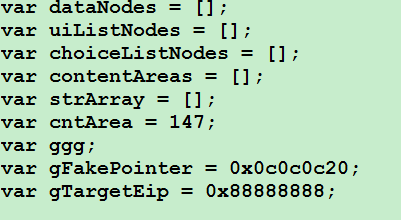
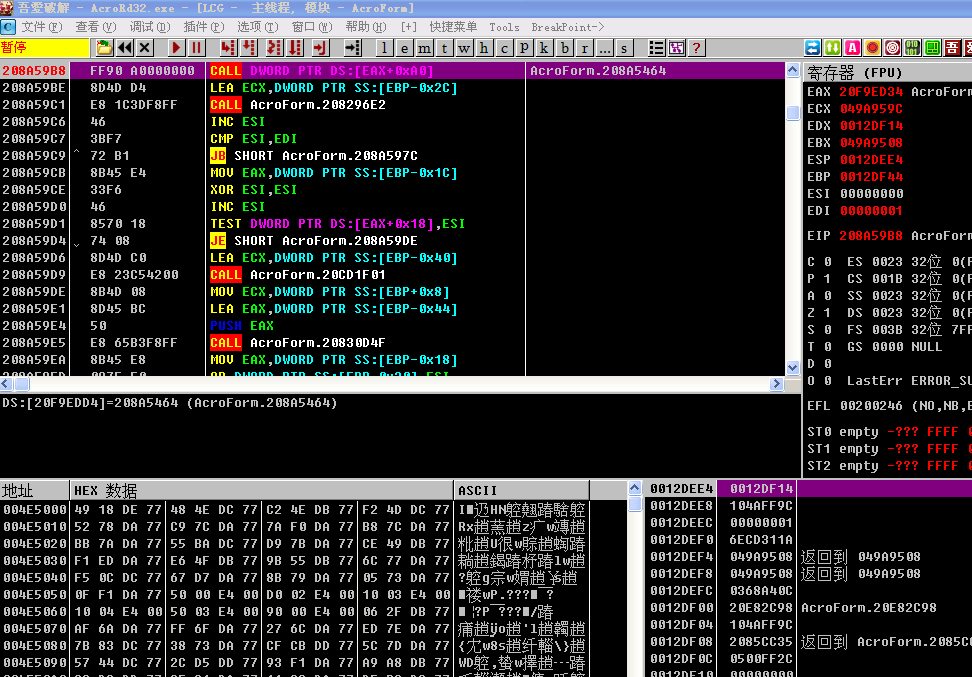
这个漏洞下断需要开启OD的中断于加载新模块。（因为在异常处的地址在adroform.dll里（其实在调试的时候，这个adroform不是一个dll而是一个.api文件？这个我不是很明白），经过试验，想要在这个模块里下断点，就必须要通过断与加载新模块）

