iOS冰与火之歌(番外篇) - 基于PEGASUS（Trident三叉戟）的OS X 10.11.6本地提权

蒸米@阿里移动安全

0x00 序

这段时间最火的漏洞当属阿联酋的人权活动人士被apt攻击所使用的iOS PEGASUS（又称Trident三叉戟）0day漏洞了。为了修复该漏洞，苹果专门发布了一个iOS 9.3.5版本。这个漏洞的厉害之处在于可以直接从沙盒内对内核进行攻击(无需沙盒逃逸)，并且同时影响iOS(9.3.4)和OS X (10.11.6)。因此，本篇文章将会从PEGASUS漏洞形成的原因开始分析，然后一步一步教大家写出OS X上利用PEGASUS提权的攻击代码。

《iOS冰与火之歌》这一系列文章的目录如下：

1. Objective-C Pwn and iOS arm64 ROP

2. 在非越狱的iOS上进行App Hook（番外篇）

3. App Hook答疑以及iOS 9砸壳（番外篇）

4. 利用XPC过App沙盒

5. UAF and Kernel PWN

6. 基于PEGASUS（三叉戟）的OS X 10.11.6本地提权 (番外篇)

另外文中涉及代码可在我的github下载: https://github.com/zhengmin1989/iOS\_ICE\_AND\_FIRE

0x01 漏洞分析

PEGASUS（三叉戟）实际上是由三个漏洞组成，分别是：

CVE-2016-4657：Safari的Webkit内核上的内存破坏漏洞。

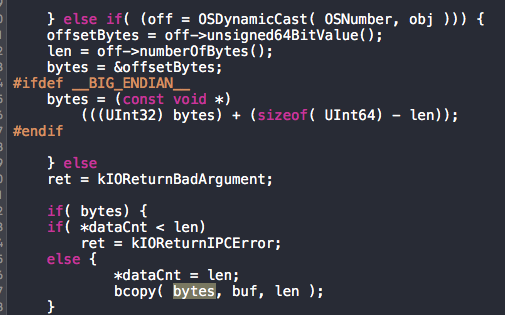
CVE-2016-4655：内核信息泄露漏洞，可用于绕过KASLR。

CVE-2016-4656：内核UAF漏洞，可用于控制内核并执行任意代码。

关于CVE-2016-4657目前还没有公开的资料，但Stefan Esser和Pangu分别爆出了另外两个漏洞CVE-2016-4655和CVE-2016-4656的细节。利用已经曝光的这两个漏洞，其实已经可以做到iOS非完美越狱和OS X本地提权了。下面我们就来讲解一下这两个漏洞形成的原因以及如何利用。

0x02 CVE-2016-4655 内核信息泄露

CVE-2016-4655这个漏洞形成的原因是内核在序列化和反序列化OSNumber的时候没有验证长度的正确性。因此，如果我们将number的长度设置的非常长，并用io\_registry\_entry\_get\_property()去获取number数据的话，就会造成内核的信息泄露。



我们知道内核栈中会保存函数的返回地址，因此我们可以利用这个返回地址去计算出内核的kslide，从而攻破kaslr的保护。

那么如何编写利用代码呢？我们先创建一个序列化后的dictionary。对内核来说，这个dictionary应该是这样的：

<dict>

<key>min</key>

<number>0x4141414141414141</number>

</dict>

但是我们对OSNumber的长度进行了修改，变成了0x200：

uint32\_t data[] = {

0x000000d3,

0x81000001,

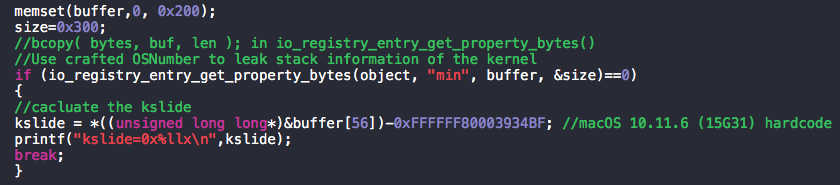
0x08000004, 0x006e696d,

0x84000200, //change the length of OSNumber

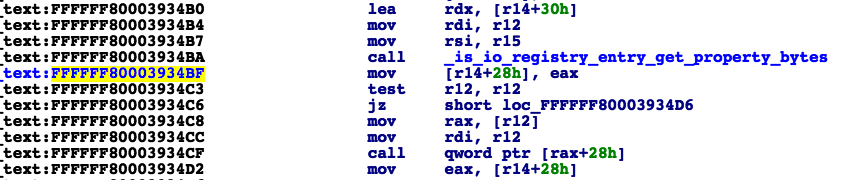
0x41414141, 0x41414141

};

