28D0+73↑j ...
.vmp2:000000013F4F29FB                mov     al, 1
.vmp2:000000013F4F29FD                add     rsp, 48h
.vmp2:000000013F4F2A01                retn


下面这一句就是权限重新赋值的  我们直接NOP掉
.vmp2:000000013F4F29EE                mov     [rcx+8], eax
```

后经过测试CE能正常读写

方式4：清零回调表+Patch

结果：无黑屏重启 CE正常读写