Windows memory analysis issues and Linux memory analysis footnotes

R. Carbone
Infosec Analyst and Researcher
EC-Council CHFI / SANS GCIH & GREM
DRDC – Valcartier Research Centre

Defence Research and Development Canada

Scientific Report
DRDC-RDDC-2017-R004
January 2017

ADMINISTRATIVE INFORMATION

Abstract

The purpose of this work was to identify and document the various issues that in the opinion of the author remain concerning Windows memory analysis. Minor Linux specific memory analysis issues are also discussed. Too often, publicly available memory analysis specific case studies, analyses, books, guides and how-to gloss over analysis problems including current limitations, pitfalls and caveats. Finding documentation discussing these issues is problematic as no single useful source could be found after multiple searches. Because of this, and based on the work already conducted by the author in his public and private memory analysis case studies, this report highlights and examines the more important remaining memory analysis issues. The author proposes a very short manual methodology for analysing damaged or corrupted memory images.

Significance to defence and security

This report details various issues affecting or potentially affecting digital memory analysis. Today considered a standard digital forensic practice, it is important that the Canadian Armed Forces and Canadian Law Enforcement be aware of these issues so that while conducting memory investigations they are able, when required, to determine appropriate course actions. The fact is that memory analysis is not a panacea The work described in this report was performed in collaboration with the Royal Canadian Mounted Police, as part of the Platform-to-Assembly Secured Systems (PASS) project under Joint Force Development.

Résumé

Ces travaux visaient à cerner et à documenter différents problèmes qui subsistent, selon l’auteur, au sujet de l’analyse de la mémoire de Windows. On discute aussi de problèmes mineurs liés à l’analyse de la mémoire dans Linux. Trop souvent, les études de cas spécifiques, les analyses, les livres, les guides et les savoir-faire d’analyse de la mémoire offerts au public font abstraction des problèmes d’analyse, y compris les limites, les pièges et les mises en garde actuels. Il est difficile de trouver de la documentation qui aborde ces problèmes, car il a été impossible de trouver une seule source utile après de multiples recherches. Par conséquent, et à la lumière des travaux déjà effectués par l’auteur dans ses études de cas d’analyse de mémoire publiques et privées, le présent rapport souligne et examine les principaux problèmes qui persistent dans l’analyse de la mémoire. Enfin, l’auteur propose une méthodologie manuelle très courte permettant d’analyser les images mémoire endommagées ou altérées.

Importance pour la défense et la sécurité

Ce rapport décrit en détail différents problèmes touchant ou pouvant toucher l’analyse de la mémoire numérique. De nos jours, cette dernière est considérée comme une pratique normalisée de l’informatique judiciaire. Il est donc important que les Forces armées canadiennes et les corps policiers canadiens connaissent ces problèmes, de manière à ce qu’ils puissent déterminer, au besoin, les mesures à suivre lorsqu’ils mènent des enquêtes portant sur la mémoire. En fait, l’analyse de la mémoire n’est pas une panacée. Les travaux décrits dans le présent rapport ont été effectués en collaboration avec la Gendarmerie royale du Canada, dans le cadre du projet Systèmes sécurisés de plateformes à assembler (SSPA) sous le Développement des forces interarmées.

Table of contents

Acknowledgements

The author would like to thank Martin Salois, Defence Scientist for a thorough editing of this report and extend a thanks to the peer reviewer.

Disclaimer Policy

This report examines computer forensic technologies and as such is not without risk. The reader must be familiar with both computer and memory forensic techniques and methodologies to avoid harming potential digital evidence. The reader should never work from original copies of digital evidence.

The reader must neither construe nor interpret this work as an endorsement for the various techniques, methodologies and tools examined herein as suitable for any specific purpose, construed, implied or otherwise. The use of any software discussed herein is neither an endorsement nor a recommendation for said software. Only individuals knowledgeable of the inherent risks in using said software should use it.

