**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**

avm2.intrinsics.memory.casi32函数存在整数溢出漏洞。该函数是AS3脚本里用来操作domainMemory的函数。

casi32用来比较并交换domainMemory的值，其定义如下：

public function casi32 (addr:int, expectedVal:int, newVal:int) : int;

addr ——操作的内存地址，相对domainMemory的偏移地址

expectedVal ——原来的值

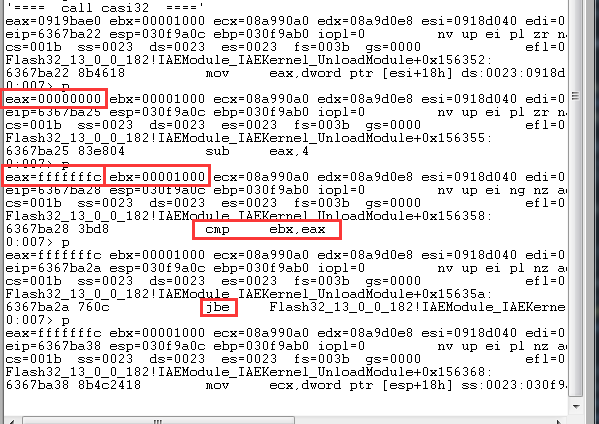
newVal ——要交换的新值

avmplus中casi32的代码在ConcurrencyGlue.cpp中：



可以看到此处为无符号比较，如果domainMemSize=0则domainMemSize - sizeof(int32\_t) = FFFF FFFC，为无符号一个很大的值，从而绕过了长度检查。

具体看windbg:



1. mov eax,dword ptr [esi+18h]

这步取出domainMemory的长度存放在eax，此时eax=0

1. sub eax,4

接着将eax-4=0-4=FFFF FFFC

1. cmp ebx,eax

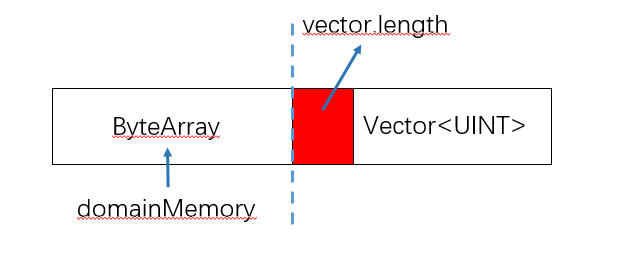
此时ebx存放casi32要写入的偏移地址0x1000，而eax=FFFF FFFC

1. jbe Flash32\_13\_0\_0\_182!IAEModule\_IAEKernel\_UnloadModule+0x156368

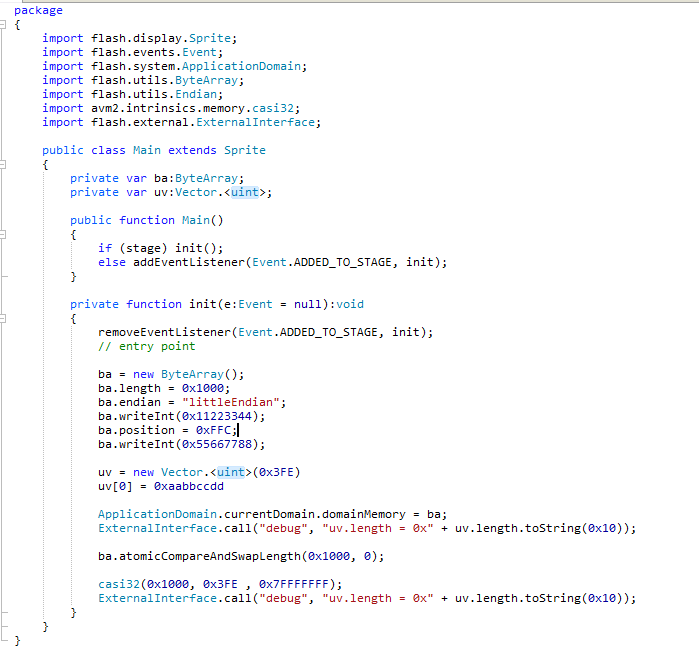
此处为无符号整数比较，ebx < eax，从而绕过了长度检测，形成整数溢出漏洞

**0x02 漏洞利用**

通过casi32的整数溢出漏洞可以修改domainMemory以外的内存空间，这里考虑如下内存布局：



domainMemory指向一个ByteArray，并紧接着申请一个Vector.<uint>，由于Vector.<uint>前四个字节存放的是该Vector.<uint>的长度，那么通过casi32的整数溢出就可以修改该Vector.<uint>的长度，从而得到一个很大的Vector.<uint>。具体代码：

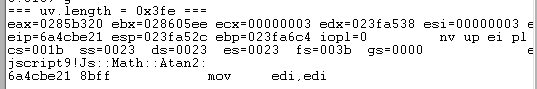


ba的大小为0x1000，而Vector.<uint>的大小为0x3FE\*4+8=0x1000。

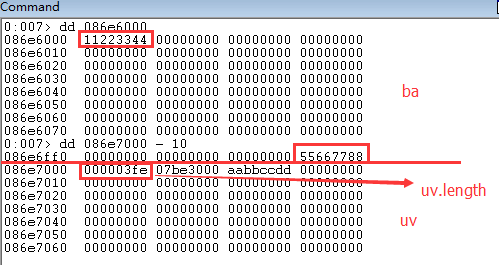
这里通过ba.atomicCompareAndSwapLength(0x1000, 0)将ba的长度设为0，注意到将ba的长度设为0并不会立即将他的内存释放，所以domainMemory依然指向这块内存。

具体调试过程：

1. 修改长度前Vector.<uint>的长度：

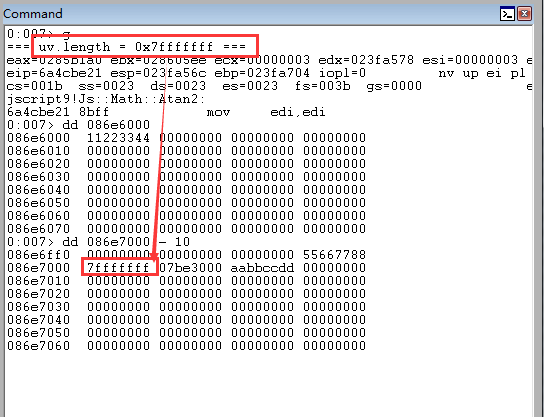


此时ba和uv的内存布局：



可以看到的ba的m\_bufferr的array后面紧挨着uv，ba偏移0x1000即为uv的长度3fe。

1. 触发漏洞，修改Vector.<uint>的长度：



此时已获得一个长度为7ffffffff的Vector.<uint>，后面的exploit手法可以参考CVE-2015-0311，不再详述。