发送这个给内核后，内核在反序列化的时候就会出现错误。随后我们使用io\_registry\_entry\_get\_property\_bytes()这个用户态函数就可以获取到内核返回的数据了。



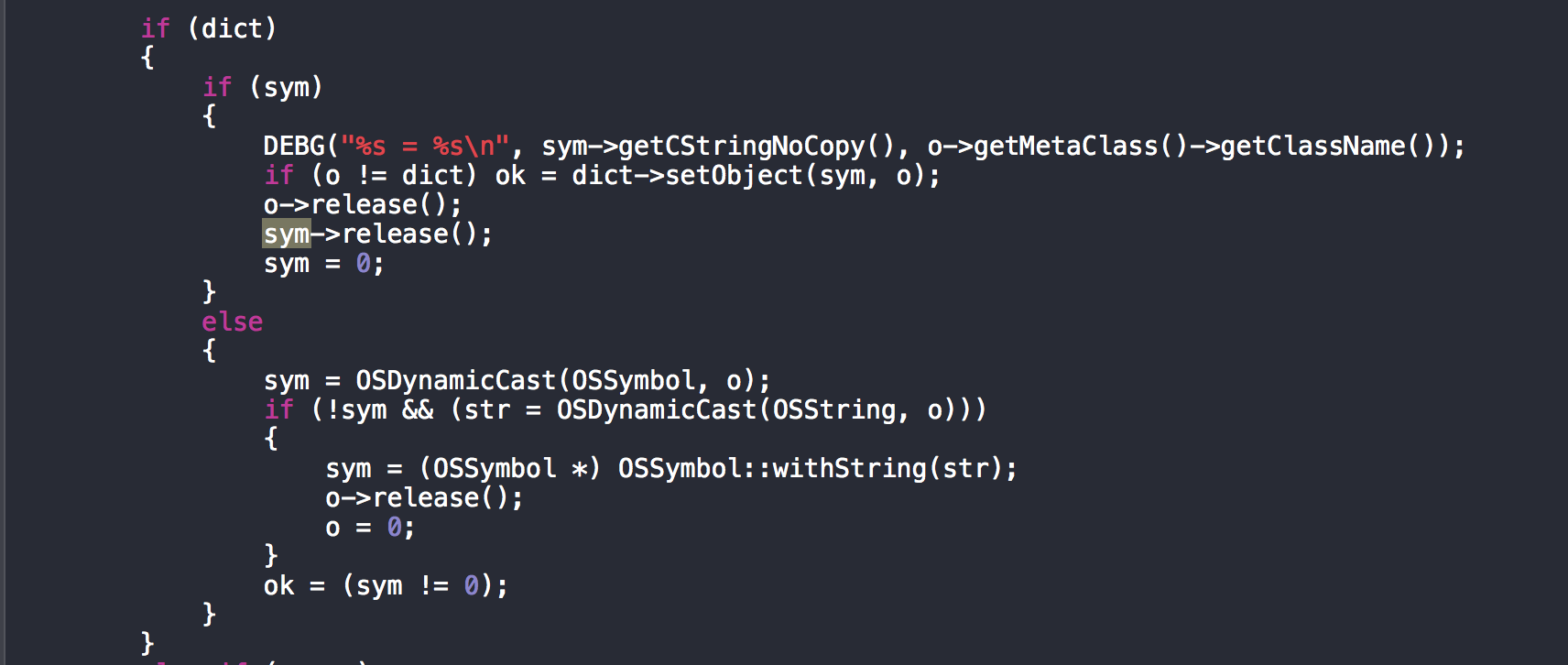
因为我们修改了OS number的长度，所以返回的数据不光有我们发送给内核的number，还有栈上数据，比如函数ret时候的返回地址-0xFFFFFF80003934BF。



