Memory forencsics The path forward

RegID:17510136
Jeong-Hwan Kim

OUTLINE

1. Introduction

- **Traditionally - digital forensics**

1.Digital forensics focused on artifacts located on the storage devices

2. Importantly, memory forensics techniques can reveal a substantial amount of volatile evidence that would be completely lost if traditional "pull the plug" forensic procedures were followed.

3. This evidence includes lists of running processes, network connections, fragments of volatile data such as chat messages, and keying material for drive encryption.

4. While there has been tremendous progress in building advanced memory forensics tools since the first rudimentary techniques were developed around 2004, much work remains to be done in this exciting research area.

■2.1 Page smearing

- **Current issues**

1.page smearing is an inconsistency that occurs in memory captures when the acquired page tables reference physical pages whose contents changed during the acquisition process.

2. In our experience, this problem is commonly encountered on systems that have 8 gigabytes or more of RAM installed as well as systems that are under heavy load.

3. Depending on where the smear occurs, this can result in undesirable results of varying degrees of severity, from memory pages belonging to one process being assigned to another in the view of the memory analysis tool, to corrupted kernel data structures.

■2.1 Page smearing

- **Issues and limitations**

1. As discussed in Ligh et al. (2014),virtualization technologies such as VMware, HyperV, and Virtual-Box provide the ability to acquire guest memory VM from the host.

2. This acquisition can be performed “instantly” by leveraging hypervisor specific features, such as snapshots and suspended states, to freeze the guest inplace.

3. This prevents smearing from occurring as the guest can no longer make modifications to memory, and the analyst can simply copy the saved memory state from the hypervisor's file system.

■2.1 Page smearing

- **Future directions**

1. Our first proposed method is a modern implementation of the BodySnatcher tool(Schatz, 2007).

2. BodySnatcher attempted to 'freeze' the running operating system and load a small, second operating system at runtime.

3. This was performed without the support of hardwarebased virtual machine support as that technology was not yet in use.

4. We envision that a modern approach to BodySnatcher could instead leverage hardware based virtualization to cleanly.

2.2 Incorporation of non-resident pages

- **Current state**

1. Demand paging is a mechanism whereby the operating system does not bring information from files on disk into memory until they are absolutely needed, usually as a result of a read or a write operation to a portion of a file.

2. Combined, these issues prevent analysts from gaining a complete picture of what was happening on the system at the time of acquisition.

3. Such pages will have revealing details of user activity, such as web browsing, command line activity, DNS requests and responses, and email activity that is often crucial to investigations.

■2.2 Incorporation of non-resident pages

- **Issues and limitations**

1. First, in order to determine which files are being accessed by running code requires the acquisition tool to essentially perform memory forensics.

2. In particular, it would need to parse the handle (file descriptor) tables as well as file-backed memory mappings of each process in order to gather a list of files to acquire.

3. Second, in order to maintain the Order of Volatility, all memory should be acquired first, and then the handle tables and memory mappings parsed second.

4. This will necessarily lead to incomplete and inconsistent data as processes will have opened and closed handles during this time as well as mapped and unmapped files.

2.2 Incorporation of non-resident pages

- **Future directions**

1. Due to the importance of data in the swap file, we feel that it is crucial for the acquisition process to correctly acquire the swap file as well as for acquisition to then incorporate analysis of it.

2. In this subsection we describe our proposed methods for acquisition tools to acquire the swap file with limited (or no) smear, and in Section "Area of focus – memory analysis" we describe how analysis frameworks can seamlessly integrate swap.

3.Memory analysis tools that integrated swap file analysis could then leverage this metadata to ignore regions that changed.

■2.3 Changes to Windows hibernation file analysis

- **Current state**

1. During the hibernation process, the operating system writes a complete capture of RAM to the local file system in order to allow for a full power down.

2. This is contrast to system ‘sleep’ modes, which put the system into a low power mode that maintains the state of system RAM.

3. Since Windows XP, memory forensic analysts have leveraged the contents of RAM written during to disk during hibernation, referred to as ‘hibernation files’, in order to acquire a historical capture of memory.

4. Analysis of these files was initially made possible due to research performed by Matthe Suiche and presented at Black Hat 2008 (Suiche, 2008).

■2.3 Changes to Windows hibernation file analysis

- **Issues and limitations**

1. Beginning with Windows 8, Microsoft substantially changed both the on-disk format of hibernation files as well as the operations performed upon it during system resume.

2. Starting with Windows 8, however, this procedure is modified and the header is left in-tact, but the rest of file's contents is overwritten with zeros(Sylve et al., 2017).

