# Volatile Memory Acquisition via Warm Boot Memory Survivability

Timothy Vidas
Carnegie Mellon University
tvidas@cmu.edu

## Abstract

*As with other areas of digital forensics the validity and in some cases the sheer possibility of media analysis depends upon successfully acquisition of data from the media. The analysis of acquired Random Access Memory has been an active area of recent research. This paper demonstrates a USB based method of memory acquisition invoked via a system reboot. The method does not depend upon the operating system type or version.*

## 1. Introduction

The area of memory analysis has only recently been applied to digital forensics. Only in the past five years has the acquisition and analysis of volatile memory become an active research topic [1, 2, 3, 4, and others]. Most of this research has focused on the analysis of Windows memory images. As with any area of digital forensics, it is of upmost importance that the media in question be acquired correctly. Improper acquisition can cause problems during analysis or prohibit analysis altogether.

## 2. The need for memory acquisition

Over the past several years, the case has been made for the acquisition of volatile data in addition to the non-volatile media such as hard disk drives [1, 2, 3, 5, and others]. This is partially due to having the ability to acquire such data, but primarily the argument is out of necessity: some information is never written to disk and increasingly this data may be of consequence.

### 2.1. Memory resident malware

Some pedigrees of malcode exist solely in memory. Historically, codes classified as "worms" would fit into this category. Today, malware has reached an unprecedented level of sophistication and even malcode that does utilize resources such as the hard disk may store other data solely in memory. Simple examples may include routing tables, botnet command and control information, and decryption/deobfuscation keys. More complex situations also exist with packages such as MetaSploits meterpreter [6] or CANVAS' MOSDEF [7].

### 2.2. Ephemeral program data

When analyzing non-volatile media, the examiner is largely only able to consider document data that the user indented to save to disk. Occasionally a program may write temporary files, debugging logs or similar to disk, but by in large data such as document drafts and conversations for chat programs that are not configured to log chat history are lost. Similarly applications that are designed for "portable usage"[8] on LiveUSB media such as Firefox Portable, and Pidgeon Portable will primarily only leave artifacts in memory or on the portable media.

### 2.3. Active Encryption

Volume and disk encryption is becoming more pervasive every day. Many operating systems ship with encryption options, there are several production quality open source encryption projects, and vendors of storage media distribute encryption software with their products. Some tools exist that can aide the first responder in discovering the presence of active encryption [9], but this only serves and an indicator that live imaging should be performed (as opposed to "pulling the plug"). In this case, it is still recommended to acquire memory for possibly discovery of passwords and/or encryption key material.

### 2.4. Preservation of State

When power is removed from a contemporary computing device, much state is lost. Among the state that is lost is information about current network connections, active processes, dynamically allocated memory, etc. Acquiring memory allows for

preservation of many state related data structures. As the area of captured memory analysis matures, the need to perform many live system response steps such as pslist, netstat, etc [10,11] will be reduced.

### 2.5. Maturing analysis capabilities

Arguably, the area of memory forensics analysis publically began with the 2005 DFRWS Challenge. As such, merely five years later, this field is still in its infancy. Large steps have been taken toward extracting information from a memory image akin to that which is attainable via live system response tools, but the capability is still incomplete.

Prior to the creation of dedicated tools for memory analysis, general purpose tools such as the 'strings' utility could be used to inspect memory images for strings of interest. Similarly today, when adding a memory image to a case in a mainstream forensics tool such as Encase or the Forensic Toolkit, keyword searching is the predominate method of locating information in a memory image.

One of the largest memory analysis projects on the market today is Volatility[12], a python based open source project maintained by Volatile Systems. Volatility seems to have the largest user and developer base and currently several researchers use this tool as a base when creating proof of concept for new capabilities. Unfortunately, many add-ons and even volatility itself only support a handful of operating systems.

Another product worth noting is a commercial product called KNTTools, by GMG Systems[13]. This tool does have several useful features, including the automation of a "cross view diff" between a top down and a bottom up view of a memory data structures. Distribution of this tool is limited.

Several other individuals and companies have also released various tools and products [5, 14, 15, 16, 17, 18, 19, 20]. Unfortunately, in order to have a complete toolset, an analyst must manually obtain and assemble a toolkit from this array of products each at various stages of maturity, all with a wide range of dependencies.

## 3. Current memory acquisition methods

There are several memory acquisition methods currently available. Each method has benefits and drawbacks, causing the user to maintain a relatively complex decision process when a live computing system is encountered (in comparison to a powered down system).

### 3.1. Software acquisition using a provided interface

As a feature, an operating system may provide an interface to inspect physical memory. In Windows this has historically been the \\.\Device\PhysicalMemory object, in Linux /dev/mem. These interfaces were originally provided as an inspection mechanism, for example a management application could read lowmem[1] BIOS information.[21]. On a system that supports such an interface, acquiring memory is as simple as copying raw blocks from this object using a commodity tool such as dd.

