**Flash 0Day CVE-2018-4878漏洞利用原理分析**

**0x00 调试环境**

|  |  |
| --- | --- |
| OS: | win7 x86 |
| IE: | ie10 |
| Flash: | Flash32\_27\_0\_0\_159.ocx |

**0x01 背景**

2018年2月4日，Freebuf刊登了360的最新的Flash 0Day的在野预警分析《Adobe Flash零日漏洞（cve-2018-4878）在野攻击预警分析》（<http://www.freebuf.com/vuls/162049.html>），但是360却未放出样本。2月8日有幸拿到样本，简单的调试下了解其原理后正准备在FB发文，却发现腾讯的大牛已经先一步发出分析文章了：《Flash 0Day漏洞（CVE-2018-4878）原理及利用代码分析》(<http://www.freebuf.com/vuls/162369.html>)，但是大牛的文章永远是那么飘逸，代码一贴，windbg一上，还没看的尽兴就这么结束了。

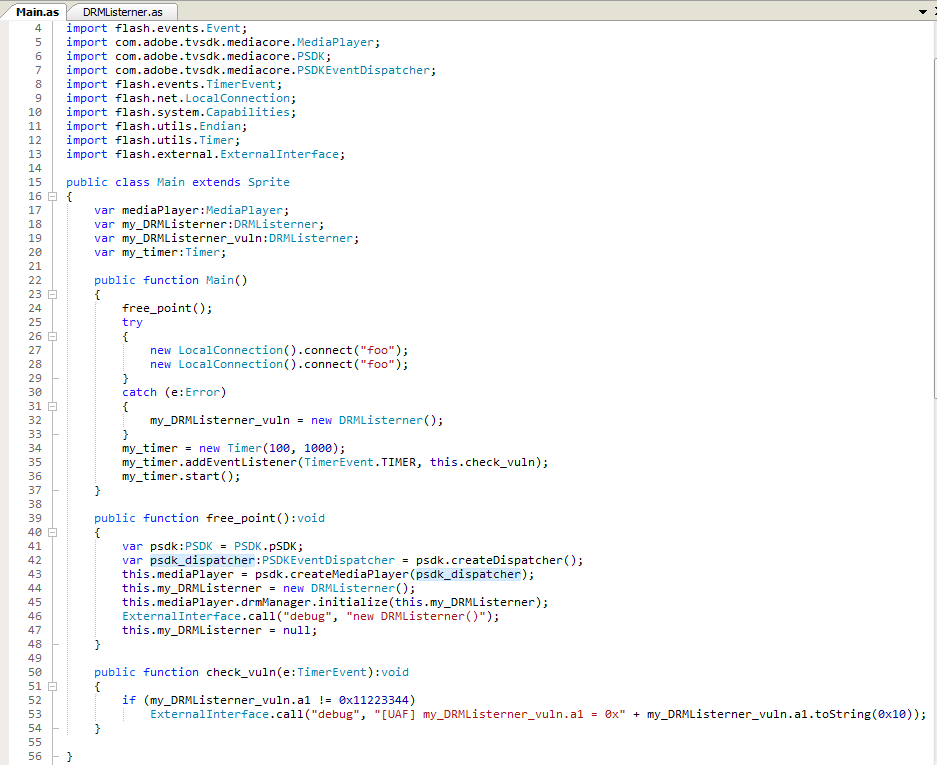
对于二进制的小白来说，样本都没有，何来调试，只能看个热闹。因此我将样本简化，做了个能够说明漏洞原理的poc，并详细附上调试方法，希望对flash漏洞利用感兴趣的同学们有所帮助。

**0x02 漏洞原理**

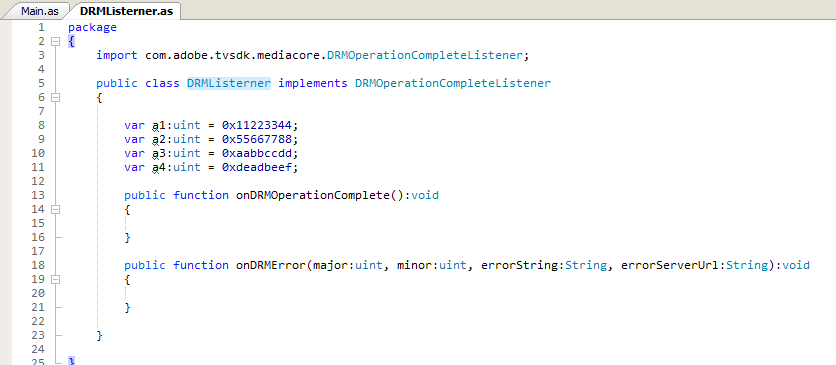
CVE-2018-4878出现在com.adobe.tvsdk包中，之前的CVE-2016-4117，CVE-2017-11292同样也出自该包，看来adobe要好好审计下这个包的代码了。