3. This makes analysis of hibernation files from running systems impossible on modern Windows versions.

■2.3 Changes to Windows hibernation file analysis

- **Future directions**

1. Instead, we propose that live acquisition scripts and procedures be altered to exclude acquisition of these files.

2. Avoiding analysis of these files will also prevent wasted analyst time and effort.

3. Furthermore, we suggest that developers of memory analysis tools and frameworks provide end-users with explicit warning messages when users try to analyze hibernation files from such systems.

■2.4 Windows 10

- **Current state**

1. In addition to the changes in hibernation analysis, Windows 10 also introduced several other features that affect memory forensics.

2. Due to the rapid adoption of Windows 10 throughout corporate enterprises, it is vital that the digital forensics community better understand this new operating system and proper memory forensic approaches applicable to it.

■2.4 Windows 10

- **Issues and limitations**

1.There are currently at least two areas where further research for Windows 10 acquisition is needed.

2. Device Guard. The first, Device Guard (Device guard deployment guide, 2016), is a major architectural change to Windows whosepurpose is to protect critical components of the operating system from tampering by malware.

3. Universal and Metro Apps. New Metro Apps not only introduced changes related to swapping of virtual memory, but they also introduce a rich set of new APIs that allow for interacting with touch screens and other peripherals that are not present on tradi-tional devices (MSDN, 2016a).

4. These new APIs allow for legitimate applications to monitor a wide range of user behavior and hardware interactions in order to respond to such events.

2.4 Windows 10

- **Future directions**

1.Universal and Metro Apps. While the introduction of the new Metro APIs and interfaces are meant for legitimate purposes, malware has historically abused such APIs for keylogging, monitoring web cameras and microphones, and a wide range of other nefarious purposes (Ligh, 2012).

2. We expect that malware authors will begin to adopt their toolkits to these new APIs as they are not currently monitored by endpoint agents and memory forensics toolkits

3. Monitoring of these interfaces will require research and development time by forensics tool developers, which gives attackers a gap in time in which to operate undetected.

2.5 Linux and Android acquisition

- **Current state**

1. Unlike Windows and Mac OS X, where a memory acquisition kernel driver can be compiled for a broad range of kernel versions, Linux kernel modules need to be compiled for every version and subversion of the kernel.

2. Creating and maintaining such a database requires dedicated support by a team of engineers as well as automation of checking for new updates and building of new operating system version kernel modules.

3. On the commercial side, Threat Protection for Linux (Garner, 2016), formerly known as Second Look, maintains a database of thousands of kernel modules and profiles.

■2.5 Linux and Android acquisition

- **Issues and limitations for Linux acquisition**

1. Leveraging a kernel module database. While useful in many in stances, the dependency of a memory analysis tool or framework on a database of all existing kernel modules has several short-comings.

2. Leveraging existing kernel modules. A unique approach to avoiding the need for a new kernel acquisition model for every kernel version was presented in Stüttgen and Cohen (2014).

3. In contrast to this approach is the one used by LiME (Sylve et al.,2012), the most widely used acquisition tool for Linux. LiME pro-vides a rich set of capabilities, such as the ability to write to local disk or over the network.

4. Leveraging /proc/kcore. The loss of /dev/mem led many in the digital forensics community to assume that all future acquisitions of Linux memory would require kernel drivers.

2.5 Linux and Android acquisition

- **Issues and limitations for Android acquisition**

1.Android acquisition. Attempting to acquire memory from Android devices entails all of the previously described problems with Linux acquisition along with several additional ones.

2.The issues facing Android memory acquisition include:

-The inability to bypass locked screens

-The need to root the device without rebooting the phone

-The need to gather the kernel headers and configuration for the phone, for which vendors are incredibly slow to produce and often never produce at all

-The lack of /proc/kcore functionality, as most of the millions of Android devices are on the 32-bit ARM platform

-The optional presence of /dev/mem, and in the authors' testing, attempting to read from /dev/mem on Android devices always resets the phone.

3. The community has struggled with these problems with Android analysis for several years, but no solution has been discovered (masdif, 2014).

2.5 Linux and Android acquisition

- **Future directions**

1.Linux memory analysis tools are highly dependent on per-version information and no research effort to date has successfully made them less dependent across a wide variety of systems.

2. This system would support the needs of both acquisition and analysis tools, but will require partners across industry in order for such an effort to be sustainable.

3. As discussed in Ali-Gombe (2012), there is great value in even userland memory dumps that contain all of the data accessible to a process.

4. Userland analysis will put malware investigators at a disadvantage, however, as any type of sophisticated userland or kernel-level malware will not be detected by such approaches.

3.1 Userland platform analysis

- **Current state**

1. The threat of rootkits and the inability to detect them on live systems has led to a substantial amount of memory forensics research time being geared towards detecting system state anomalies in kernel memory.

2. To start, both Microsoft and Apple decided to enforce that all kernel drivers must be signed (Case and Richard, 2016).

