SilverMoonSecurity / PassiveFuzzFrameworkOSX

This framework is for fuzzing OSX kernel vulnerability based on passive inline hook mechanism in kernel mode.

23 commits	1 branch	0 releases	1 contributor

Branch: master ▾ | New pull request | Create new file | Upload files | Find file | Clone or download ▾

SilverMoonSecurity committed on GitHub Update README.md … Latest commit 9c984ea on 9 Mar

bin	bin	11 months ago
src	src	11 months ago
Mac OSX key notes.docx	osx key note doc	7 months ago
README.md	Update README.md	7 months ago
Thumbs.db	thumbs	7 months ago
bin.zip	ready	11 months ago
github_readme.txt	ready	11 months ago
pacsec2016-Fuzzing and Exploiting OSX Vulnerabilities for Fun a…	PPT for passive fuzz	11 months ago

README.md

1. What is it

   This framework is for fuzzing OSX kernel vulnerability based on passive inline hook mechanism in kernel mode.

   Basically, it is a typical kernel driver which inline-hooked import API related to IOKit framework and kernel service.

   You could collect kernel dump and reproduce the vulnerability if kernel crash happens.

   You can follow my twitter: @Flyic (of moony li)to more info in detail.

   The source code is to be released after our presentation "Active fuzzing as complementary for passive fuzzing" on PacSec 2016 in Tokyo(10.26/10.27)

   https://pacsec.jp/speakers.html

   The passive fuzzing framework is based on “the_flying_circus” rootkit for OSX by fG! Special thanks to fG!

   (A Mountain Lion rootkit for Phrack #69! Copyright (c) fG!, 2012, 2013 - reverser@put.as - http://reverse.put.as All rights reserved.)

2. Requirement for running

In principle, the passive fuzzing framework could support popular OSX version for Mac Pro,Air.

As our experience, there kernel revision from 10.11 to 10.11.6 cause little or non interference to passive fuzzing.

The framework has been tested on 10.11.6 MacPro with KDK_10.11.6_15G31.kdk.

3. How to use

IMPORTANT TIPS:

Running the kernel driver would cause sudden kernel crash so as to lose all your data. Take your own risk to use this kernel driver.

3.1 Quick Start

```
If you want to try the passive fuzz just for fun, please quick try like this:
```

```
a. Load driver for  quick passive fuzz

  sh-3.2# chown -R root:wheel ./quick-pasive_kernel_fuzz.kext

  sh-3.2# kextutil ./quick-pasive_kernel_fuzz.kext

b. quick-pasive_kernel_fuzz would appear in kernel module

  sh-3.2# kextstat

  You would see the driver appears in the kernel module list. However,
```

3.2 Full Start with ThunderBolt wire

```
This start guide is suitable for all Mac OSX machine(e.g. MacPro, MacAir, Mac Mini).

This solution require for another debugger OSX machine and extra ThunderBolt wire.

1. On  target(or debugee) OSX machine:

  a. Prepare KDK and nvram

    I. Download KDK_10.11.6_15G31.kdk (take this KDK for example) and install on your Mac machine

    II. Copy kernel.development to system folder and synchronise kernel cache

      sh-3.2# cp -fr /Library/Developer/KDKs/KDK_10.11.6_15G31.kdk/System/Library/Kernels/kernel.development*
/System/Library/Kernels/

      sh-3.2# kextcache -invalid /

      sh-3.2# reboot

    III. Set up boot-args for debugging

      sh-3.2# nvram boot-args="debug=0x566 kdp_match_name=firewire fwkdp=0x8000 pmuflags=1 kext-dev-mode=1  -v"

      sh-3.2# reboot


  b. Load driver for passive fuzz

    sh-3.2# chown -R root:wheel ./pasive_kernel_fuzz.kext

    sh-3.2# kextutil ./pasive_kernel_fuzz.kext


  c. Proberbly your Mac Machine would kernel crash waiting for further deubugging


2. On  debugger OSX machine:


  IMPORTANT TIP:

    Always keep the ThunderBolt connected with the two machine always because plug and play (recognize) is NOT
supported for the crashed kernel.

  a. Prepare KDK

    I. Download KDK_10.11.6_15G31.kdk (take this KDK for example) and install on your Mac machine

    This step is not necessary but strongly recommented. When lldb in furthur following steps, lldb would match
*.dSYM symbol file between debugee machine and debugger machine. Otherwise, symbol info would NOT be shown during
your debugging.

  b.   Debug the crashed target machine

      I. Launch fwkdp service

      flyic-pro:pasive_kernel_fuzz.kext root1$ fwkdp

      II. lldb debug

      sh-3.2# cd /Library/Developer/KDKs/KDK_10.11.6_15G31.kdk/System/Library/Kernels
      sh-3.2# lldb ./kernel.development

      (lldb) kdp-remote localhost

      The debugger would wait until the target machine  crashes, and then you can type any command for debugging
including collect core dump file.
```

3.3 Full Start with Wire LAN

```
Because only lagecy OSX machine support Wire LAN (e.g. Old MacMini), this kind of debugging is not popular.

To be done
```

4. External Resource

   4.1. Crashes

   Ever collected core dump files, lldb debug logs, poc and so forth (keep updating, about 100GB).

   External link:

   http://pan.baidu.com/s/1dEJO3TJ

5. Known issue:

   5.1 MacOS 10.12 MachO format parse error

   process_kernel_mach_header(void *kernel_header, struct kernel_info *kinfo)

   could not analyze the machO file format in 10.12 which may be different with 10.11 or before so as to cause kext load fail.