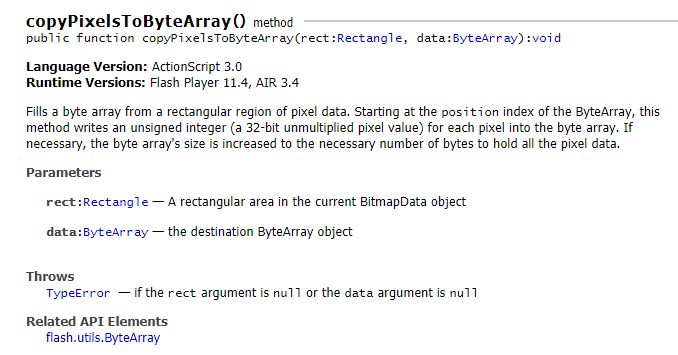
**CVE-2014-0569漏洞利用分析**

**0x00 调试环境**

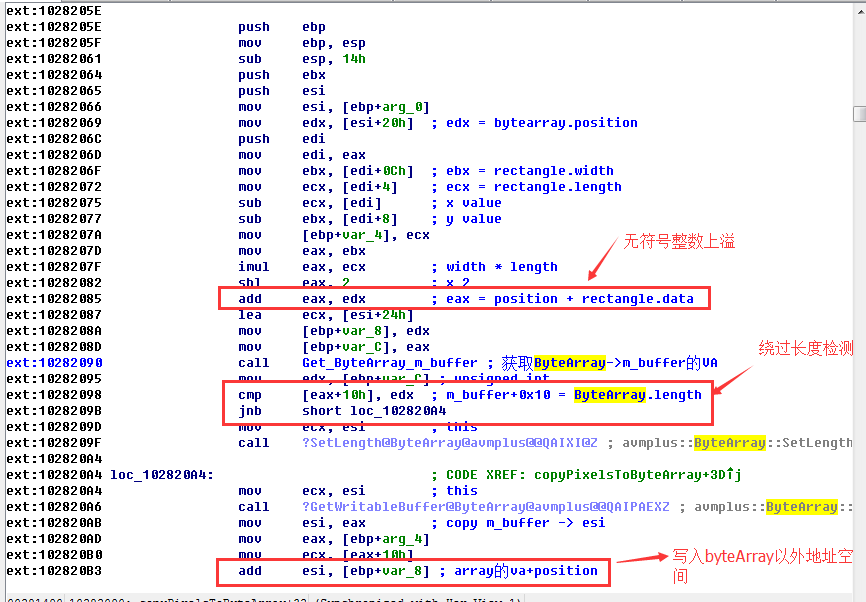
|  |  |
| --- | --- |
| OS: | win7 x86 |
| IE: | ie10 |
| Flash: | flashplayer13\_0r0\_182\_win\_debug |

**0x01 漏洞成因**

AS3的copyPixelsToByteArray()函数存在整型溢出漏洞。攻击者可以通过构造该函数的参数实现溢出攻击。copyPixelsToByteArray函数定义如下：



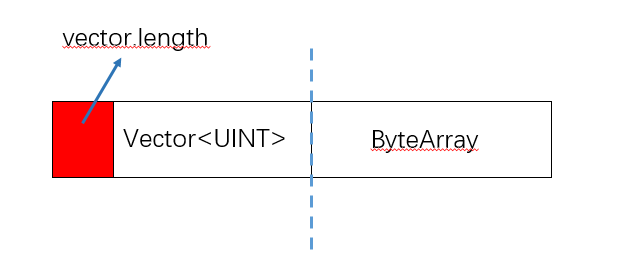
该函数将一个BitmapData的像素数据通过Rectangle指定的区域保存到ByteArray中。其逻辑如下图：



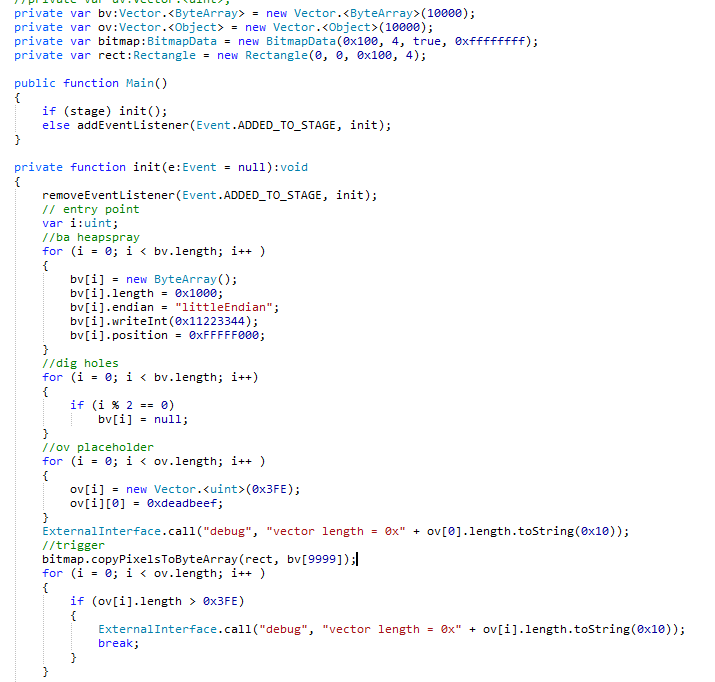
copyPixelsToByteArray()会对当前ByteArray的长度和写入的BitmapData大小进行校验，如果构造的position很大就可能与BitmapData大小相加后发生无符号整数上溢，从而绕过copyPixelsToByteArray()的长度检查，最终写入的地址=array+position，position是有符号整数的负数，从而可以向array的低地址空间写入数据。

