Memory Forensics and the Macintosh OS X Operating System

Charles B. Leopard(✉), Neil C. Rowe, and Michael R. McCarrin

U.S. Naval Postgraduate School, Monterey, CA 93940, USA
cbleopard@gmail.com, {ncrowe,mrmccarr}@nps.edu

Abstract. Memory acquisition is essential to defeat anti-forensic operating system features and investigate clever cyberattacks that leave little or no evidence on physical storage media. The forensic community has developed tools to acquire physical memory from Apple's Macintosh computers, but they have not much been tested. This work in progress tested three major OS X memory-acquisition tools. Although all tools tested could capture system memory in most cases, the open-source tool OSXPmem bettered its proprietary counterparts in reliability and support for memory configurations and versions of the OS X operating system.

Keywords: Digital forensics · Acquisition · Main memory · Apple · Macintosh OSX · Testing · MacQuisition · OSXPMem · RECON · Reserved area

1 Introduction

Recent Macintosh OS X operating systems incorporate many recent anti-forensic features, most notably cloud storage and encryption. Users can fully encrypt many things including whole operating system volumes, making it impossible to recover forensic evidence in a reasonable time frame without passwords. Because of this, forensics on the main memory of such systems is increasingly valuable. Memory forensics can recover encryption keys, network packets, injected code, hidden processes and communications from volatile memory.

While there are many memory-acquisition tools and analysis programs for Windows operating systems, there are only a few for Macintosh systems. Ligh et al. (2014) provides a survey of information pertaining to Macintosh OS X memory forensics. A resource is the Rekall Memory Forensic Framework which began as a branch within the Volatility Project (Volatility, 2015) and became a stand-alone project in December 2013 (Rekall, 2015). Since main-memory capture is challenging, it is helpful to compare these tools to see what differences they have.

2 Methodology

This work tested three tools: BlackBag Technologies MacQuisition, Version 2014R1; OSXPMem, Version RC3; and Sumuri Forensics RECON, Version 1.0.11 (Leopard, 2015). The systems were first tested with OS X Mavericks (10.9.5), and then after upgrading to Yosemite 10.10.1 and 10.10.2. Each tool was directed to write a memory capture to an external USB 3 hard drive (7200RPM). In total 450 captures were performed (50 machines, 3 operating systems, and 3 forensic tools).

We evaluated the success rate with respect to (1) the ability of the tools to write a physical-memory capture without crashing the computer system; (2) how obtrusive the tool was (what its memory footprint was and how long it took to run); and (3) its ability to produce a capture from which standard forensic artifacts could be recovered using two memory-analysis tools, the Volatility Framework and the Rekall Memory Forensic Framework. The Rekall plugins used were arp, ifconfig, lsof, mount, netstat, psaux, and route, and the Volatility plugins used were mac_arp, mac_bash, mac-ifconfig, mac_lsof, mac-mount, mac_netstat, mac_psaux, and mac_route. We were particularly interested in differences between the memory snapshots obtained since they could indicate functional differences or coverage gaps of the tools.

The Passware Password Recovery Kit Forensic Version 13.1 was used to the confirm that encryption keys for FileVault2 were located within the OS X memory captures and could be used to decrypt the volume. To do this, FileVault2 was enabled on a MacBook Pro and a Mac Pro computer running Mavericks as well as on a MacBook Pro and a Mac Pro computer running Yosemite. Memory captures were done with MacQuisition, OSXPMem, and RECON on both the MacBook Pro and Mac Pro computers running Mavericks. RECON failed to capture physical memory from the Mac Pro computers. OSXPMem was used to capture memory from the Mac Pro and MacBook Pro running Yosemite. MacQuisition does not support Yosemite.

