Digital Investigation 11 (2014) S3–S12

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

In lieu of swap: Analyzing compressed RAM in Mac OS X and Linux

Golden G. Richard III [a, *], Andrew Case [b]

[a] Department of Computer Science, University of New Orleans, New Orleans, LA 70148, USA
[b] Volatility Foundation, New Orleans, LA 70001, USA

Keywords:
Memory analysis
Live forensics
Compressed RAM
Virtual memory
Digital forensics

A B S T R A C T

The forensics community is increasingly embracing the use of memory analysis to enhance traditional storage-based forensics techniques, because memory analysis yields a wealth of information not available on non-volatile storage. Memory analysis involves capture of a system's physical memory so that the live state of a system can be investigated, including executing and terminated processes, application data, network connections, and more. One aspect of memory analysis that remains elusive is investigation of the system's swap file, which is a backing store for the operating system's virtual memory system. Swap files are a potentially interesting source of forensic evidence, but traditionally, most swap file analysis has consisted of string searches and scans for small binary structures, which may in some cases be revelatory, but are also fraught with provenance issues. Unfortunately, more sophisticated swap file analysis is complicated by the difficulty of capturing mutually consistent memory dumps and swap files, the increasing use of swap file encryption, and other issues. Fortunately, compressed RAM facilities, such as those in Mac OS X Mavericks and recent versions of the Linux kernel, attempt to reduce or eliminate swapping to disk through compression. The storage of compressed pages in RAM both increases performance and offers an opportunity to gather digital evidence which in the past would have been swapped out. This paper discusses the difficulty of analyzing swap files in more detail, the compressed RAM facilities in Mac OS X and Linux, and our new tools for analysis of compressed RAM. These tools are integrated into the open-source Volatility framework.

Introduction

Traditionally, digital forensics has focused primarily on non-volatile storage devices and involved preservation, imaging, recovery, and analysis of files stored on hard drives, removable media, etc. That investigative model typically embraced a "pull the plug and image" strategy, which involved powering down forensic targets without regard for their live state and making copies of non-volatile storage devices for analysis. This resulted in loss of a significant amount of potentially actionable digital evidence, including information about currently executing processes, live network connections, data in the clipboard, volatile malware, and other OS and application data structures. Increasingly, the forensics community has become aware of the potential for *live forensics* and *memory analysis* to enhance the investigative process, yielding evidence not available on non-volatile storage. Live forensics typically involves a survey of a running machine "on-the-spot", using a set of statically compiled binaries which are executed on the target to glean information about its state and available evidence. These tools are often traditional systems administration tools, which list running processes, monitor filesystem activity, capture network traffic, monitor changes to the Windows registry, and attempt to

* Corresponding author.
E-mail addresses: golden@cs.uno.edu (G.G. Richard), andrew@dfir.org (A. Case).

http://dx.doi.org/10.1016/j.diin.2014.05.011
1742-2876/

detect malware, such as keystroke loggers. Memory analysis typically involves capture of a system's physical memory (e.g., a RAM dump, acquired via a combination of software and/or hardware) for later investigation, offline, using memory analysis tools. Live forensics and memory analysis are similar in that they both potentially offer a wealth of data to a forensic investigator that would be otherwise unavailable. Similarly, they are both potentially invasive, disturbing the state of a running system to varying degrees, but memory analysis strives to minimize this disruption by requiring only that a memory dumping utility be executed on the system, rather than a number of evidence-gathering applications. Because of recent research advances in memory analysis, much of the live system state observable with live forensics can now be recreated in the lab from a physical memory dump. One aspect of memory analysis that remains elusive is investigation of the system's swap file, which is a backing store for the operating system's virtual memory system. Virtual memory is discussed in greater detail in Section Memory analysis for modern virtual memory systems, but briefly, the swap file is typically stored on disk and contains the contents of physical memory pages that have been swapped out due to high *memory pressure*, essentially, a shortage of RAM induced by running large numbers of or particularly memory-hungry applications. The swap file can therefore contain actionable evidence, but because the swap file can be large and is stored on slow, non-volatile media, capturing a *mutually consistent* copy of both RAM and the swap file while a system continues to execute is very challenging.[1] There are additional challenges in swap file analysis, which are discussed in detail in Section Swap files as a source of evidence, but a new virtual memory component emerging in modern operating systems, called *compressed RAM* or *compressed swap*, offers an opportunity to gather digital evidence which in the past would have been swapped to disk. After providing some additional background in the following sections on virtual memory systems and memory analysis, we discuss our newly developed plugins for the Volatility framework, which automatically identify and decompress compressed memory regions in both Mac OS X Mavericks and Linux, making this data available for analysis. We also discuss the results of a series of experiments, which offer insight into the quantity and quality of the additional evidence made available by our plugins.

