Local privilege escalation for OS X 10.11.6 via PEGASUS

Author: Min (Spark) Zheng @ Team OverSky

0x00 Introduction

Because of the PEGASUS apt issue on iOS, Trident exploit is very hot recently. From Lookout’s report, there are three vulnerabilities in the Trident exploit:

CVE-2016-4657: Visiting a maliciously crafted website may lead to arbitrary code execution.

CVE-2016-4655: An application may be able to disclose kernel memory.

CVE-2016-4656: An application may be able to execute arbitrary code with kernel privileges.

Although Lookout didn’t release the malware, Stefan Esser and Pangu still found the vulnerabilities of CVE-2016-4655 and CVE-2016-4656. Therefore, we can use these two vulnerabilities to achieve local privilege escalation for OS X 10.11.6 and jailbreak for iOS 9.3.4.

0x01 CVE-2016-4655 kernel info leak

Because XNU kernel doesn’t check the length of serialized OSNumber in the OSUnserializeBinary() function, we can create an OSNumber with a very long length:

uint32\_t data[] = {

0x000000d3,

0x81000001,

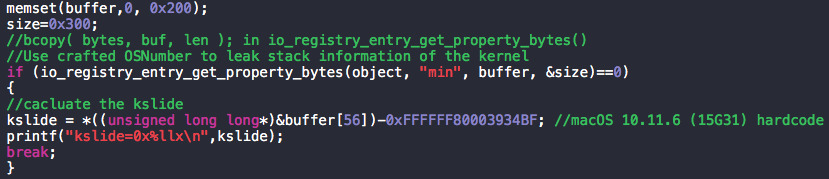
0x08000004, 0x006e696d,

0x84000200, //change the length of OSNumber

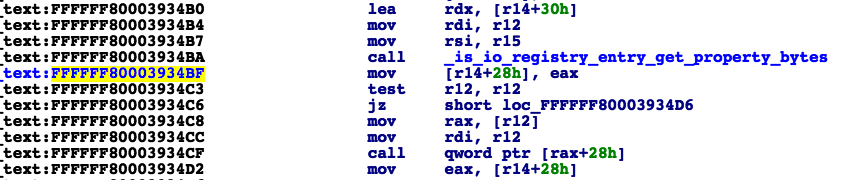
0x41414141, 0x41414141

};

After sending the serialized OSNumber to the kernel, we can use io\_registry\_entry\_get\_property\_bytes() to get the data back from the kernel：



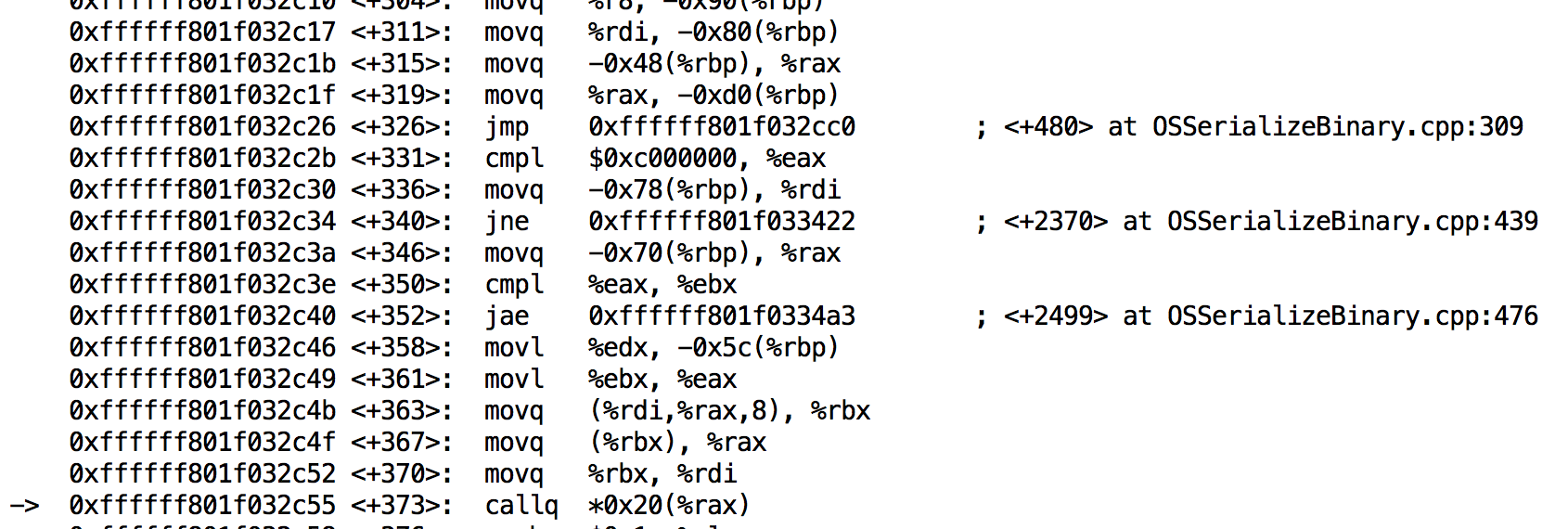
Because we can control the length of returned data, we can get extra data from the kernel stack. Some useful information like function return address can help us to calculate the kernel slide and break the kalsr protection.