Furthermore, the author of this report absolves himself in all ways conceivable with respect to how the reader may use, interpret or construe this report. The author assumes absolutely no liability or responsibility, implied or explicit. The onus is entirely on the reader who must be adequately equipped and knowledgeable in the application of digital forensics.

Finally, the author and the Government of Canada are henceforth absolved from all wrongdoing, whether intentional, unintentional, construed or misunderstood on the part of the reader or any other party. If the reader does not agree to these terms, then his copy of this Scientific Report must be destroyed or returned to its source of origin. Only if the reader agrees to these terms should he continue in reading this report.

It is further assumed by all participants that if the reader has not read said Disclaimer upon reading this report and has acted upon its contents then the reader assumes all responsibility for any repercussions that may result from the information and data contained herein.

Exclusions and Limitations

This report examines issues surrounding Windows memory analysis and specific concerns with respect to Linux memory analysis. It continues with work and research pursuant to the 005A PASS RCMP "Live Forensics" Collaboration.

This report is not an introduction to digital forensics. It is expected that the reader is already familiar with the application of digital forensics.

This report only examines post-mortem memory analysis. Direct live system memory analysis is outside the scope of this work and is ordinarily not suggested, in order to prevent system contamination.

The author does not suggest or endorse any framework or tool mentioned in this report. It is entirely for the reader to decide what to use.

This report does not take into account Windows 10 or Windows Server 2012, although it is expected that in many ways what is discussed herein applies to these as well.

Finally, this work was originally written and completed November 2015 and is accurate as of that. However, due to operational requirements the author was not able to submit this report for publication until November 2016.

1 Background

In this section, important background material is presented to explain the motivations behind this report.

1.1 Objective

Many reports, detailing the minutia of Windows memory analysis, were produced by DRDC in recent years. In order to avoid detailing Law Enforcement techniques or areas of interest, malware memory analyses were used to provide concrete workable examples that progressively became more challenging to solve. Reports [5–8] dealt with malware memory analysis for the Windows operating system against Prolaco and SpyEye, 0zapftis (R2D2), Stuxnet and Tigger, respectively. Linux memory malware analysis was carried out in reports [42–44] against the IVYL, Jynx2 and KBeast rootkits, respectively.

The objective of this report is to produce an overview of all this work, performed in collaboration with RCMP, and to highlight the remaining issues.

This work was performed to support the Canadian Armed Forces (CAF), especially the Canadian Forces Network Operation Centre (CFNOC), the Royal Canadian Mounted Police (RCMP) and Canadian Law Enforcement (CLE) in general.

A future report will propose and investigative, generic broad-scoped Windows memory analysis methodology that takes into account the results and issues presented in this report.

1.2 Audience

The direct audience are RCMP and CAF decision makers. To make the work as open as possible, the audience is expanded to the general computer forensic community.

As this report is somewhat technical, the audience should be knowledgeable in information security or computer forensics. It is not an introduction to digital forensics or its application.

1.3 Relevance and accuracy

Having conducted research into memory analysis in reports [5–8] and [42–44], it has become evident that much work remains. Nevertheless, because of the hard work of countless computer forensic researchers, far too many to name here, the field of computer memory forensics is a burgeoning and promising field.

This Scientific Report represents an ensemble of forensically-pertinent information, some of which was consolidated from sometimes-conflicting sources that were ultimately unified.

2 Major issues affecting memory analysis

Contrary to what may be surmised from countless books about computer forensics and memory forensics, some very important limitations exist but are rarely discussed. Only by discussing and bringing them to the forefront will it be possible to address the shortcomings of the current tools and techniques.

2.1 Damaged or corrupted memory dump

Probably the most overlooked limitation, present since the beginning of memory forensics, is the problem of corrupted or otherwise damaged memory dumps. Memory acquisition is far from an exact science, as clearly presented in [1]. In fact, it is rife with issues and problems, some of which will not be solved any time soon. Because of the persistence of these issues, corrupted or damaged memory dumps are commonplace. Even the most carefully conducted acquisition can result in a near-useless memory dump.