The rate at which memory changes affects the memory dumps acquired by tools. To analyze this we created a virtual machine using VMware Fusion Professional version 7.1.1, and used VMware to take a series of snapshots. The virtual machine ran Mavericks and was configured to use two processor cores and 8 GiB of memory. The host machine was a MacBook Pro (Retina, 15-inch, Mid 2014) with a 2.8 GHz Intel i7 processor and 16 GiB of 1600 MHz DDR3 memory. Our procedure was: (1) log into the VM and take a snapshot every fifteen minutes; (2) use the Volatility plugin to decompress the snapshots; (3) create MD5 hashes for every 4 KiB block using MD5Deep; (4) compare the MD5 hashes with the original hashes and note differences; (5) and repeat three times. We compared memory captured by each of our tools against the VM Snapshot that completed at a time closest to when each capture completed. We then compared the memory captures taken with the tools. To control for potential variations due to VMware's environment, we tested the tools on a physical Mac Mini (late 2014) with a 2.6 GHz Intel i7 processor, and 8 GiB of 1600 MHz DDR3 memory, running Mavericks. The memory captures were taken over a period of 30 min.

3 Results

Memory-acquisition speeds were within 7% for the three tools. Physical memory sizes were 67.45 MiB for MacQuisition, 0.944 MiB for OSXPmem, and 206.7 MiB for RECON, and shared memory and private memory sizes were proportional. OSXPmem had the advantage that it is a command-line tool without a graphical user interface.

We observed several crashes caused by tools. The machines that crashed were not the same nor did any one machine crash more than once. After a crash, a second acquisition attempt was often successful after the machines restarted. The exception was when RECON was used to acquire memory from the Mac Pros with 64 GiB of RAM; all the machines crashed and additional attempts also failed. Valuable forensic data is often permanently lost in a crash, so crash danger is important. Nonetheless, our experiments confirmed that if a memory capture was completed without a crash, then the capture contained every forensic artifact found by the other tools on any run.

The Passware Password Recovery Kit located the encryption keys in all of the memory captures and successfully decrypted the FileVault2 volumes. The OS X user's login password was located within all the memory captures using the hex editor iBored by searching for the term "longname" which we found frequently near a user's password. The password remained in the same block of memory during the thirty-minute period on all captures.

In another experiment, VM snapshots were taken over a period on a VM with Mac OS version Mavericks installed. A Python script counted the 4 KiB blocks whose hash values changed from the original snapshot. Results showed that on average only 5.33% of the blocks had changed after 30 min when running default processes.

Memory captures were considerably larger than the allocated physical memory due to the presence of reserved areas. The datasheet for 4th Generation Intel Core Processor Address Map describes reserved areas below 4 GiB that do not belong to the DRAM. A similar structure was observed in the virtual machines. The vmem files containing the memory in the VM snapshots were converted to raw images. Each vmem file as well as each tool memory capture was 9 GiB in size, though the VM configuration allocated 8 GiB to physical memory. (Stuttgen and Cohen, 2015) discuss how physical memory addresses are used for communication with devices (video cards, PCI cards, and flash memory) on the motherboard with memory-mapped I/O. The 1 GiB block ranges observed between 3 GiB and 4 GiB appear to be reserved for this. The chipset routes memory access around these reserved regions so that all RAM is used. This increases the size of memory captures.

All three tools captured the same range of null values observed in the first half of the memory graphs. The MacQuisition device log reported "bad addresses" were padded with zeroes beginning at block 786432 and ending at block 1048575. Each block contained 4096 bytes resulting in approximately 1 GiB of null characters. We inspected the block range in all three tool memory captures with a hex editor and confirmed that all the tools padded the same block range with zeroes.

Comparisons were done of the tool memory captures and the VM memory snapshots taken at 1 min after logging in to system, as well as the results of each tool memory capture compared to each other. Overall, we saw similar regions of non-null matches in all comparisons. However, the tool memory captures showed that most of the null characters observed in the VM snapshots were overwritten with data when the acquisition tools themselves acquired the memory. This would suggest as the memory-capture tools run, blocks of memory containing null characters get changed while other blocks remain mostly unchanged. Since these regions are large and outside the memory space used by the tools, it is unlikely that the tools themselves are changing the data directly. Rather, we conclude that some other mechanism of the operating system is writing to unused space in memory during the acquisition process.

Figure 1 shows example 4 KiB block matches between the tool-acquired memory captures and the VM snapshot SV1 (initiated at time = 1 min). The three plots show MacQuisition, OSXPmem, and RECON. Red blocks represent matches to the initial memory state which do not contain null (zero) characters; grey blocks represent blocks that match but contain null characters; and the white blocks represent blocks that have changed and do not match. The top of the diagram represents the beginning of memory, and each row represents 1024 blocks from left to right in 4 KiB block increments, so each horizontal slice represents 4 MiB of memory.