**0x02 漏洞利用**

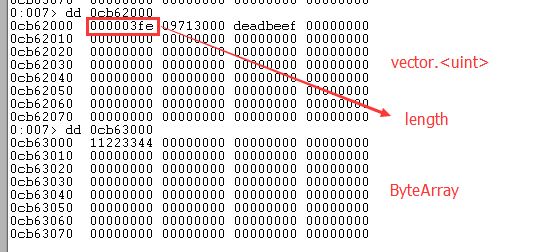
因为利用该漏洞可以向ByteArray的低地址空间写数据，所以考虑如下内存布局：



具体利用代码：



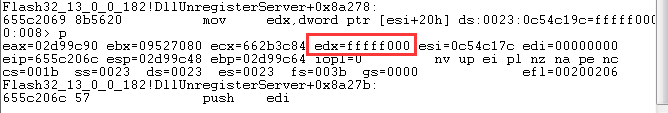
这里先申请了10000个ByteArray占位，然后间隔释放ByteArray，留出4kB的内存空洞，接着用同样大小的Vector.<uint>占位，此时的内存布局：



进入copyPixelsToByteArray()关键步骤调试情况：

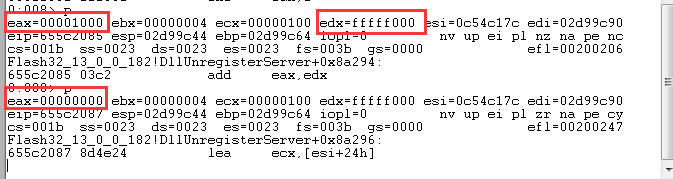
1. .text:10282069 mov edx, [esi+20h]

保存ByteArray的position至edx



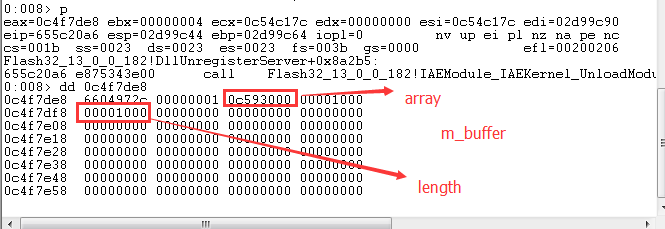
2. .text:10282085 add eax, edx

将ByteArray的position和写入BitmapData的大小相加，这里发生了整数溢出（FFFF F000 + 1000 = 0）



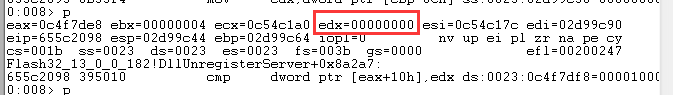
3 .text:10282090 call Get\_ByteArray\_m\_buffer

获取ByteArray的m\_buffer的VA



4. .text:10282098 cmp [eax+10h], edx

将步骤2相加的结果和ByteArray的length（m\_buffer+0x10处）进行长度校验，因为步骤2发生了整数溢出，从而绕过了步骤5的验证

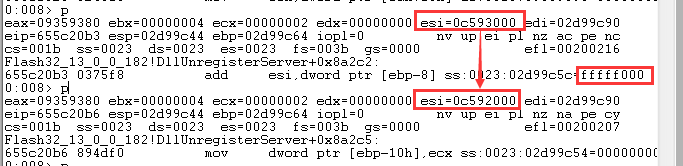


5. text:1028209B jnb short loc\_102820A

整数溢出导致进入loc\_102820A分支

6. .text:102820B3 add esi, [ebp+var\_8]

计算写入ba的地址，esi保存了m\_buffer.array地址， [ebp+var\_8]是ByteArray的position = FFFF F000



这里m\_buffer.array的地址是0c593000, 0c593000 + FFFF F000 = 0c592000，即m\_buffer.array低地址偏移0x1000处。

最终获得一个长度为ffff ffff的vector.<uint>

