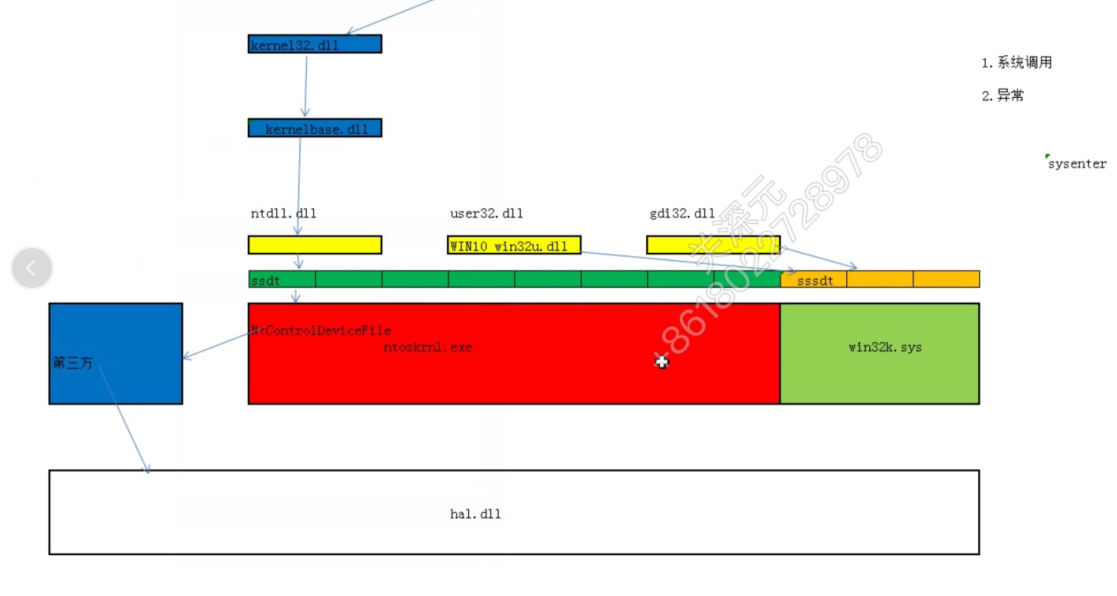
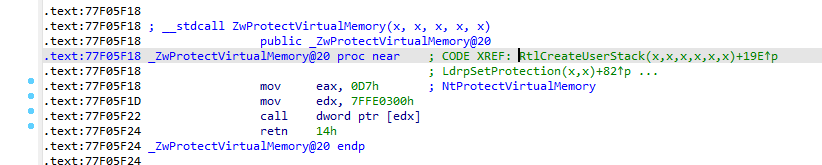
# 系统调用R3分析

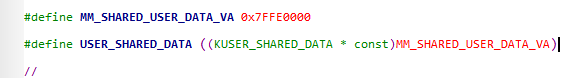
下面以win7x86的VirtualProtect为例说明。这个函数在kernel32和kernelbase这俩dll里有定义，但是没啥意思，反正最终都会去到ntdll的ZwProtectVirtualMemory.



ZwProtectVirtualMemory 函数长这样

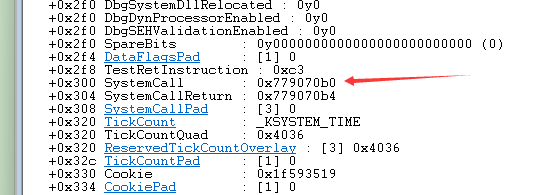


把系统调用号 D7h 存到eax，把 systemcall 这个函数指针存到edx并且调用它，为什么说 7ffe0300存的是systemcall函数呢？



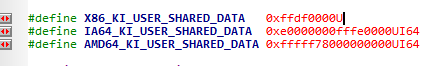
从WRK中可以看到，7ffe0000是一个叫 KUSER\_SHARED\_DATA 的结构体，去windbg看看它+300h的偏移是什么东西：