直接上poc，一共由两个文件组成：

1. main.as



2. DRMListerner.as



简单说明：

1. main.as中Main() 是入口函数，首先调用了free\_point()函数。free\_point()函数是触发漏洞的关键函数，free\_point()里首先创建了一个mediaPlayer对象和一个实现DRMOperationCompleteListener接口的DRMListerner对象，并将DRMListerner对象传递给了this.mediaPlayer.drmManager.initialize()方法，因此mediaPlayer对象中保存了DRMListerner对象的引用，最后通过this.my\_DRMListerner = null;释放了DRMListerner对象（但是mediaPlayer对象仍然中保存了DRMListerner对象的引用）；

2. 释放了DRMListerner对象后通过两次new LocalConnection().connect("foo");触发异常，在catch分支再次创建了DRMListerner对象；

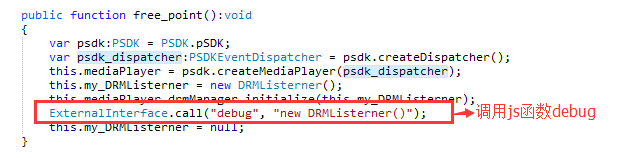
3. 创建一个定时器，绑定callback函数check\_vuln，每100ms检查一次是否获得一个悬挂指针（这里通过my\_DRMListerner\_vuln.a1 != 0x11223344 来检查，事实上如果不存在UAF漏洞的话my\_DRMListerner\_vuln.a1 == 0x11223344，如果不等，说明my\_DRMListerner\_vuln指向了一片未使用的内存，从而引起UAF漏洞）。

4. DRMListerner.as实现了DRMOperationCompleteListener接口，并且设计了a1~a4 4个成员变量来方便在内存中标记DRMListerner的数据部分。

编译as文件，保存成CVE-2018-4878.swf文件后就可以调试了。在调试之前需要介绍一些调试的小技巧：

一般来说，对flash样本最方便的调试方法就是对jitcode下断点，但是如果没有现成的jitcode下断点的工具，可能调试起来会比较困难，考虑到flash可以通过ExternalInterface.call来调用js函数，而jscipt9.dll又有符号表，因此可以这样下断点：