Memory analysis for modern virtual memory systems

Virtual memory is an essential component of modern operating systems, providing a linear address space for processes and significantly simplifying memory management. Operating systems often include a paging mechanism in the virtual memory system, to allow the total size of the allocated memory regions of executing processes to exceed the size of physical RAM, by overflowing RAM into a swap file. Primitive versions of paging have existed since the Atlas system in the 1960s (Morris et al., 1967). In this paper, we focus on operating systems that fully support paging, although some operating systems, particularly those for mobile or embedded devices, do support virtual memory but either do not support paging at all (e.g., QNX) or support paging but without a swap file, bringing in read-only pages as necessary from files on non-volatile storage (e.g., iOS). On modern hardware, virtual memory is implemented using a combination of hardware and software, with most modern CPUs providing hardware support for virtual to physical address translation and tracking whether pages are resident in RAM. Access to non-resident pages results in a *page fault*, which is handled by the operating system, triggering one of a number of possible actions, including allocation or the page being swapped in.

Swapping must be minimized to avoid *thrashing* (Denning, 1968a), where pages are continuously moved to and from the swap file because of a critical shortage of RAM, and the resultant impact on performance. Part of the reason that excessive swapping has such a serious impact on performance is the disparity between disk bandwidth and memory bandwidth, which differ by orders of magnitude. To illustrate this disparity, consider the memory bandwidth of the high- performance Mac Pro, introduced by Apple in 2013, which peaks at 60 GB/s. This model also sports some of the fastest flash storage to date, but storage bandwidth still peaks at 1.2 GB/s. To maximize performance, modern operating systems employ sophisticated

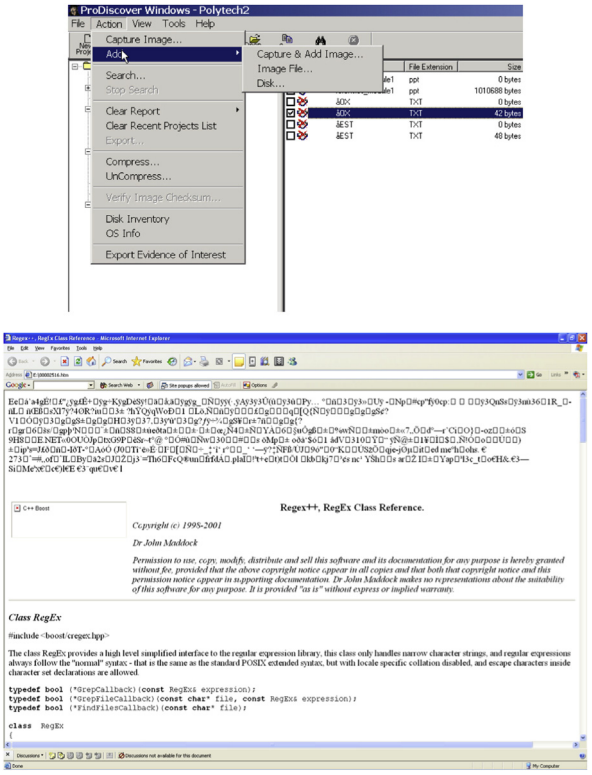

Fig. 1. Image file and HTML fragment carved from a Windows swap file. These are deleted documents "trapped" in the un-sanitized space allocated by Windows to the swap file.

[1] However, in virtualized environments, a virtual machine snapshot can be generated, which may reduce the level of inconsistency.