dt \_KUSER\_SHARED\_DATA 7ffe0000

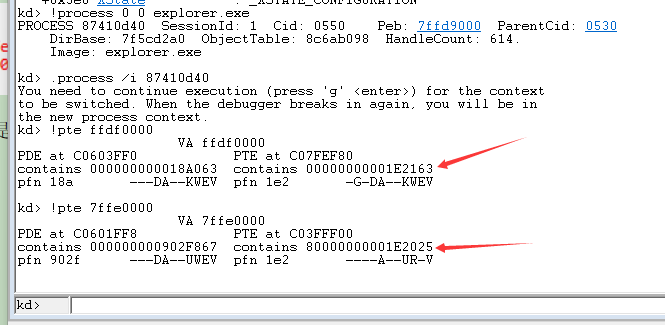


可以看到 +300h 是一个叫SystemCall 的函数。

为什么叫shared data呢，因为在内核里有一个虚拟地址和R3的7ffe0000映射了同一个物理页

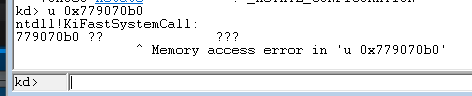


用windbg随便附加一个进程，!pte 看看 ffdf0000 和 7ffe0000 是不是同一个物理地址？

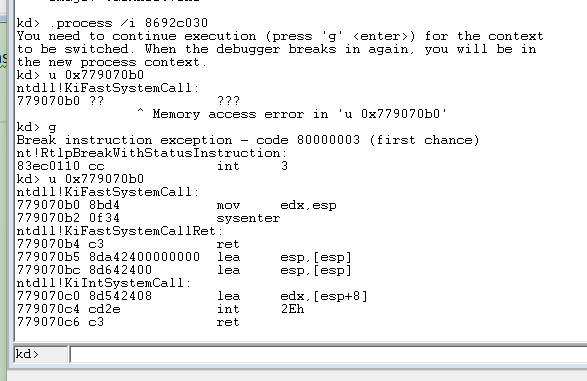


除了低12位属性不同，和最高位NX位之外，都是相同的，所以叫shared data。

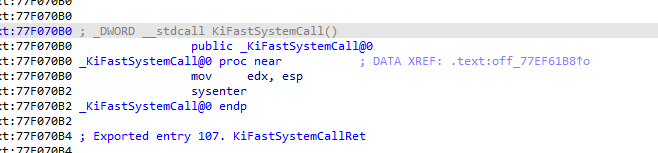
回到这个 ZwProtectVirtualMemory 函数，他调用了 SystemCall 函数，我们可以u一下看看是啥玩意：



好像是缺页了，要这样：



先附加一个进程，然后g一下让他切换上下文，然后会再次断下，这时候u才有东西，可以看到这个函数名叫 KiFastSystemCall，去ntdll看看：



把R3的栈顶ESP存到EDX，然后调用sysenter指令快速调用进R0了。

到这里为止，我们知道，eax存的是系统调用号，edx存的是R3栈顶，这样进到R0之后，就可以知道我们想调用的函数，以及R3的参数了。

# 实现VirtualProtect

|  |
| --- |
| #include <windows.h>  \_\_declspec(naked) void MyKiFastSystemCall()  {  \_\_asm  {  mov edx, esp;  \_emit 0x0F;  \_emit 0x34;  }  }  \_\_declspec(naked) NTSTATUS \_\_stdcall MyNtProtectVirtualMemory(HANDLE ProcessHandle, PVOID\* BaseAddress, SIZE\_T\* NumberOfBytesToProtect, ULONG NewAccessProtection, PULONG OldAccessProtection)  {  \_\_asm  {  mov eax, 0xD7; 我的win7 NtProtectVirtualMemory 系统调用号  mov edx, 7FFE0300h;  call dword ptr[edx];  retn 0x14;  }  }  BOOL \_\_stdcall MyVirtualProtectEx(HANDLE hProcess, LPVOID lpAddress, SIZE\_T dwSize, DWORD flNewProtect, PDWORD lpflOldProtect)  {  MyNtProtectVirtualMemory(hProcess, &lpAddress, &dwSize, flNewProtect, lpflOldProtect);  return 0;  }  int main()  {  char bufcode[] =  {  0x6A,0,  0x6A,0,  0x6A,0,  0x6A,0,  0xb8,0,0,0,0,  0xff,0xd0,  0xc3,  };  \*(int\*)&bufcode[9] = (ULONG)MessageBoxA;  DWORD OldProtect;  MyVirtualProtectEx((HANDLE)0xFFFFFFFF, bufcode, sizeof(bufcode), PAGE\_EXECUTE\_READWRITE, &OldProtect);    //VirtualProtectEx((HANDLE)0xFFFFFFFF, bufcode, sizeof(bufcode), PAGE\_EXECUTE\_READWRITE, &OldProtect);  CreateThread(0, 0, (LPTHREAD\_START\_ROUTINE)&bufcode, 0, 0, 0);  system("pause");  return 0;  } |