3. Microsoft also went a step further than Apple and implemented Kernel Patch Protection, which is commonly referred to as Patch Guard (Kernel patch protection, 2016).

4. The rise of userland malware. The inability for malware authors to easily load their rootkits into the kernel has led to a surge in userland-based malware.

3.1 Userland platform analysis

- **Issues and limitations**

1. While memory forensic frameworks do provide the previously listed capabilities against userland malware, there is still much more work left to be done.

2. In particular, memory forensic does not provide deep coverage of the many userland runtime platforms provided by each operating system in order to make development simpler and more standardized.

3. This leads toa lack of detection capabilities as the malware is operating higher in the application stack than the memory analysis algorithms know how to inspect.

■3.1 Userland platform analysis

- **Issues and limitations for Windows**

1.Windows. Memory forensics is currently lacking in two key areas for Windows systems.

2.The first is the ability to detect Powershell activity in a post mortem investigation.

3. Empire performs all of its Powershell work completely in-memory, without actually executing powershell.

4. .NET malware has already been found in numerous real world attacks and malicious campaigns (SecureList, 2015;Cisco, 2014; MalwareBytes, 2016), but there are currently no memory forensic capabilities to detect such activity.

■3.1 Userland platform analysis

- **Issues and limitations for Mac OS X**

1. Mac OS X. Apple provides two runtimes to allow developers simple and standard access to a wide range of system resources, including memory management, hardware devices, userand GUI activity, and more.

2. The first of these runtimes, Objective-C, is widely used throughout the operating system, and is heavily targeted by malware.

3. As discussed in a recent DFRWS paper (Case and Richard,2016), the notorious Crisis malware abused Objective-C in order to monitor user's browser activity, log system wide keystrokes,activate the microphone and webcam, and to hide from live analysis.

4. The second Apple-provided runtime platform is Swift (Apple,2016).

■3.1 Userland platform analysis

- **Issues and limitations for Linux**

1. Linux. While Linux is generally used from the command line, such as through SSH, there are specific engineering fields and companies that rely heavily on the Linux desktop.

2. Crisis, along with several other malware samples found the wild, target the Linux graphical runtime, XOrg, to spy on user activity.

3. As with Swift and .NET, there is currently no memory forensic capability to examine or detect this activity.

3.1 Userland platform analysis

- **Issues and limitations for Android**

1. Android. Android applications, such as those downloaded from the Play Store, are powered by the Android runtime.

2. Memory forensics researchers implemented deep memory analysis capabilities against Dalvik(Case; Macht, 2012).

3. Leveraging this data, investigators were quickly able to deal with packed malware, as the unpacked values would be in memory, as well as quickly being able to focus on key features of the malware.

4. The main downside to this research is that it was difficult to use across a wide range of phones due to the data structures being highly dependent on the Dalvik version as well as general Android acquisition issues as described previously.

■3.1 Userland platform analysis

- **Future directions**

1. The runtime platforms that are implemented across operating systems are fertile grounds for malware as the platforms have complete control over the runtime and there are few security tools that adequately check for malware tampering.

2. We advise that memory forensics research projects that target these platforms do so in a manner that will discover a range of malware tampering and also support a wide range of OS versions.

3. This process can be time consuming and difficult, however, as many of the platforms are fully closed source or at least partially closed source, which requires an experienced reverse engineer to perform the initial analysis.

■3.2 Application specific analysis

- **Current state**

1. This includes investigations targeting rogue insiders, anti-forensic applications, and during civil and criminal proceedings involving electronic devices.

2. In many of these cases, memory forensics techniques are able to recover information that is not available to network or disk forensics.

3.Memory forensics can assist in these situations since application information that is written in encrypted form to the local disk or the network is stored in cleartext while in memory for processing.

4. We propose that to help scale investigations and to ensure that all artifacts are recovered, that memory forensic frameworks begin to deeply analyze application artifacts in a structured manner.

3.2 Application specific analysis

- **Issues and limitations for Web browsers**

1. Web browsers and browser activity. Although disk forensics of browser activity can often recover detailed information about a user's previous browsing activity, it often still leaves much to be desired.

2. For instance,when tracking data exfiltration, an investigator may be able to see that a person browsed to Dropbox or accessed his Gmail account,but due to HTTPS, will be unable to determine what data was actually transferred when relying solely on disk forensics.

3. These applications are a prime target for malicious insiders,but unfortunately web servers rarely store data sent through the POST HTTP method.

4. With memory forensics techniques, this data can often be recovered and associated with specific web browser processes and user accounts.

3.2 Application specific analysis

- **Issues and limitations for Office applications**

1.Office applications. Microsoft Office is installed on nearly every corporate end-user system, but there is currently no memory forensic support specific to this suite of applications.