后来我把API文件提取出来，发现他其实就是个MZ文件，所以直接可以拖入IDA分析。

直接打开poc后，来到异常88888888处（这个数值在对象6中），



栈回溯，发现指令call 【eax】，eax=0c0c0c0c，还要往前再回溯。

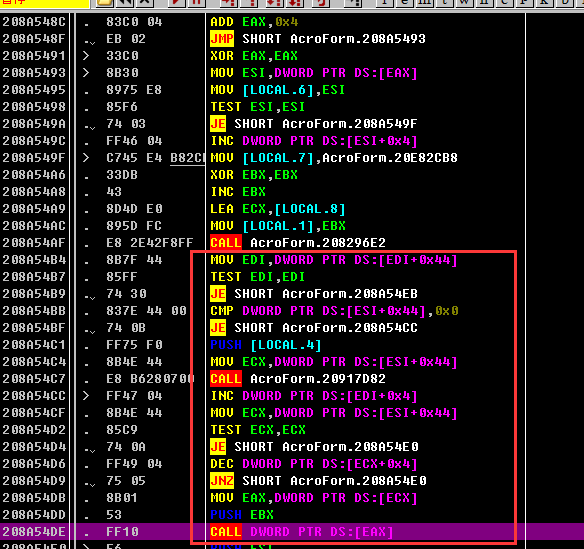


单步步入步过后没发现什么，后来发现要会多次断在这里，经过调试

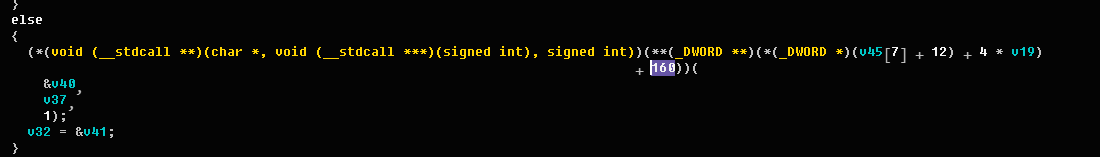
5次之后崩溃，所以重头再投，断4次后单步分析。（要在程序弹出错误的对话框后，继续F9 4次）

4次断下后，要F7单步进去，异常发生在这个函数里面。

其实漏洞就发生在这里。（我没有按照分析文章的过程分析。而是想直接定位漏洞异常点后，ida分析这个函数）



这里ida F5后出来



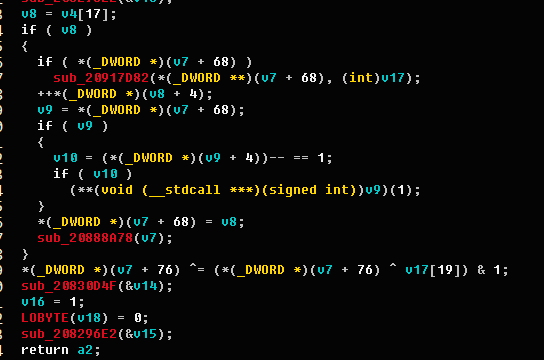
所以直接IDA跳转到

C:\Users\Administrator\AppData\Roaming\Tencent\Users\781720849\QQ\WinTemp\RichOle\C_C5WYIKBC$2{03PEK[9Q~Q.png