Most modern operating systems have since removed or restricted use of such interfaces due to security concerns, thus relying solely on this method is inadequate.

### 3.2. Software installed into a running system

Upon the placement of restrictions on the physical memory interfaces, researchers and companies moved to address the need for memory acquisition. There are niche tools such as win32dd [22], there are tools by companies that have an operational requirement for such tools such as mdd [23], and eventually Guidance Software and Accessdata followed suite [24,25]. All of these methods involve the installation of a kernel mode driver to facilitate user mode access to physical memory. Each of them have a distinct set of features and limitations. For example, some of them support 64 bit configurations, some only support up to 4GB or RAM, and some operating systems require drivers to be signed in order to be loaded.

### 3.3. Hardware based acquisition

Software based mechanisms require interacting with already executing software. Even the best software mechanisms will affect the resulting memory image. However, hardware based mechanisms have the ability to create a more atomic image of memory.

Several products have been proposed that require special hardware to be in place prior to an incident.

[1] Low memory refers to the lowest addressable memory addresses. For MS-DOS this typically refers to the first 640 kilobytes of RAM.

The first to be published was a PCI based solution called tribble [1], followed by Komoku's CoPilot[26] (later acquired by Microsoft[27]), and others [28]. None of these products have ever become widely available.

For machines that have a firewire controller, the Direct Memory Access (DMA) feature of the IEEE 1394 specification can be used for memory acquisition [29]. This technique requires careful configuration of a firewire device or general purpose computer with a firewire port.

### 3.4. Hardware based circumvention

Software access controls still rely upon the foundations that hardware provides. It is conceivable that a toolkit could be constructed that would allow for connections to be made directly to memory chips or to board contacts. However, the creation of such a kit is largely infeasible due to the vast assortment of board form factors, memory types, case configuration, etc.

Similarly other methods are not feasible in most situations, such as a so called "Cold Boot Attack" involving cooling memory chips using liquid nitrogen and transplanting them into a donor device capable of reading the memory chips[30].

### 3.5. Other methods of acquisition

For completeness, it is worth mentioning that "full crash dumps" or "core dumps" are another method of acquiring memory. In this method, the system is somehow forced to crash, which may not be a valid course of action. Additionally, this method often involves committing the full amount of memory to disk in which is undesirable because that has the potential to overwrite evidence existing on the disk.

Finally, when dealing with a virtual machine environment, memory can often be "captured" by simply copying or exporting the virtual abstraction of physical memory as maintained by the virtual machine software. For example, the .vmem file of a paused virtual machine in some VMWare products. This is, of course, only applied to the fraction of incidents which involve a virtual machine target, and will never work for the machine hosting the virtual machines. But as entities move toward using more virtualization this method should be kept in mind for when the situation does present itself.

## 4. Another method of acquisition

It is known that under some circumstances contents of memory persist across a reboot. [31]. This phenomenon can be effected by a variety of factors including duration of power removal to the system, type of memory employed, quality of components, board designed, etc. As [30] points out, different computing systems exhibit different behaviors when power is removed. Some systems are capable of preserving large percentages of data across a reboot. In these cases, there is no requirement for liquid nitrogen or other rapid cooling techniques. Since this method does not require the use of the existing operating system, no driver must be installed and any limitations on an interface such as a PhysicalMemory object cannot be enforced.

### 4.1. Capturing remnants across system reboot

Afterlife is a proof of concept tool that employs this method, and was inspired by Robert Wesley McGrew's msramdump[32]. Afterlife is a syslinux based com32 application that upon system reboot copies physical memory contents to USB media. Since physical memory is copied to external media, the net effect is similar to the crash dump method except that the memory image is not written to the host hard disk drive.

Upon reboot (warm or cold), syslinux loads the Afterlife application into a known memory location and transfers control to Afterlife. Afterlife then locates an unused system partition on the USB media from which Afterlife was loaded (Figure 1 enumerates possible partition states). The amount of memory installed in the system is determined from BIOS calls and through a syslinux interface. Afterlife marks the partition in use, then begins to copy blocks of uninitialized memory (leftover from the previously running operating system) to this partition. When the load address for Afterlife is reached portions of the loaded Afterlife application in memory are overwritten in the acquired image with metadata about the memory acquisition. In particular the amount of memory intended to be captured is recorded. When Afterlife has successfully reached the end of physical memory (and written respective blocks to the USB media), the partition is marked as "complete" and the user is graphically notified that the acquisition is complete. Once Afterlife is booted, no user interaction is required.