2. On the insider threat side, being able to determine the contents of documents viewed by users would be of tremendous value.

3. Leveraging memory forensics features specific to Office analysis could greatly enhance this process.

4. Memory forensics is currently only able to recover side effect activity of malicious documents, such as if the payload injects code into another process, but is not able to detect the malicious documents themselves.

■3.2 Application specific analysis

- **Future directions**

1. For memory forensics to be useful across all types of investigations and investigative scenarios, research must be performed that targets specific userland applications and the data they process.

2. Current approaches to analysis of such data do not scale and are not easily repeatable, as they rely on a mix of manual extraction and examination of strings and the knowledge and skills of the investigator.

■3.3 Compressed, in-memory swap

- **Current state**

1.The importance of the integration of swap data into memory analysis has been discussed several times throughout this paper.

2. In those instances, the discussion focused on the integration of swap from disk, but in modern operating system versions there is also a separate store of swap in memory.

3. First discussed in a forensics context in Case and Richard (2014), these in-memory stores of swapped pages are highly compressed and provide a substantial performance improvement versus pages being written out to disk.

3.3 Compressed, in-memory swap

- **Issues and limitations**

1. Swapped pages being in memory alleviates the issue of pagefile collection as discussed previously, but also introduces additional complications.

2. To start, data being compressed in-memory means that it will not be discoverable by unstructured methods such as strings, grep, and Yara scans.

3. The referenced research implemented decompression capabilities for certain Linux and Mac Volatility plugins, but this only applied to particular portions of the framework

4. There is currently no research that documents the algorithms used by this compressed store, which means that the compressed pages are essentially inaccessible to memory forensics investigators.

3.3 Compressed, in-memory swap

- **Future directions**

1. The heavily reliance of OS X, and now Windows, on in-memory compressed stores means that memory forensics frameworks must incorporate them into their page translation algorithms.

2. Just as these frameworks must be adapted to read page files from disk in a transparent manner, they must also be able to recognize when pages are compressed and then decompress them transparently for analysis.

3. Only then will the full range of information stored in memory become available.

4. This will also allow traditional unstructured analysis to become fully usable again as the framework can extract and present the decompressed pages to the investigator.

■3.4 Windows 10

- **Current state**

1. Windows 10 not only changes approaches related to memory ac quisition, but also has effects on memory analysis.

2. Based on current research, these include the introduction of native Linux support as well as changes to operating system update cycles.

3.4 Windows 10

- **Issues and limitations**

1. Starting with Windows 10, Microsoft introduced a native capability to run Linux applications inside of Windows (MSDN,2016b).

2. There is currently no forensics-oriented research on this new feature, but noted security researcher Alex Ionescu has presented extensive research on this topic.

3. Microsoft traditionally saved major updates to the operating system for service packs.

4. This new approach to updates affects Windows 7,8, and 10, but most heavily affects Windows 10.

5. The rapid development of changes,sometimes in less than a week between releases, means that memory analysis tools must be constantly checked against the latest versions of the OS in order to detect major changes to data structures and algorithms.

■3.4 Windows 10

- **Future directions**

1. The introduction of Linux capabilities into Windows will require substantial forensics research.

2. The first is that reverse engineering of a large subsystem is required for any substantial effort.

3. The introduction of rapid kernel updates changes the way that memory forensics tools traditionally represented the data structures and algorithms of analyzed operating systems.

4. For example,Volatility 2.x distributes Windows profiles on service pack boundaries, such as a profile for Windows 7 service pack 1 or Windows XP service pack 2.

5. This has led to the Volatility developers needing to generate new profiles for the subset of updates that change key data structures.

4. Technologies without memory forensics coverage

- **Apple iOS**

1. After Android, Apple's iOS, which powers iPhones and iPads, has the largest market share in the world.

2. To date, there has been no public memory forensics research published for iOS devices.

3. In fact, Stefan Essar, one of the most noted members of the jailbreak community,developed a hardware kernel debugger for older versions of the iPhone (Esser, 2011).

4. Furthermore, it's very likely that any vulnerabilities exploited to allow memory acquisition will be quickly patched.

4. Technologies without memory forensics coverage

- **Chromebooks**

1. Google's Chromebooks are locked down computing “appliances” that run a heavily customized version of Linux.

2. For security reasons,users are shielded from all parts of the operating system that digital forensics analysts traditionally collect for analysis.

- **The Internet of Things**

1. The billions of devices that power the “Internet of Things” (IOT),are increasingly being used in situations that require forensic analysis.

2. Many of these devices are built upon Linux, and due to their restrictive access environments and lackluster vendor support for 3rd party kernel code, it is difficult or impossible to acquire memory samples.

Thank you for your time and attention.