Bringing Mac Memory Forensics to the Mainstream

Andrew Case / @attrc

Purpose of the Talk

- Discuss the state of Volatility's Mac support
- Show the plugins most useful for analysis
- Detect rootkit techniques with memory forensics

Acquisition

- Mac Memory Reader (ATC-NY)
  - Saves files to Macho-o format
  - Works from 10.5.x to 10.8.x, reported broken on 10.9
- OSXPmem (Michael Cohen)
  - Works on 10.9
- Mac Memoryze (Mandiant)
- 10.7+ guests in VMware Fusion
  - Fully supported by Apple

Previous Efforts before Volatility Support

- Matthieu Suiche - Mac OS X Physical Memory Analysis [1]
  - Finding page tables, processes, mounted file systems, and system call table
- Volafox
  - First real plugin based OS X analysis
  - Around 7 plugins for analysis
  - Brittle support for new versions and difficult to add

Volatility & Mac Memory Forensics

- 2.3 is the first official release with Mac support
- Has been in SVN for quite some time
  - 10.7.x support since summer 2012
  - Full support since early 2013
    - Many more OS versions supported
    - New plugins
    - Bug fixes

Supported Operating System Versions

- 32-bit 10.5.x Leopard (no 64 bit version)
- 32-bit & 64-bit 10.6.x Snow Leopard
- 32-bit & 64-bit 10.7.x Lion
- 64-bit 10.8.x Mountain Lion (no 32-bit version)
- 64-bit 10.9 support since ~3 days ago (no 32-bit version)

Mac Profiles [3]

- On the Google Code page we provide a download of all Mac profiles
  - Only copy in the ones you need
- Let us know if a version you are analyzing is missing

10.8+ ASLR

- Kernel ASLR on by default
- The static symbols in the profile are the base address before the ASLR shift
- On 10.8+ samples, Volatility will auto scan for this shift
  - You can use mac_find_aslr_shift and pass the address to the --shift option to speed up processing

Analysis Capabilities

- We will now discuss the most relevant plugins to forensics analysis and incident response handling
- These plugins find artifacts similar to Windows and Linux
  - Still a couple missing features that will be added soon

Process Enumeration

- mac_pslist*
  - Often hits an endless loop due to acquisition issues, plugin checks for the condition and bails
- mac_tasks
- mac_psaux
  - Command line arguments from userland
- mac_pstree
  - Parent/child relationship

Process Memory

- mac_proc_maps

```
$ python vol.py --profile=MacMountainLion_10_8_3_AMDx64 -f 10.8.3.mmr.macho mac_proc_maps -p 1
Volatile Systems Volatility Framework 2.3
Pid    Name            Start                End                  Perms    Map Name
-----  ----------      ----------           ----------           -----    ----------
1      launchd         0x000000010630c000 0x0000000106333000 r-x      Macintosh HD/sbin/launchd
1      launchd         0x0000000106333000 0x0000000106335000 rw-      Macintosh HD/sbin/launchd
1      launchd         0x0000000106335000 0x000000010633b000 r--      Macintosh HD/sbin/launchd
```

- mac_dump_maps
  - Writes selected maps to disk

Opened File Handles

$ python vol.py --profile=MacMountainLion_10_8_1_AMDx64 -f 10.8.1.macho mac_lsof

Volatile Systems Volatility Framework 2.3

```
Pid     File Descriptor  File Path
-----   ---------------  --------------------------
1                     0  /Macintosh HD/dev/null
1                     1  /Macintosh HD/dev/null
1                     2  /Macintosh HD/dev/null
1                     4  /Macintosh HD/dev/console
1                    81 /Macintosh HD/dev/autofs_nowait
[snip]
1031                 19  /Macintosh HD/Users/vol/Desktop/volatility/volatility/plugins/mac/pstasks.py
1031                 20 /Macintosh HD/Users/vol/Desktop/volatility/volatility/plugins/mac/pstree.py
1031                 21 /Macintosh HD/Users/vol/Desktop/volatility/volatility/plugins/mac/pgrp_hash_table.py
1031                 22 /Macintosh HD/Users/vol/Desktop/volatility/volatility/plugins/mac/pslist.py
```

Networking Information

- mac_ifconfig
  - Lists information on active network devices
- mac_netstat
  - Similar to netstat on a running system