as脚本中：



Html文件中：

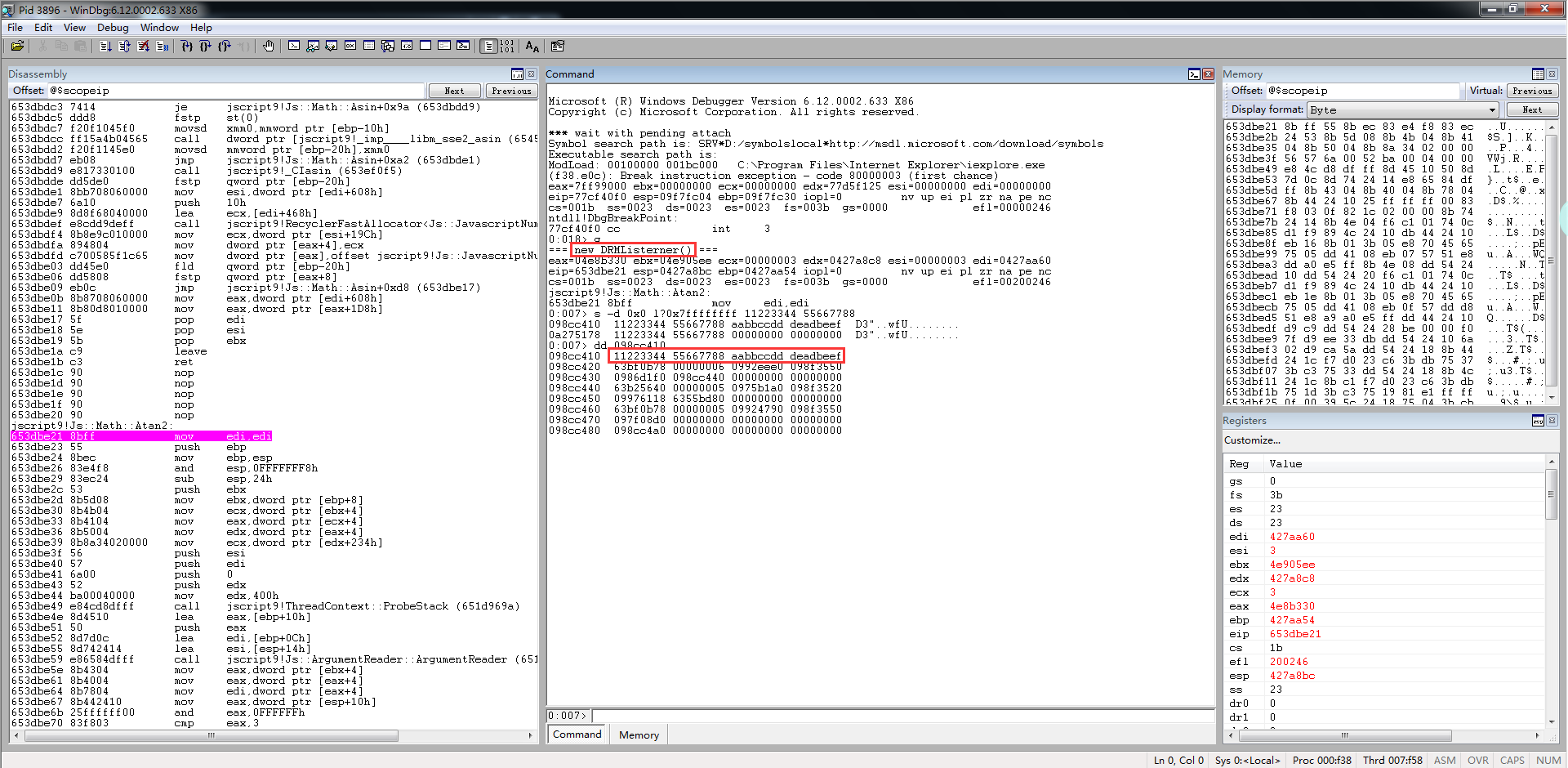


Windbg下条件断点：

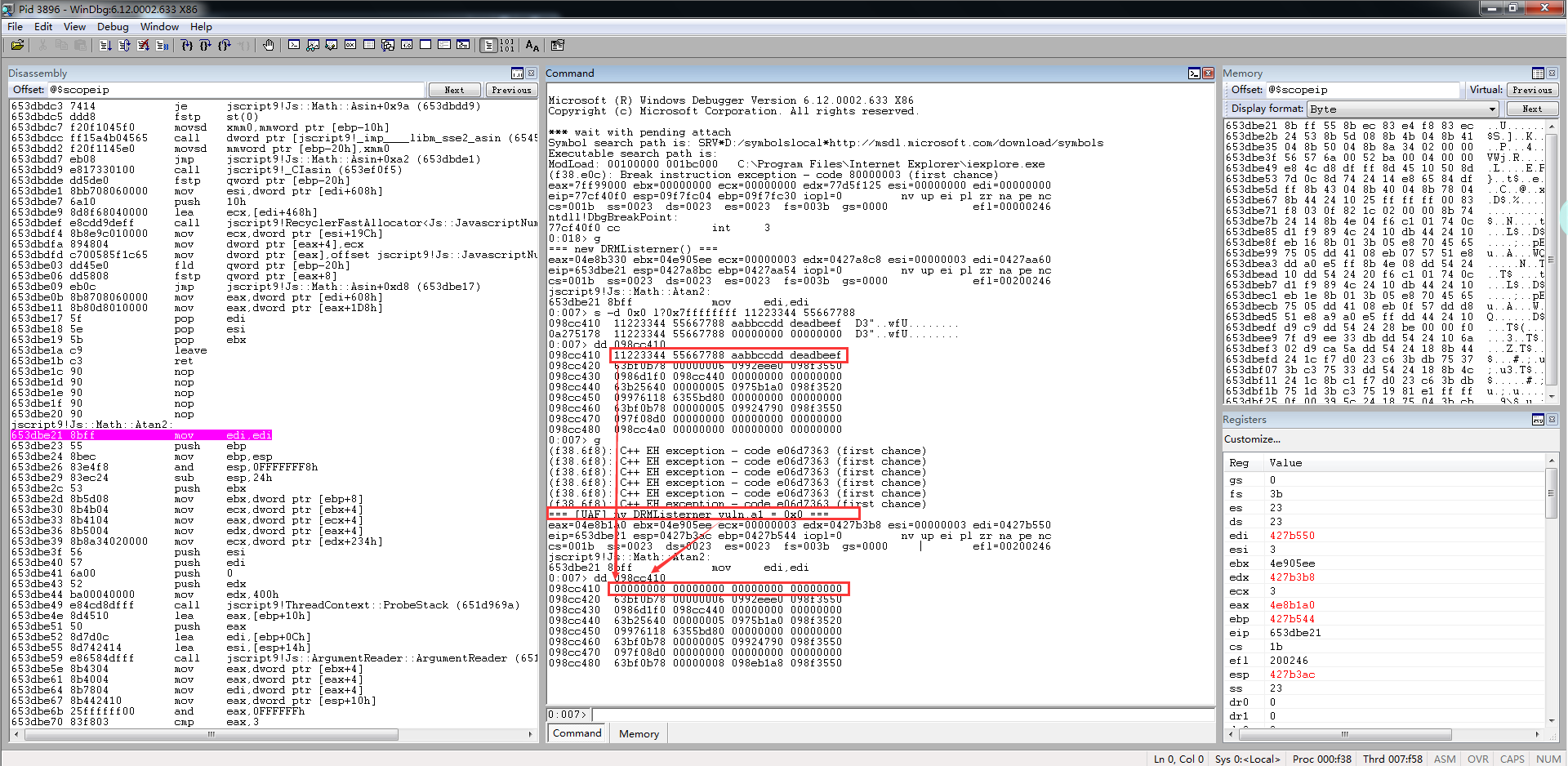
bu jscript9!Js::Math::Atan2 "j ((poi(esp+10)/2) == badc) '.printf \"=== %mu ===\",poi(poi(esp+14)+c);.echo';g"