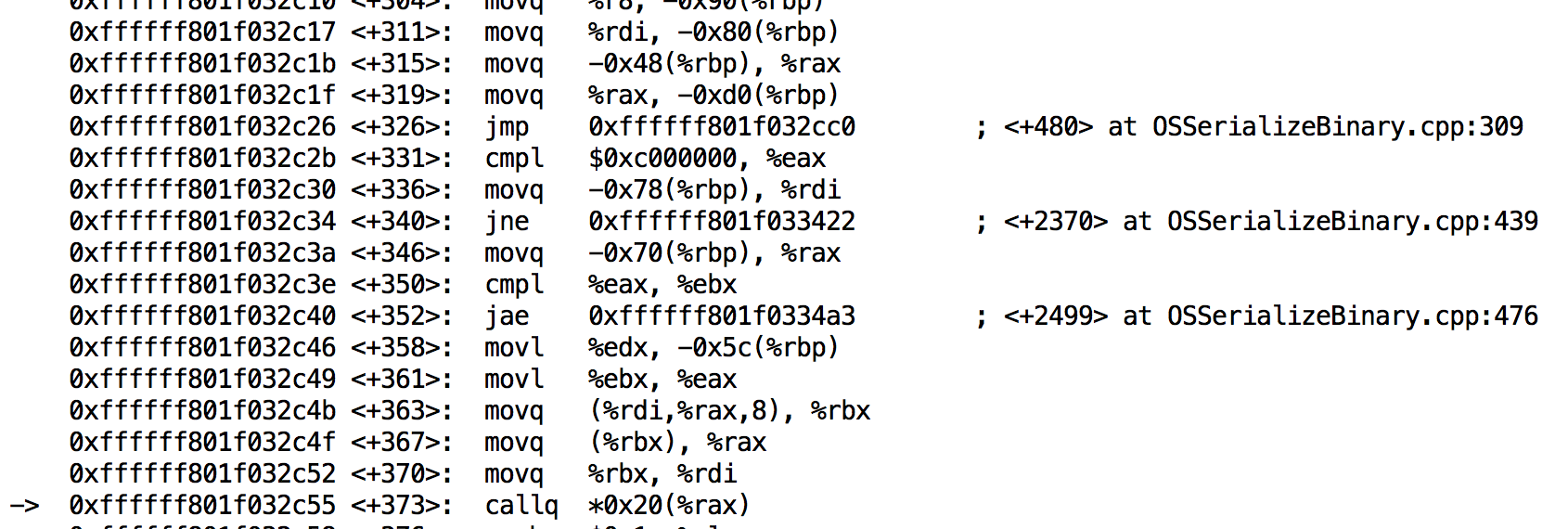
通过这个地址我们就可以计算出来kslide了。

0x03 CVE-2016-4656 内核代码执行

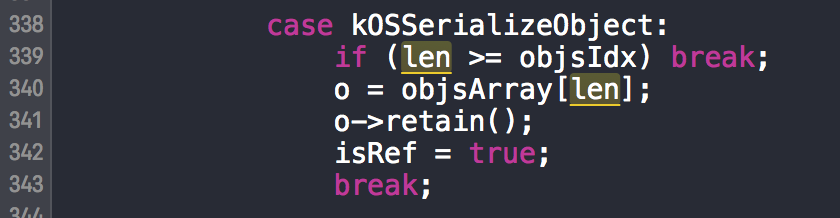
CVE-2016-4656这个漏洞其实有两种触发UAF的方法，我们这里先讲比较简单的那一种（两种方法在Stefan Esser的文章中都有介绍）。简单UAF漏洞形成的原因是OSUnserializeBinary支持用OSString和OSSymbol来作为key，并且支持用OSObject去引用之前的key。但是OSString和OSSymbol不一样的地方在于，OSString key转换为OSSymbol的过程中OSString已经被free掉了，但这个OSString却被加入了对象列表里。



因此当我们OSObject类型去引用一个已经被释放了的OSString的时候，就会产生UAF崩溃：



通过汇编崩溃的位置我们可以找到源码对应的位置是在341行创建OSObject对象的时候：



因此，如果我们能够在OSString被free的时候，立刻申请一段和OSString一样大小的内存并且构造好对应的vtable数据，当程序执行到OSObject创建的时候，内核就能成功的被我们控制。