Routing Table & Arp Cache

- For each entry:
  - Src/Dest
  - # of packet sent/recv
  - Time route was created
  - Interface

Kernel Data

Loaded Kernel Modules

$ python vol.py --profile=MacMountainLion_10_8_3_AMDx64 -f 10.8.3.mmr.macho mac_lsmod

Volatile Systems Volatility Framework 2.3

Address	Size	Refs	Version	Name
0xffffff7f91847000	0x3000	0	3.0.2	com.atc-nycorp.devmem.kext
0xffffff7f91841000	0x6000	0	10.1.24	com.vmware.kext.vmioplug.10.1.24
0xffffff7f91834000	0xd000	0	0104.03.86	com.vmware.kext.vmx86
0xffffff7f9182a000	0xa000	0	0104.03.86	com.vmware.kext.vmnet
0xffffff7f9181a000	0x10000	0	90.4.23	com.vmware.kext.vsockets
0xffffff7f91808000	0x12000	1	90.4.18	com.vmware.kext.vmci
0xffffff7f916d2000	0xe000	0	75.19	com.apple.driver.AppleBluetoothMultitouch

“Fake” Kernel Modules

```
/* This list is used in IOStartIOKit.cpp to declare fake kmod_info
* structs for kext dependencies that are built into the kernel.
* Empty version strings get replaced with osrelease at runtime.
*/
const char * gIOKernelKmods =
  "{"
  "'com.apple.kernel'                    = '';"
  "'com.apple.kpi.bsd'                   = '';"
  "'com.apple.kpi.iokit'                 = '';"
  "'com.apple.kpi.libkern'               = '';"
  "'com.apple.kpi.mach'                  = '';"
  "'com.apple.kpi.unsupported'           = '';"
  "'com.apple.iokit.IONVRAMFamily'       = '';"
  "'com.apple.driver.AppleNMI'           = '';"
  "'com.apple.iokit.IOSystemManagementFamily' = '';"
  "'com.apple.iokit.ApplePlatformFamily'  = '';"
  "'com.apple.kernel.6.0'                = '7.9.9';"
  "'com.apple.kernel.bsd'                = '7.9.9';"
  "'com.apple.kernel.iokit'              = '7.9.9';"
  "'com.apple.kernel.libkern'            = '7.9.9';"
  "'com.apple.kernel.mach'               = '7.9.9';"
  "}";
```

Mounted Filesystems

$ python vol.py --profile=MacMountainLion_10_8_3_AMDx64 -f 10.8.3.mmr.macho mac_mount

Volatile Systems Volatility Framework 2.3

Device	Mount Point	Type
/	/dev/disk3	hfs
/dev	devfs	devfs
/net	map -hosts	autofs
/home	map auto_home	autofs
/Volumes/LaCie	/dev/disk2s2	hfs

Kernel Debug Buffer

$ python vol.py --profile=MacMountainLion_10_8_3_AMDx64 -f 10.8.3.mmr.macho mac_dmesg
Volatile Systems Volatility Framework 2.3
deny mach-lookup com.apple.coresymbolicationd
MacAuthEvent en1 Auth result for: 00:26:bb:77:d2:a7 MAC AUTH succeeded
wlEvent: en1 en1 Link UP virtIf = 0
AirPort: RSN handshake complete on en1
wl0: Roamed or switched channel, reason #8, bssid 00:26:bb:77:d2:a7
en1: BSSID changed to 00:26:bb:77:d2:a7
en1::IO80211Interface::postMessage bssid changed
MacAuthEvent en1 Auth result for: 00:26:bb:77:d2:a7 MAC AUTH succeeded
wlEvent: en1 en1 Link UP virtIf = 0
AirPort: RSN handshake complete on en1
[snip]

Allocator Zones

$ python vol.py --profile=MacMountainLion_10_8_3_AMDx64 -f 10.8.3.mmr.macho mac_list_zones

Volatile Systems Volatility Framework 2.3_alpha

Name	Active Count	Free Count	Element Size
zones	182	0	592
vm.objects	153401	8832498	224
vm.object.hash.entries	135206	882875	40
maps	149	34033	232
VM.map.entries	26463	24372727	80
Reserved.VM.map.entries	35	13164	80
VM.map.copies	0	220097	80
pmap	139	7962	256
pagetable.anchors	139	7962	4096
proc	133	4042	1120