The problem is multifold. Sutherland et al. [1] examined the factors at length. The issue at hand is to make the most out of what is available in a dump that cannot be analysed by a given framework because the data structures they rely on are either damaged or missing. Without these structures, these frameworks, commercial and open source alike, cannot analyse much on such a memory image. In these cases, a more generic approach must be taken to exploit the data that still reside in them. The author is currently refining such a generic approach but a simplified version is provided below.

There are alternatives to coax as much useful information as possible out from the image. Before modern memory analysis techniques, frameworks and tools, there was the DFRWS 2005 memory forensics challenge [48], which helped to pave the way forward by bringing the issue to the forefront and rendering publicly available all sound solutions for future use by others.

Even modern memory acquisition tools and techniques commonly result in only partially and sometimes completely unusable memory dumps. Problems may sometimes may be the fault of the operator. Other times, it is due to issues with the acquisition software, Windows and its Blue Screen of Death (BSoD), or other software or hardware fault that occurred during acquisition.

As always, the first action before working with a copy of the memory image is to ensure that it has been set to read-only/immutable. [4–8]

Before any data processing is conducted against the memory image, it may well be worth the effort to run *Bulk_Extractor* against the memory image. This tool is capable of identifying many non-obvious data structures (e.g., compressed data and streams) in arbitrary disk and memory images, including encryption keys and other structured data (e.g., SSNs[1], CCNs). This tool is one of the few truly multithreaded programs available for digital forensics and on very large CPU systems is capable of performing very intensive data processing. What is found by the tool is stored into various output files that may help influence the direction of the investigation. [14]

---

[1] In Canada, SSNs are known as SINs or Social Insurance Number.

The next step is to perform data recovery or carving against the memory image. There are many reliable data recovery and carving software currently available [9] that are likely to recover not just executables but web histories and various other documents. Of course, what is in the memory image will depend largely on its size. If the system had a small amount of RAM (<1 GiB) then it is likely that much of the usable data files may have been paged out. At the same time, data that has not been used in some time may also have been paged out. It is suggested to use at least two different tools for recovery/carving as the mechanisms these tools use are often quite different. [4–8]

Once the various data files have been extracted/recovered from the memory image, they should be hashed and compared against the NIST NSRL [49, 50], or other authoritative hash databases (e.g., FTK KFF [51]). All confirmed matches against known and harmless data files, with a particular emphasis on known operating system components and applications, should be moved elsewhere or deleted from the recovered/carved files, as these can be safely ignored. [4–8]

The data files that remain should be inventoried for strings using 7, 8, 16 and 32-bit strings [4–8]. String extraction can be configured to recognize strings of a minimum length. The output from these extractions can be readily indexed into a database for faster lookups. In turn, various pieces of information can be extracted from these strings, including IP addresses, web pages, user names, passwords, etc. Sometimes, the strings are encoded (e.g., Base64). These can easily be recovered using one of many base-converting tools and scripts available from the web.

All extracted/recovered files should be validated against a signature-based database. The Linux *file* command relies on the *magic* file that contains hundreds of various file signatures. However, relying on this alone may be unwise as what is often found at the beginning of a given data file may not reflect its actual contents; such is the risk of data recovery and carving. [4–8]

Regardless of the type of data recognized by the signature detection tool, use appropriate software, utilities and tools to perform additional analyses against these data files. For example, when seeking evidence of child pornography, all recovered/carved image files can be analysed using pornography detection software. In web-based crime cases, various tools and utilities can extract pertinent information from the web histories.

