# 一、笔记

### CPL,DPL和RPL

提权要CS和SS都提才作数。提权成功的标志是CS和SS的DPL都变成0

DPL是段描述符里的两个位，表示这个段的特权级；

CPL是当前特权级别，他是CS寄存器对应的段描述符的DPL；

RPL是请求特权级别，它是写在指令里的，如 mov dword ptr ds:[123],eax里面的ds的值的低二位，又如 jmp far 48:0x12345678 里的48的低二位。

对于普通数据段DS,ES，RPL没影响，可以乱给；

对于堆栈段SS，必须满足 RPL==CPL==DPL，否则异常；

对于代码段，必须满足CPL==DPL，否则异常。

### 跨段跳转不提权

jmp far 和 call far是用于跨段跳转的指令，其中jmp far无法提权，因为它无法同时修改CS和SS。

以 jmp far 48:0x12345678 为例，它是跳转到0x12345678，并把cs修改为0x48

call far 除了跳转以外，还会依次把旧CS和返回地址压栈。返回时用 retf 指令，跳转到返回地址并将CS恢复成旧值。

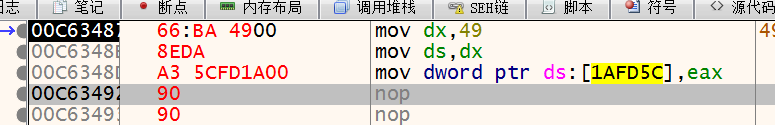
# 作业

### 1.数据段与代码段，堆栈段 权限规则总结，请动手后在总结

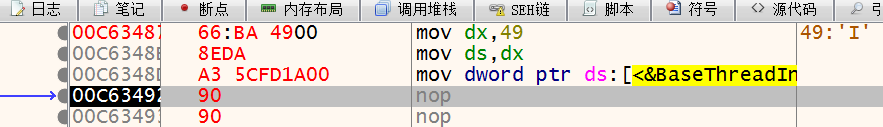
**论证1：对于数据段，rpl可以乱给**



拷贝DS到48



RPL乱给一个1，看看会咋样



此时CPL==3，DPL==3，RPL==1，RPL明显是乱给的，但是没报错，证必。

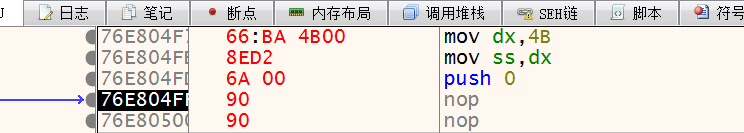
**论证2：对于堆栈段，rpl，cpl，dpl必须相等**

先看看三者相等的情况

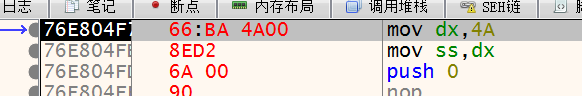
由于刚才已经把DS拷贝到48处，DS和SS是相等的（都是23），就不用再eq了。



此时，CPL==DPL==RPL==3



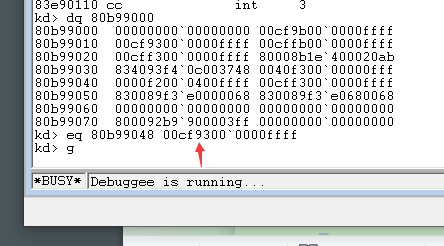
执行正常



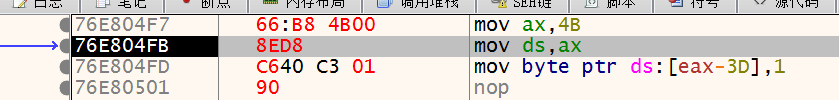
把RPL改成2试试，发现执行到mov ss,dx的时候就异常了，和预想的执行到push时才崩有出入。查资料发现这是操作系统漏洞。https://www.anquanke.com/post/id/145163?from=groupmessage

**论证3：对于代码段，只需cpl==dpl**

这里就不测CPL==DPL了，因为正常的指令都是这样的。做一个实验，在CPL==3的情况下访问DPL=0的段。



把48处的DPL改成0了，CPL还是3



执行到 mov ds,ax 时异常，原因不明。

### 2.JMP跨段跳转，CALL跨段跳转 RETF指令对堆栈产生的变化理解

答：JMP FAR 不会改堆栈；CALL FAR 会依次把旧CS和返回地址压栈；RETF会把[ESP]作为返回地址，用[ESP+4]恢复CS。

### JMP跨段跳转到新的段，顺利返回，并且CS要恢复调回的原样

void \_\_declspec(naked) test()

{

\_\_asm

{

ret;

}

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

printf("%x\r\n",test);

char bufcode[6]={0,0,0,0,0x48,0};

\*(int \*)&bufcode[0]=(int )test;

\_\_asm

{

push heihei

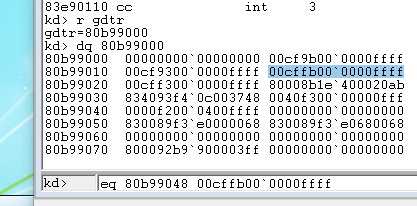
jmp far bufcode

heihei:

}

return 0;

}



拷贝CS到48处

我的方法是提前把cs压栈，然后jmp far，返回就用retf

