Mac OS X Memory Analysis: An Overview

Rocco Gagliardi
Defense Department, scip AG
roga@scip.ch
https://www.scip.ch

Marc Ruef (Editor)
Research Department, scip AG
maru@scip.ch
https://www.scip.ch

Keywords: Apple, False Positive, Forensic, Framework, Hacker, HTTP, Law, Mac, Malware, Market

1. Preface

This paper was written in 2012 as part of a research project at scip AG, Switzerland. It was initially published online at *https://www.scip.ch/en/?labs.20120719* and is available in English and German. Providing our clients with innovative research for the information technology of the future is an essential part of our company culture.

2. Introduction

This article is an overview of current methods and tools for volatile memory analysis of a Apple Mac OS X system; additional references for each subject are listed. This is not a guide for dumping or analysing memory.

The forensic analysis of a computer involves many complex and delicate tasks. To make an accurate and reliable copy of the data stored on hard disks, there are well documented and reliable procedures. The reasons are simple: the acquisition procedure is quite easy, so an expert is not strictly required, and there are a plenty of examination tools available on the market that can be used to investigate the collected data. More complex and unreliable is the acquisition of volatile memory.

The Random-Access Memory (RAM) is an area of the computer which is used to store data while the computer is working on it. A large amount of clear text sensitive information resides only within the RAM, assuming that the OS will prevent unauthorized access and that when the computer is powered off the content will be unavailable.

It is quite obvious that we can loose evidence if we omit volatile data during an acquisition procedure. Additionally, a growing number of infections show us that the memory content will be the only place where evidence can be found.

From a forensic perspective, RAM is extremely important, because it gives an idea of what the computer was doing at the time of analysis. With the increasing number of Apple Macintosh computers in the industry, the investigation of Mac OSX RAM content is becoming very important.

3. Acquisition

Most standards and best practice guidelines, such as the "Computer Security Incident Handling Guide" from NIST or RFC 3227 "Guidelines for Evidence Collection and Archiving", include procedures of gathering volatile data: current network connections, running processes, users sessions, kernel parameters, open files etc. The problem is that to acquire data, some tools like netstat, lsof, ifconfig must be executed. These tools collect only obvious data, leaving most of the system's memory unanalyzed. Moreover, these tools are executed from user mode and even if statically linked they can print unreliable data because of a kernel level modification. The perfect tool for collecting volatile data should not rely on an operating system (see the Tribble PCI device, [Carrier2003]).

A memory acquisition procedure should be useful in different environments so in most cases it relies on a software solution, and, if well designed, just uses a very short collection process, if possible, reduced to a single command in order to minimize the impact on the machine.

Several methods for the acquisition of the memory of a Mac OSX system may be used, all with some problems/limitations. Following a list of currently most used procedures some of them not specific for the Mac world.

4. Kernel module to dump memory [Singh2006]

This method, implemented for example in MacMemoryReader, uses a kernel extension to create temporary, read-only /dev/mem and /dev/pmap devices. /dev/mem provides the same functionality provided by /dev/mem on other Unix operating systems and gives access to physical memory of the following types, as defined by EFI: "available", Loader Code, Loader Data, Bootstrap Code, Bootstrap Data, Runtime Code, Runtime Data, and, optionally, "reserved".

It does not allow access to memory ports or memory-mapped I/O devices, so it cannot be used to write device drivers.

Superuser access is required to load the extension. In addition, since something is loaded in the memory, a footprint is left in the memory itself and changes the state of the acquired system.

5. Boot time argument [Singh2006]

As a trivial alternative to the kernel extension, it is possible to use the kmem=1 boot-time argument. If kernel supports the argument, this setting will reenable the kernel memory device. Since is a boot-time argument, a reboot is required, so it is useless in case acquisition of a running computer.

6. Direct Memory Access using Firewire [Boileau2006]

This method uses a "feature" of the Firewire spec (OHCI-1394), that allows read/write access to physical memory (via DMA) for external Firewire devices. As this is DMA, the CPU/OS will not even know what's going on, so may work regardless of whether you have locked your screen; If not mitigated, Mac OSX prior to Lion 10.7.2 was vulnerable to this kind of attack; in Lion 10.7.2 it only works if a user is logged in.