Kernel Rootkit Detection

- Volatility provides the most comprehensive kernel-rootkit detection available
- We will now walkthrough analyzing a memory sample infected with the rubilyn rootkit

mac_psxview

$ python vol.py -f rubilyn.vmem --profile=MacLion_10_7_5_AMDx64 mac_psxview

Volatile Systems Volatility Framework 2.3

Offset(P)	Name	PID	pslist	parents	pid_hash	pgrp_hash_table	session leaders	task processes
0xffffff80008d8d40	kernel_task	0	True	True	False	True	True	True
0xffffff8005ee4b80	launchd	1	False	True	True	True	True	True
0xffffff8005ee4300	kextd	10	True	True	True	True	True	True
0xffffff8005ee3ec0	UserEventAgent	11	True	False	True	True	True	True
0xffffff8005ee3640	notifyd	12	True	False	True	True	True	True
0xffffff8005ee3200	mDNSResponder	13	True	False	True	True	True	True
0xffffff8005ee2dc0	opendirectoryd	14	True	False	True	True	True	True
0xffffff8005ee2980	diskarbitrationd	15	True	False	True	True	True	True

mac_check_sysctl

 # python vol.py --profile=MacLion_10_7_5_AMDx64 -f rubilyn.vmem mac_check_sysctl

<snip>

pid2	102	RW-	0xffffff7f807ff14b	UNKNOWN	0
pid3	103	RW-	0xffffff7f807ff1ed	UNKNOWN	0
dir	104	RW-	0xffffff7f807ff2aa	UNKNOWN	
cmd	105	RW-	0xffffff7f807ff2bb	UNKNOWN	
user	106	RW-	0xffffff7f807ff2cc	UNKNOWN	
port	107	RW-	0xffffff7f807ff2dd	UNKNOWN	

mac_check_syscalls / mac_check_trap_table

$ python vol.py -f rubilyn.vmem --profile=MacLion_10_7_5_AMDx64 mac_check_syscalls | grep HOOK

Volatile Systems Volatility Framework 2.3

SyscallTable 222 0xffffff7f807ff41d HOOKED

SyscallTable 344 0xffffff7f807ff2ee HOOKED

SyscallTable 397 0xffffff7f807ffa7e HOOKED

The hooked entries allow the rootkit to hide files and file data from the filesystem

mac_ip_filters

 # python vol.py -f rubilyn.vmem --profile=MacLion_10_7_5_AMDx64 mac_ip_filters

Volatile Systems Volatility Framework 2.3

Context	Filter	Pointer	Status
INPUT	rubilyn	0xffffff7f807ff577	OK
OUTPUT	rubilyn	0xffffff7f807ff5ff	OK
DETACH	rubilyn	0xffffff7f807ff607	OK

Work from @osxreverser & Friends

- Their initial releases led to mac_notifiers
- Their second round of rootkit techniques led to Cem's submission to the plugin contest
  - Be sure to catch his talk!

mac_volshell & mac_yarascan

- MHL ported Volatility's yarascan infrastructure and volshell plugin to work with both Linux & Mac
- yarascan:
  - Search yara rules or simple strings across processes or kernel memory
- volshell:
  - Fully interactive Python shell inside Volatility environment

Userland Rootkit Detection

- I recently looked at six well-known Mac malware samples
  - Leverage
  - KitM
  - Icefog
  - …
- They all were fairly tame and did similar actions

Mac Userland Rootkits

- None of the samples performed library/code injection, privilege escalation, or new persistence mechanisms
- Noticeable Activity:
  - Run as their own process (mac_tasks)
  - Bind on a port or connect to C&C (mac_netstat)
  - Manipulate files on disk (mac_lsof)
  - Store interesting data on the heap (mac_dump_maps)

Conclusion

- Mac memory forensics has come a long way in the last year
  - Still some work to be done to reach the level of Windows & Linux, but that will be fixed soon
- 10.9.x has some interesting new research areas
  - Particularly the compressed free pages

References

[1] https://www.blackhat.com/presentations/bh-dc-10/Suiche_Matthieu/Blackhat-DC-2010-Advanced-Mac-OS-X-Physical-Memory-Analysis-slides.pdf

[2] http://code.google.com/p/volafox/

[3] https://code.google.com/p/volatility/wiki/MacMemoryForensics#Download_pre-built_profiles