Malware scanners may also be of service too. Scanners should be used if malware activity is suspected, but be suspicious of any one scanner's results–they need to be corroborated with other scanners. There are a great many false positives. Various online services exist to scan a selection of files but we recommend using an offline scanning utilities for sensitive files (e.g., Metadefender[2], F.K.A. MetaScan). Implementing one's own set of scanners is not particularly difficult if you do not have the budget; all that is needed are the command line versions of the scanners. However, not all scanners have an online version and they do not all play nice with others. [4–8]

It may be useful to consider fuzzy hashing the suspicious recovered/carved data files. Fuzzy hashes can be used to identify if any of the other files recovered or extracted share a high level of similarity; such similarity may suggest a modified form of the original malware, infection or unwanted program. [4, 5, 6, 7 and 8]

---

[2] Please see https://www.opswat.com/metadefender-core for more details.

Fuzzy hashing need not be applied merely to suspicious executables. It can be applied to uncompressed images, documents, and other data. The needs of the investigation will determine if only a certain set or subset of recovered/carved data files need to be fuzzy hashed.

Finally, using YARA [52], a malware researcher tool akin to a cross between *grep* and *awk*, may be of use to the investigator and can be used regardless of a memory forensic framework's ability to analyse a memory image. It can be used with or without memory analysis framework. [18, 27 and 28]

2.2 Unsupported memory structures due to changes in Windows

Changes to key data and Windows-based memory structures typically occur in one of two ways: a new patch, fix or service pack makes undocumented kernel-based modifications or inherent changes are made in newer versions of Windows.

It takes time for the various memory forensic frameworks, commercial and open source alike, to support a new version of Windows. Alpha and beta support are sometimes offered, at least by commercial framework vendors, to specific customers to try out or to solve specific issues. For open source solutions, often times new non-production code is available for download, but not yet ready to be made part of a mature or regular release. Either way, the developers of these solutions typically work against release candidate versions of Windows to test, debug, and, where necessary, reverse engineer the underlying Windows memory data structures to implement the required changes in their products.

The primary issue with this lag is significant delay. Depending on the number of memory dumps are to analyse, their size and when support becomes available, a significant backlog can arise.

2.3 Acquisition-triggered issues

At the same time, not all software memory acquisition tools support the latest versions of Windows either. This too can result in damaged, corrupted or incomplete memory images.

When acquisition tools do not work correctly, system crashes can occur. Little can be done to acquire memory from a Windows system once it has begun the process of crashing, unless it has been configured to completely save its memory dump. Most systems are not configured this way, as this is not the default for Windows. Most Windows systems are not configured for the Crash-on-Scroll dump either.

As for hardware acquisition support, the acquisition software or tool (or both) may require FireWire, USB or other interface, all of which may not be available or may have been disabled. In fact, this situation arises regularly enough that sometimes the only way to even hope of acquiring a memory dump is to cold (or warm) boot a problematic system. Yet, this is an extreme solution which itself is fraught with perils, and can also result in incomplete, damaged or corrupted memory images due to a myriad of factors that are outside the scope of this report, although the realities of cold booting were thoroughly examined in [53].

2.4 Lack of pagefile support

Currently, only ResponderPro [30] and Rekall [11] provide direct pagefile analysis. The literature suggests that the first to provide any tangible work in pagefile analysis was Michael Gruhn [12]. Written in 2014, his paper was ahead of the pagefile support provided by Rekall that at that time was not yet supporting it. It is currently unknown when exactly ResponderPro first started supporting pagefile analysis; there is no information to go on.

Volatility does not yet support pagefile support but it is in the works [13]. According to [3], there is no framework that provides structured tandem-based analysis of a memory image and its associated pagefile. However, [3] was written several months before Rekall was made available. That said it is not known why ResponderPro was not mentioned therein.

2.5 Non-framework based pagefile analysis and acquisition

YARA analysis techniques can be applied to the pagefile [22]. Such techniques may help investigators extract as much useable information as possible from the pagefile.

Interestingly, the Windows pagefile does not always provide a current view from the perspective of the memory image. The contents of the pagefile and its pertinence to an investigation depends on how much virtual memory was in use by the operating system, what was present in the pagefile from previous uses[3] and its age with respect to the date of the suspect crime or act. This is known as pagefile drift, and while the various memory analysis frameworks that support pagefile analysis suffer from this issue, they are able to handle and accommodate for it to some extent. How far that extent goes is worth researching further.