| Partition ID | State |
|---|---|
| 0x40 | Unused. (The Partition *should* be zeroed during the USB configuration process) |
| 0x41 | In-use. Only used while Afterlife is executing. If Afterlife fails to complete the acquisition, the partition will be left in this state. |
| 0x42 | Complete. Afterlife has finished acquiring and the partition is ready for use with memextract. |

Figure 1: Afterlife Partition States

The very small memory footprint that syslinux offers makes it an ideal compromise between consuming too much memory upon reboot, and coding the entire tool in low level assembly to assure the smallest footprint possible. Figure 2 shows the memory layout after using Afterlife. Note that the figure is not to scale, the region from the beginning of physical memory to the end of the Afterlife image is just over one MB. Obviously the size of Afterlife remains static so acquiring memory from machines with larger amount of physical memory will yield a larger percentage of recovered memory.

| | |
|---|---|
| Top of memory (MAX 4GB) | ACPI |
| | syslinux tables |
| | Afterlife Stack (grows down) |
| . . . | uninitialized memory (remnants from system prior to syslinux execution) |
| 0x101000 | Afterlife entry point (Afterlife meta information will appear here in the acquired image) |
| 0x20000 | Dynamic space (.bss, etc) |
| 0x10000 | BIOS use |
| 0x7c00 | syslinux image |
| 0 | syslinux core and System |

Figure 2: Afterlife memory map

## 4.2. Limitations

While Afterlife does not require a firewire controller nor liquid nitrogen, it does require USB and it does require that the BIOS be configured to boot from USB[2]. It is important to explicitly note that upon reboot, some amount of memory will be overwritten by syslinux and Afterlife. Some systems will not fully enable USB 2.0 even if the installed controller is USB 2.0 compatible. In these cases acquisition duration will be similar to USB 1.0 or 1.1 speeds.

Since Afterlife depends on syslinux which runs in 32 bit protected mode without paging, there is an acquisition limit of 4GB of physical memory [33,34]. Note that Afterlife USB partitions should be made to be 4GB or larger in order to facilitate capturing any amount of memory up to and including 4GB.

## 4.3. Auxiliary tools

To keep the memory footprint small, syslinux and Afterlife have no file system drivers. For this reason, the acquired memory image exists entirely as raw data beginning at the start of a partition marked as completed. A memextract tool is distributed along with Afterlife that facilitates the conversion of an Afterlife completed partition to a raw memory image file. Memextract makes use of the metadata inserted by Afterlife to facilitate the conversion with little user interaction. For example, the user is not prompted to provide the size of the acquired image. The size cannot be determined without the metadata or user input as there is no "last byte" marker present in physical memory. Algorithmically determining the end of a memory image is problematic as the last pages of memory may have contained zeros, F's, or even the exact pattern of an arbitrary tag (if you were to attempt to tag the end of a memory image after acquisition).

Note that upon completion, Afterlife also displays some unix command line options to achieve this conversion so there is no hard dependency between memextract and Afterlife.

For testing purposes a memfill application was also developed. Memfill simply fills physical memory with a known pattern. Using memfill immediately prior to Afterlife allows for the comparison of an Afterlife acquired memory image to a expected outcome. This comparison is automated with a small application called memcompare. All of the tools listed in the section should be distributed alongside Afterlife.

[2] Other common incident response procedures relating to BIOS settings should be observed. For example one should ensure that 'quick boot' is enabled and/or 'memory check' is disabled.

### 4.4. Preliminary Results

Using memfill instead of a host OS allows for the memory images acquired via Afterlife to be more accurately inspected for deviations from the expected. Initial results varied widely across a range of manufacturers and models. For this reason, using an acquisition method such as Afterlife is currently only recommended as a last effort when no other acquisition method is feasible. For instance, some IBM laptop machines demonstrated very high retention percentages (approaching 100) while some Dell laptops experience very low percentages (0). Additionally, some machines acquire memory very slowly (~25 minutes per GB), this phenomenon has not been conclusively attributed to BIOS initialization, syslinux, or any other factor at this point.

## 5. Conclusions and Future work

The intent of this work was to fill a possible gap in the set of tools available for memory acquisition. Afterlife software is in a state where it can be used in the field, it remains to be seen whether this method is useful in enough cases to warrant use. The USB form factor is definitely desirable as USB is currently practically ubiquitous. This could lighten the need for additional tools in an acquisition toolkit (firewire, pxe, etc).

Given the vast diversity of the initial results, obvious future work includes collecting samples from a larger, diverse set of computers. Hopefully the results of a larger sample set can provide the user with a historical success rate that can aide the users decision to use Afterlife (or not).

Another direction for further work would be to extend functionality to 64 bit and physical memory capacities larger than 4 GB. This would require significant changes to syslinux or the creation of much more complex Afterlife code.

