64位inline api hook

说起Api hook朋友们都不陌生，说白了也就是运用WINDOWS系统钩子，捕获程序对系统API的调用，从面改变API默认的处理方式，以达到某种特殊的目的。而inline api hook在32位系统中通常是先保存API函数入口处的5个字节，然后改变API函数入口处的5个字节为一个JMP xx xx xx xx跳转指令，xx xx xx xx处是我们自己的处理函数的偏移地址。在我们的处理函数中，进行一系列动作，达到我们的目的后，再把所保存的5个字节还原到API函数入口处，然后进行调用，以实现API函数原有的功能。

那么在64位系统中能不能实现inline api hook呢？答案是肯定的，但实现的方法有所不同。因为64位系统中API函数的地址和我们的处理函数地址之间的偏移量会大于2GB，而jmp指令最远只能跳2GB，所以无法通过改前5个字节的方式实现，我们只能通过其他的方式来达到目的。

通过实践，证明可以通过两种方法很容易地实现：一是先保存API函数入口处的12个字节，然后把API入口处改为48H B8H XX XX XX XX XX XX XX XX 50H C3H,其中XX XX XX XX XX XX XX XX为我们的处理函数的64位地址，这12个字节相当于三条指令

mov rax,XX XX XX XX XX XX XX XX

PUSH RAX

RET

以实现到我们的处理函数的跳转;

二是先保存API函数入口处的14个字节，然后改为FFH,25H,0,0,0,0,XX XX XX XX XX XX XX XX，XX XX XX XX XX XX XX XX处为处理函数地址，等效于指令jmp [rip + XX XX XX XX XX XX XX XX]。

在网上，有人说可以把前14个字节改为FFH,15H,0,0,0,0 XX XX XX XX XX XX XX XX，等效为call [rip + XX XX XX XX XX XX XX XX]，这样应该是可以的，但call指令要把下一指令的地址压栈，这样就会改变rsp的值，在我们的处理函数中就要涉及到堆栈的调整，增加程序的复杂程度。而且我们不能保证API函数入口第15个字节开始处正好是一条有效的指令。