那么如何编写利用代码呢？我们还是先创建一个序列化后的dictionary。对内核来说，这个dictionary应该是这样的：

<dict>

<string>A</string>

<bool>true</bool>

<key>B</key>

<data>vtable data...</data>

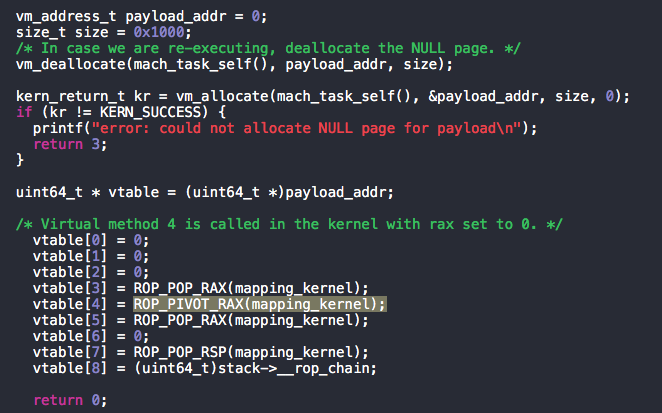
<object>1</object>

</dict>

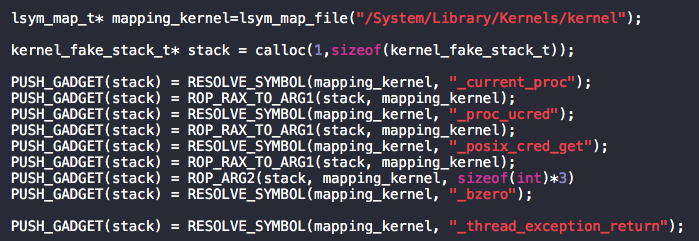
内核随后会解析这个dictionary，正如我们之前分析的，OSString-”A”在创建完后就被free掉了，这时候，我们立刻创建OSSymbol-”B”以及和OSString-”A”大小相同的OSData，就可以在OSString-”A” free后重新控制这块内存，随后当内核使用OSObject引用OSString-”A”，并调用retain()函数的时候，其实就是在调用我们已经控制的vtable了。

0x04 利用ROP提权

首先我们先申请一块内存来放vtable和ROP chain，在OS X上有一种取巧的方法，如果我们是32位的程序的话，可以使用NULL page。因此，我们先用vm\_allocate()申请到NULL Page，然后将vtable和ROP chain都保存在NULL page里：



随后在OS X上用rop提权的代码我们可以直接使用tpwn的：首先获得当前进程的ucred，然后将cr\_svuid设置为0，最后用thread\_exception\_return退出进程。



0x05 测试EXP

编写完代码后，我们来尝试执行一下我们的exp。

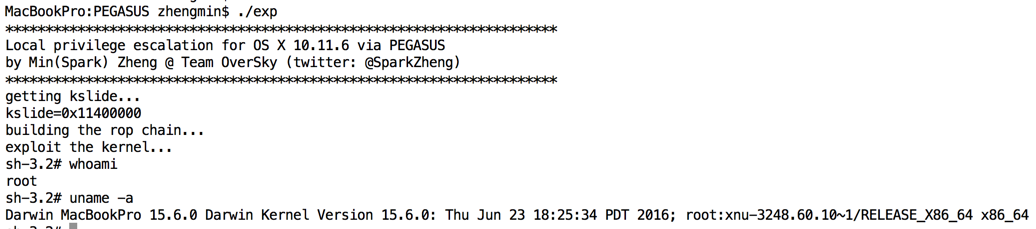
首先说一下测试环境：Mac OS X EI Capitan 10.11.6 (15G31)，在没有安装2016-01的security update的情况下（这时候内核相当于iOS 9.3.4，如果安装完2016-01 update就相当于iOS 9.3.5）。



接下来我们编译一下我们的exp：

clang -framework IOKit -framework Foundation -framework CoreFoundation -m32 -Wl,-pagezero\_size,0 -O3 exp.m lsym.m -o exp

然后运行：



可以看到我们已经成功的获取了root权限。

0x06 总结

这篇文章介绍了如何利用PEGASUS（Trident三叉戟）做到内核信息泄露以及内核代码执行，然后利用rop获取root权限。另外，因为PEGASUS（Trident三叉戟）同时存在于iOS和OS X，有兴趣的同学可以在我们发布的攻击代码的基础上，尝试一下iOS攻击代码的编写。

基于PEGASUS（Trident三叉戟）的OS X 10.11.6本地提权exp的下载地址：

<https://github.com/zhengmin1989/OS-X-10.11.6-Exp-via-PEGASUS>

0x07 参考资料

1. http://blog.pangu.io/cve-2016-4655/

2. https://sektioneins.de/en/blog/16-09-02-pegasus-ios-kernel-vulnerability-explained.html

3. https://bazad.github.io/2016/05/mac-os-x-use-after-free/

4. https://github.com/kpwn/tpwn