## 6. References .


[1] Carrier, B. D., and Grand, J. "A hardware-based memory acquisition procedure for digital investigations", Digital Investigation 1, Dec. 2003, pp. 50–60.

[2] DFRWS 2005 challenge.
http://www.dfrws.org/2005/challenge/

[3] Vidas, T. "The acquisition and analysis of random access memory". Journal of Digital Forensic Practice 1. Dec. 2006, pp 315–323

[4] S. Stover and M. Dickerson, "Using Memory Dumps in Digital Forensics," Login: Magazine 30, no 6, Dec 2005, pp 43–48.

[5] Burdach, Mariusz. "An introduction to the windows memory forensic" http://forensic.seccure.
net/pdf/introduction to windows memory forensic.pdf., 2005.

[6] Skape, "Metasploits Meterpreter. skape. 12/26/2004.
http://www.nologin.org/Downloads/Papers/meterpreter.pdf

[7] Aitel, Dave. "An Introduction to MOSDEF". Blackhat USA 2004. http://www.blackhat.com/presentations/win-usa-04/bh-win-04-aitel.pdf, 2004.

[8] Portable Apps. Documentation.
http://portableapps.com/about/what_is_a_portable_app

[9] Waits, C. "CERT Forensics Tools" DoD Cybercrime Conference 2007, St Louis, Mo., 2007.

[10] Nolan R., O'Sullivan C., et al. "First Responders Guide to Computer Forensics" CMU/SEI-2005-HB-001. March 2005.

[11] Adelstein, F. 2006. "Live forensics: diagnosing your system without killing it first". Communications of the ACM 49, 2 Feb. 2006, pp 63-66

[12] Volatility - by AAron Walters and Nick Pedroni Jr.
https://www.volatilesystems.com/default/volatility

[13] KnTTools and KnTList by GMG systems.
http://gmgsystemsinc.com/knttools/

[14] Schuster, Andreas. "Searching for processes and threads in Microsoft Windows memory dumps." Digital Investigation Volume 3, Supplement 1, September 2006, Pages 10-16

[15] VAD Tools. By Brendan Dolan-Gavitt
http://www.dfrws.org/2007/proceedings/p62-dolan-gavitt.pdf

[16] Memparser. By Chris Betz
http://sourceforge.net/project/showfiles.php?group_id=167028

[17] Responder, By HBGary.
https://www.hbgary.com/products-services/responder-field-edition/

[18]Memoryze. By Mandiant.
http://www.mandiant.com/software/memoryze.htm

[19] lsproc. part of WindowsIR. By Harlan Carvey.
http://windowsir.blogspot.com/2006/04/lsproc-released.html

[20] ProcessLocator. By Tim Vidas
http://nucia.unomaha.edu/tvidas/

[21] Changes to functionality in Windows Server 2003, SP1.: Device\PhysicalMemory Object. Microsoft Technet. http://technet.microsoft.com/en-us/library/cc787565.aspx

[22] win32dd. By Matthieu Suiche. http://win32dd.msuiche.net/

[23] mdd. By Mantech. http://www.mantech.com/msma/mdd.asp

[24] Winen Encase Acquisition Utilitiy. Guidance Software. http://https//support.guidancesoftware.com/forum/downloads.php?do=file&id=478

[25] FTK Imager 2.6 release notes: AccessData Corp. http://www.accessdata.com/downloads/media/Imager_2-6_ReleaseNotes.pdf

[26] Co-Pilot - A High Assurance and Independent Security Auditor. USAF, AFMC, AIR FORCE RESEARCH LABORATORY Award number: FA875005C0202. 7/5/2005.

[27] Microsoft Acquires Komoku. Microsoft strengthens anti-malware protection with leading-edge rootkit detection provider. March 20, 2008
http://www.microsoft.com/security/portal/komoku/

[28] RAM Capture Tool. By BBN Technology. http://www.hstoday.us/content/view/706/111/

[29] Dornseif, M. "Firewire – all your memory are belong to us." CanSecWest/core05, May 2005. http://cansecwest.com/core05/2005-firewire-cansecwest.pdf

[30] Halderman J., Schoen S., Heninger N. et al. "Lest We Remember: Cold Boot Attacks on Encryption Keys" USENIX Security '08 proceedings. 2008. pp 45-60.

[31] Chow, J., Pfaff, B., Garfinkel, T., And Rosenblum, M. "Shredding your garbage: Reducing data lifetime through secure deallocation.". 14th USENIX Security Symposium Aug. 2005, pp. 331–346.

[32] msramdump. By McGrew security. http://www.mcgrewsecurity.com/tools/msramdmp/

[33] Syslinux memory map: Segment 0. http://syslinux.zytor.com/wiki/index.php/Memory_Map_(Segment_0)

[34] Syslinux memory map: General. http://syslinux.zytor.com/wiki/index.php/Memory_Map_(General)