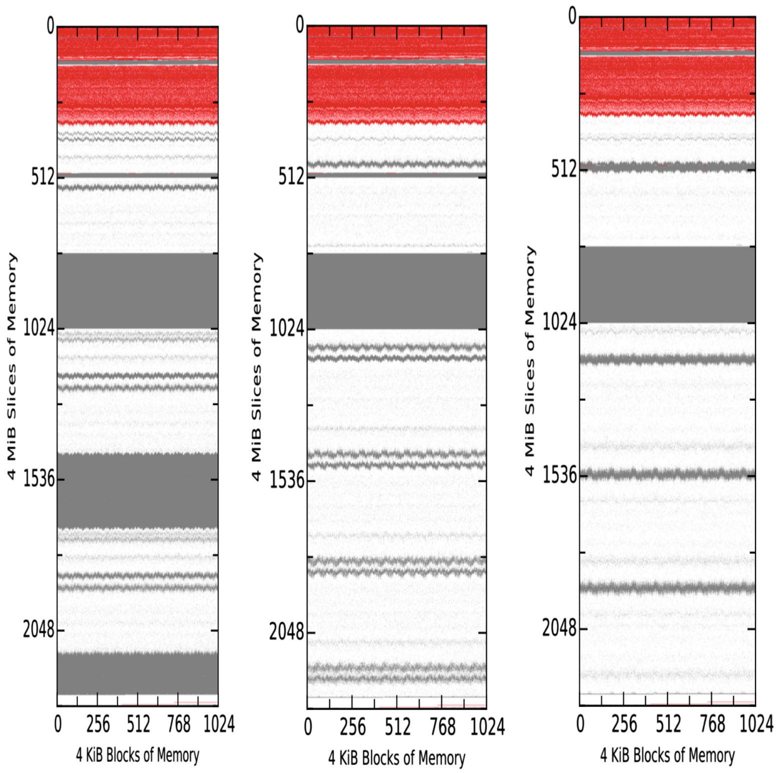

Fig. 1. Comparison of MacQuisition, OSXPmem, and RECON (left to right) on an analogous memory state. (Color figure online)

The OSXPmem data shows a reserved area that appears to agree with the output of the MacQuisition captures, again beginning at approximately the 2 GiB and continuing to 4 GiB. A range of blocks located just before the 6 GiB mark changed between OM2.5 and OM30. The edge of the red area in the figure on the right has shifted and the region of matching blocks is less. The RECON data agree with other tools about the location of reserved regions. The graphs also showed that a range of blocks, located just before the 6 GiB mark, changed between the memory capture at T2.5 and the capture at T30. The edge of the red areas between the figures was similar to before.

Data showed that the matching blocks without null characters (red) change over time by a relatively small percentage. These results were supported by the VM snapshots. It is clear that certain regions of memory are consistently captured while other regions are always in flux. Tests showed that slightly more than the first GiB of the memory capture always matches the VM snapshot. Many null values occur between the third and fourth GiB of memory which appears to be a reserved area. Two more significant blocks of null characters follow while the remaining memory appears to have changed during the acquisition.

Our tests showed that the tools represented the non-match regions (white areas) differently. This suggests not only that the tools are introducing considerable change to the memory space during the acquisition process, but also that each is changing the space in a unique way, so the changes from different tools do not match each other.

The memory-acquisition tools were also tested in non-virtual environment. The Mac Mini was configured with 8 GiB of physical memory, and each tool acquired a 9.74 GiB raw file. All three tools captured the same range of null values in the first half of the capture. This reserved area appears to be larger than in the VM memory images. Analysis with a hex editor determined that the reserved region began at 2.17 GiB and continued until 4 GiB. Data also showed that the block matches that contain non-null characters begin at approximately 30% of the total captured material, but remain relatively stable, declining by less than 10% over a 30 min period (Fig. 2). The match percent was less than what observed in the virtual environment.