Due to the firewire bus limitation, only 2GB on memory can be dumped, so with the growing memory size in modern machines, this method may be limited.

With specific HW, Macs with only the new Thunderbolt interface are also vulnerable. A summary of papers, attacks and tools related to the Firewire DMA attack can be found at *Physical memory attacks via Firewire/DMA* [1]

7. Cold boot attack [Haldermann2008]

Powering off a computer has the consequence of RAM clearing, but not immediately! Research demonstrate that without power, memory chips may retain values for a short period of time (from seconds to minutes) giving the possibilities to read the full memory content. Additionally, if the chips are cooled, they may retain values for hours.

This is deadly for disk encryption products because they rely on keeping master decryption keys in DRAM. Placing the key in memory was thought to be safe because the operating system protect them while running, and there was no way to get rid of the operating system without cutting power to the machine, which "everybody knew" would cause the keys to be erased.

8. Collecting the sleepimage

If the computer if configured to go in sleep mode, the content of the memory is saved to /var/vm/sleepimage for future restore of the exact state; this file can be used to analyze the memory. It is not a perfect image of the running system, because a process is started to put the machine in sleep-mode influencing the content itself, but a lot of valuable information can still be collected.

9. Analysis

Having a memory dump is the first step, methods to extract useful information from memory such as opened files, detailed information about each process (start/stop …), network status etc. are still needed.

Compared to Microsoft world, the Mac OSX tools are in an prehistoric era. As stated in the the MacMemoryReader Readme.txt,

> *There are currently very few tools to analyze physical memory dumps from Mac OS X machines. Hex editors, string extraction tools, search tools, and file carvers are all useful for extracting data.*

In addition, the memory can be dumped in different formats (using different offsets), and this may make some investigating tools useless.

For example, MacMemoryReader, the plug-and-play dumper, dumps the data in Mach-O binary or raw-format, while volafox (the analysis tool) requires the "linear" format (for memory addressing mechanism, consult the Intel Programmers Handbook), unless you checkout the head volafox version.

Some information can be extracted from the mach-O dump format using the command "string" and grepping for interesting sequences like – as example – "Plongname": around this string the current logged username/password can be found.

But this is a trial & error method; just dumping strings and looking around may be useful but is prone to errors and very time consuming.

10. Tools

- **Mac Memory Reader**: Mac Memory Reader is an easy to use command-line utility to capture the contents of physical RAM on a suspect computer, letting an investigator gather volatile state information prior to shutting the machine down. Results are stored in a Mach-O binary or raw-format file for later off-line analysis by the investigator. The "MacMemoryReader" can be downloaded from *here* [2].
- **volafox**: Kyeong-Sik Lee and the Korean Digital Forensic Research Center have released Volafox, a free and open-source tool to analyze Mac OS X memory images. Volafox is based on work by Matthieu Suiche and the Volatility memory analysis framework. Volafox is the only open source tool that can extract some memory information automagically; running volafox against a linear memory dump may extract following information: os_version, machine_info, mount_info, kern_kext_info, kext_info, proc_info, syscall_info, net_info. "volafox" can be downloaded from *here* [3] or checked out from http://volafox.googlecode.com/svn/trunk/. The svn checkout has the ability to read the MacMemoryReader format.
- **system tools**: The string functions manipulate strings that are terminated by a null byte; can be used to extract ASCII strings from the image. Object file displaying tool command displays specified parts of object files or libraries; can be used to look at the mach-O export made with MacMemoryReader.

- **Goldfish**: Goldfish is a free MAC OS X live forensic tool for use only by law enforcement. Its main purpose is to provide an easy to use interface to dump system RAM of a target machine via a firewire connection. It then automatically extracts the current user login password and any open AIM conversation fragments that may be available. *A short presentation about Goldfish is available* [4]

11. Summary

The methods and tools to analyze a Mac OSX memory dump are still a work in progress; currently the only tool that can extract useful information from a memory image is “volafox”; the usage of filecarvers, string and grep for known signatures is ineffiecient and may lead to false positive.