0x02 CVE-2016-4656 kernel UAF

For CVE-2016-4656, Stefan Esser introduced two ways to trigger the UAF vulnerability. We will use the sample way to exploit the kernel in this article.

We know that OSUnserializeBinary() supports OSString and OSSymbol as the keys for the dictionary and we can use an OSObject to point to an old key. However, the OSString key will be freed when it convents into an OSSymbol. Therefore, if we create an OSObject and point it to a freed OSString, it will trigger UAF in the kernel. Here is the crash point when the system wants to retain an OSObject that points to a freed OSString:



Therefore, we can create a crafted dictionary:

<dict>

<string>A</string>

<bool>true</bool>

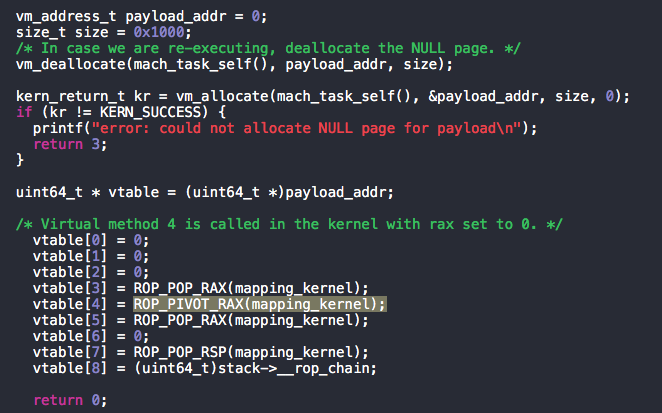
<key>B</key>

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

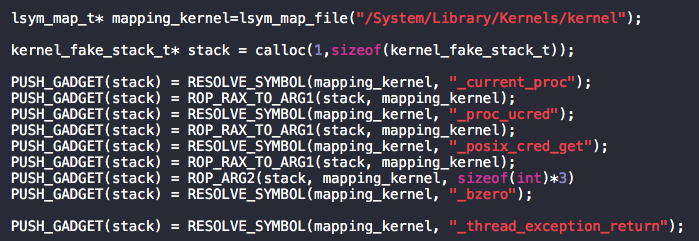
<object>1</object>

</dict>

Then we send this crafted dictionary to the kernel, RIP will be set to the vtable entry at index 4 while RAX points to the start of the vtable.



For the ROP part, we can reuse the code of tpwn and rootsh to achieve local privilege escalation on OS X.



0x03 Running the Exploit

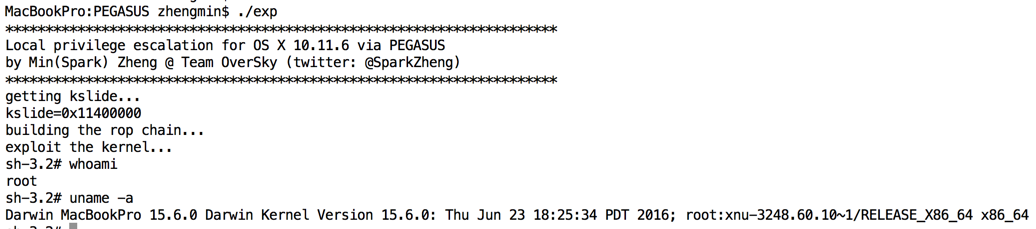
Here is the test environment: OS X EI Capitan 10.11.6 (15G31).



Note that if you want to test this exp, you should not install Security Update 2016-001 (like iOS 9.3.5 patch for PEGASUS). And I hardcoded a kernel address to calculate the kslide, it may be different on your mac.

Then we compile the exploit and run it:

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



As you can see, our exploit got the root privilege successfully.

0x04 Summary

In this article, we introduced how to use CVE-2016-4655 and CVE-2016-4656 to achieve local privilege escalation on OS X 10.11.6.

Last but not least, the exploit source code can be downloaded at:

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

0x05 Reference

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