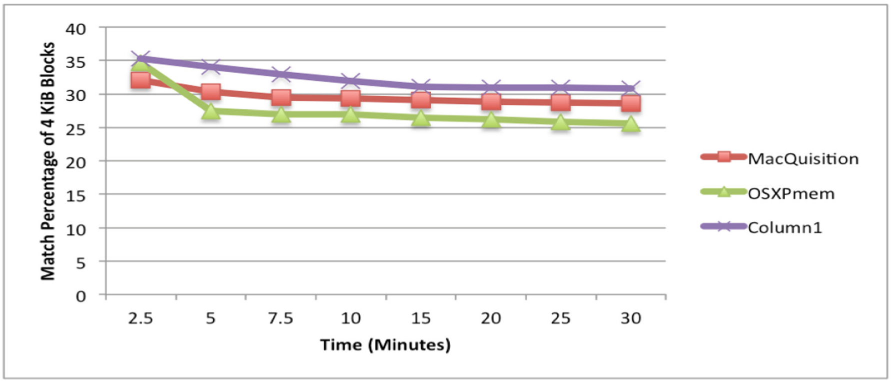

Fig. 2. Match Percentage Comparison over Time (“Column 1” = RECON).

4 Conclusions

MacQuisition, RECON, and OSXPmem were all successful in capturing memory from OS X Mavericks on Macintosh computers. They captured valuable artifacts such as FileVault2 encryption keys and volatile system data. Nonetheless, (Ligh et al., 2015) acknowledges the risk of memory-acquisition tools causing system crashes as we observed since these can be sensitive to the OS X version or the installed hardware on the system. The tool may access a reserved region or interfere with a system-critical function.

Our results showed that size of the memory capture was constant over the tools. Memory dumps were larger than the amount of physical memory (17.99 GiB versus 16 GiB for MacBook Pro was typical) due to regions reserved for firmware, ROM, and other PCI resources.

Comparison of the VM snapshots taken over thirty minutes showed that with only the default processes, memory changed only slightly. Volatility and Rekall revealed many valuable forensic artifacts remaining such as encryption keys and volatile system data. Our evidence further suggested that the tools are acquiring the blocks of memory that are not changing between captures. Though there were significant regions of memory that did not match between the tool-acquired dumps and the VM snapshots, these regions corresponded with memory blocks that contained nulls in the baseline. Our analysis of forensic artifacts using the Volatility and Rekall frameworks failed to detect any situations in which the non-matching regions corresponded to a loss of forensic evidence, since the regions or memory appear to have contained nulls before the memory acquisition.

The experiments with a non-virtual environment showed the tools successfully captured memory from a Mac Mini running Mavericks. We observed the memory captures from all three of the tools appeared similar as far as the blocks that matched and did not match as well for the blocks containing null characters. The results also agree with the results of our tests in the virtual environment in that the regions of memory that match between comparisons did not change much over time.

Future work will examine in more detail the exact changes in files over time and the discrepancies between different tools. Discrepancies suggest, without having to analyze the operating system, where volatile memory stores key operating-system parameters and links. Future work will also investigate the effects of simultaneously running various kinds of software on the operating-system memory images.

Acknowledgements. The views expressed are those of the authors and do not represent the U.S. Government.

References

Intel Corporation: Desktop 4th Generation Intel Core Processor Family, Desktop Intel Pentium Processor Family, and Desktop Intel Celeron Processor Family (2012). www.intel.com/content/dam/www/public/us/en/documents/datasheets/4th-gen-core-family-desktop-vol-2-datasheet.pdf. Accessed 25 May 2015

Leopard, C.: Memory forensics and the Macintosh OS X operating system. M.S. thesis, U.S. Naval Postgraduate School, June 2015

Ligh, M., Case, A., Levy, J., Walters, A.: Art of Memory Forensics. Wiley, Indianapolis (2014)

Rekall Team: Rekall Memory Forensic Framework: About the Rekall Memory Forensic Framework (2015). www.rekall-forensic.com/about.html. Accessed 13 March 2015

Stuttgen, J., Cohen, M.: Anti-forensic resilient memory acquisition. Digital Invest. **10**, S105–S115 (2013)

Volatility foundation: the volatility foundation – open source memory forensics (2015). www.volatilityfoundation.org/#!about/cmf3. Accessed 13 March 2015