Basically it’s possible to use following patterns:

1. MacMemoryReader -> mach-O dump -> string/grep/otool -> some unorganized and informal results
2. DMA memory dump -> volafox -> predefined set of information
3. MacMemoryReader -> volafox -> predefined set of information

12. References

- [Singh2006] A.Singh, Mac OSX Internals: A Systems Approach, Addison Wesley Professional 2006, *Chapter 8* [5]
- [Suiche2010] *M.Suiche, Advanced Mac OSX Physical Memory Analysis, Blackhat 2010* [6]
- [Haldermann2008] *Haldermann et al, Lest we remember: Cold Boot Attacks on Encryptions Keys* [7]
- [Ligh2011] *S.Adair; B.Hartstein; M.Richard, Malware Analyst’s Cookbook and DVD: Tools and Techniques for Fighting Malicious Code, Wiley 2011* [8]
- [Boileau2006] *A.Boileau, Hit by a Bus: Physical Access Attacks with FireWire* [9]
- [Carrier2003] *B.Carrier; J.Grand, A Hardware-Based Memory Acquisition Procedure for Digital Investigations* [10]

13. Sources

- *Mac OS X Hacker’s Handbook* [11]
- *Mac OSX Internals: A Systems Approach* [12]
- *Mac OSX Internals – Blog* [13]
- *About the security content of OS X Lion v10.7.2 and Security Update 2011-006* [14]
- *Physical memory attacks via Firewire/DMA* [15]
- *Adventures with Daisy in Thunderbolt-DMA-land: Hacking Macs through the Thunderbolt interface* [16]
- *NIST SP 800-61 Rev. 2 – DRAFT – Computer Security Incident Handling Guide* [17]
- *Guidelines for Evidence Collection and Archiving* [18]
- *Intel 64 and IA-32 Architectures Software Developer Manuals* [19]
- *forensicswiki.org* [20]


```
{$t:Mac OS X Memory Analysis, an
                overview,$a:rcc,$v:1}
```

14. External Links

[1] http://www.hermann-uwe.de/blog/physical-memory-attacks-via-firewire-dma-part-1-overview-and-mitigation
[2] http://download.atc-nycorp.com/utilities/MacMemoryReader_3.0.1.tar.gz
[3] http://code.google.com/p/volafox/
[4] http://cci.ucd.ie/files/images/Goldfish-web.pdf
[5] http://osxbook.com/book/bonus/chapter8/kma/
[6] http://www.msuiche.net/2010/02/05/blackhat-dc-2010-mac-os-x-physical-memory-analysis/
[7] https://citp.princeton.edu/research/memory/
[8] http://www.amazon.com/Malware-Analysts-Cookbook-DVD-Techniques/dp/0470613033
[9] http://storm.net.nz/static/files/ab_firewire_rux2k6-final.pdf
[10] http://www.digital-evidence.org/papers/tribble-preprint.pdf
[11] http://www.amazon.com/The-Hackers-Handbook-Charlie-Miller/dp/0470395362/ref%3Dsr_1_1?ie=UTF8&qid=1342086760&sr=8-1&keywords=mac+osx+hackers+handbook
[12] http://www.amazon.com/Mac-OS-Internals-Systems-Approach/dp/0321278542/ref%3Dpd_sim_b_2
[13] http://osxbook.com/blog
[14] http://support.apple.com/kb/HT5002
[15] http://www.hermann-uwe.de/blog/physical-memory-attacks-via-firewire-dma-part-1-overview-and-mitigation
[16] http://www.breaknenter.org/2012/02/adventures-with-daisy-in-thunderbolt-dma-land-hacking-macs-through-the-thunderbolt-interface/
[17] http://csrc.nist.gov/publications/PubsDrafts.html#SP-800-61-Rev.%202
[18] http://www.ietf.org/rfc/rfc3227.txt
[19] http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html/
[20] http://www.forensicswiki.org/wiki/Tools%3AMemory_Imaging