打开IE，附加windbg，下好断点，F5跑起来，命中第一次断点：

这时DRMListerner对象已经创建，由于我们在DRMListerner对象里设置了若干可以用来定位的变量，因此通过搜索来定位DRMListerner对象：



F5继续执行，很快就进入了check\_vuln函数里的断点：

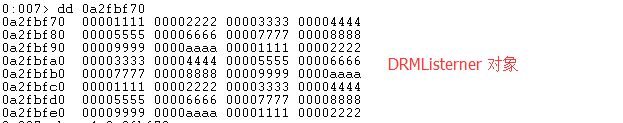


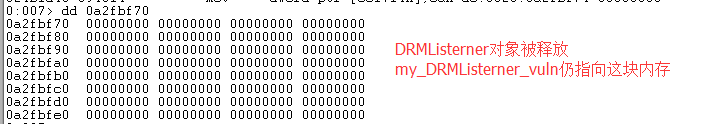
此时098cc410不再保存DRMListerner对象，但是my\_DRMListerner\_vuln仍然指向这片内存，从而引起UAF漏洞。

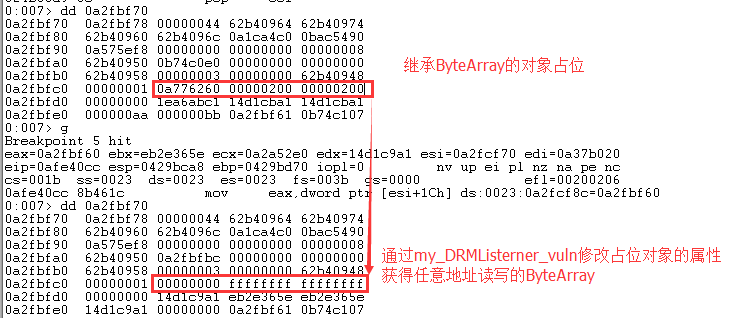
**0x03 漏洞利用**

具体的利用方法可以参考腾讯那篇文章或者是360的这篇《Adobe Flash Player漏洞（CVE-2016-4117）样本分析》（<http://www.freebuf.com/vuls/110707.html>）。

其实思路都一样，在Vector被加固后，一般都是用ByteArray来实现任意地址读写，因此在获得my\_DRMListerner\_vuln这个悬挂指针后会用一个继承ByteArray的类来占位，并通过操作my\_DRMListerner\_vuln来修改占位的ByteArray的array地址和长度分别为0x0000 0000和0xFFFF FFFF ：







获得任意地址读写的ByteArray后，后面的利用手法大同小异，不再详述。