这个函数的入口处。



关键点在这里：



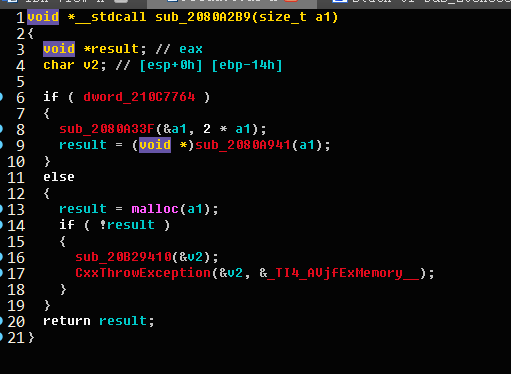
通过v7找到v6，v6=（int \*）v5+4

又因为v5 = sub\_208A551F(this, (int)&v12, a3, a4);,

所以进入208a551f这个函数。再进入553f

（下面需要逆向出来逻辑：就是 a3被分配的内存大小为0x40，所以v7+0x44，也就是上图的v7+68是非法的内存地址。）

深入这个函数会发现其实这个函数在malloc



所以这里会发现有逻辑错误，访问到了未知的地址，（其实我很好奇，这个程序在编写的时候怎么会造成这种错误的？是程序员的错？）

因为我之前看了分析文章，所以知道程序内部的流程是分配了0x40的内存，然后访问了0x44非法区域，

正常流程还应该要逆向，比如说可以动态调试那个v5的值



F8步过，直接看返回值是什么，如果是malloc，应该是返回一个句柄

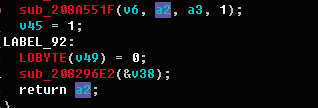
如果还是看不出什么，需要进入这个函数继续动态调试，

比如说

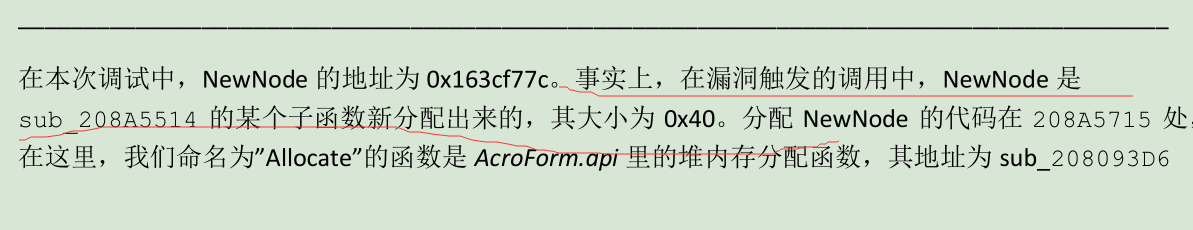


因为对a2参数做了修改，所以要看a2被改成了什么。（553f这个函数的a2是一个地址，所以作为参数传入肯定会被修改。）

结果发现只有一处修改了a2，

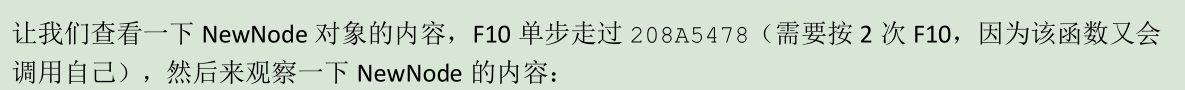


但是会发现这个函数又调用了一遍自己。

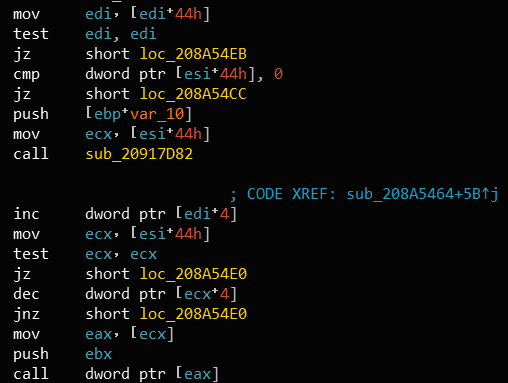
（其实后面我也不清楚要怎么跟a3，也就是分配了0x40的地方扯上关系，逻辑问题还需要逆向还原），在分析文章中也没有很详细地说明是如何分析出来的

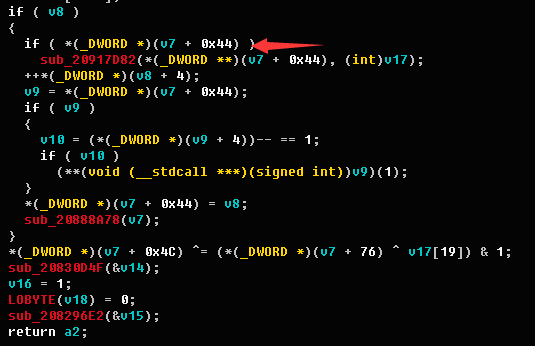
（其实208a551f这个函数是这个程序自己的堆管理函数，漏洞利用的时候会提到）

（这和之前的逆向过程相符，就是调用了2次自己

）

按照上面的思路，就应该发现了问题（这种漏洞感觉如果是自己调试的话很难发现吧…）





而且这个漏洞会触发两次

理解下这个漏洞，产生的效果就是：

将某个[[未定义地址]+4]处的值减 2（每次减 1），但是如果原始值小于等于 2 时，[[未定义地址]]处的指针将被作为函数地址调用。

在图中可以看到v10在做判断，v9作为函数被调用了。

————————————————————————————————

接下来就是漏洞利用了

其实这里之前V5=208a551f（）这个函数其实是一个私有堆管理函数，逆向后会发现这个堆管理函数很像windows堆管理（逆向这个堆管理没有明说），需要自己逆。这个函数好复杂，没逆。

利用这个堆管理来控制EIP，这个过程我草草地看了一下，好像有点类似《0day2》：重重保护下的堆\_\_利用chunk重设大小攻击堆 这个方法。