Another problem concerning the pagefile is its acquisition while the Windows operating system is running since the pagefile is locked. FDPro, among others listed in [12], successfully acquires the pagefile. KnTDD does also [29].

Tools specifically designed for pagefile analysis are rare but *Page_brute* is one such tool that may be worth trying [22].

2.6 Memory images and pagefile analysis size limits

In theory, there does not appear to be a maximum memory image size for analysis. According to Michael Hale Ligh, of the Volatility project [10], memory images of 30–40 GB have been successfully analysed and there was a report of a Volatility community member successfully analysing an 80 GB memory image.

That said there is likely a practical upper limit. The largest memory dump successfully tested by the author was a 22 GiB RAM memory dump from a Vista Enterprise SP2 64-bit system, a VMware Workstation 9 memory dump in the VMEM format (no pagefile was acquired or copied for these analyses). Some of the Volatility plugins worked against this image while others

---

[3] This assumes that the pagefile's size is static because when it is "auto-managed" by Windows it is supposed to grow and shrink according to system demands.

crashed. The plugins that crashed ran out of memory. The same issue occurred with Rekall, but not necessarily with equivalent plugins. However, various process-listing plugins succeeded with the latest versions of both Volatility (2.4.1) and Rekall (1.4.1), both of which under Windows 7 SP1 64-bit. Moreover, some of the successful plugins took well over one hour to provide results; again, not necessarily for equivalent plugins.

A Windows 7 x64 SP1, 128 GiB memory dump failed during analysis. None of the standard Volatility plugins worked. The analysis system was equipped with 256 GiB RAM to ensure sufficient memory resources.

The author did not succeed in finding any information concerning an upper memory analysis limit for ResponderPro even after extensive searching and contacting CounterTack (formerly owned by ManTech and HBGary), to which no response was ever obtained. However, successful tests conducted by a colleague indicate that memory dumps in excess of 16 GiB, on top of a 16 GiB pagefile, is readily achievable using ResponderPro version 2.2.1. Memory and pagefile were dumped using FDPro version 2.2.2560 using the HPAK format. Processing the HPAK dump file took well over an hour before ResponderPro became responsive again.

Certainly, others have succeeded in analysing far larger memory dump files but specifics are very hard to come by in the literature.

2.7 Measuring and working with memory drift

The issue of memory drift occurs when attempting to acquire both physical memory and the pagefile. Memory tends to be faster to acquire while the pagefile is typically much slower. Because of the time required to dump both, often taking many minutes for systems with as little as 4 GiB RAM and 4 to 8 GiB swap, the pages shared between are often no longer found in the memory dump. This is because memory is extremely dynamic and may undergo considerable change within a mere matter of moments, especially if the host system is under heavy use.

Because there is a slow but growing tendency to analyse both RAM and pagefile in tandem, drift is an important problem that needs to be solved.

None of the frameworks tested which support memory and pagefile analysis provided any metric or quantifiable information concerning drift between these two forms of memory. Vidas refers to this drift as a “time sliding-window” [31].

In the author’s tests, neither Rekall nor ResponderPro complained about drift and both appeared capable of processing them in tandem, according to the abilities each had. As far as can be discerned, there are no tools or plugins specifically available to manage memory drift.

However, this is an important research question that must be answered to validate memory analysis, especially as pagefile analysis becomes more common.

Various researchers and authors have pondered and made comments concerning the support of the pagefile in a memory forensics investigation [15–26]. To date, none beyond Rekall and ResponderPro has succeeded in incorporating pagefile support, even if it is partial support into a memory forensics framework.

2.8 DKOM and other anti-forensics

Direct Kernel Object Manipulation (DKOM) is typically associated with Windows systems. DKOM-based rootkits are able to manipulate kernel structures and can hide processes and ports, change privileges and fool the Windows Event Viewer. These rootkits can be implemented through device drivers or loadable kernel modules that, due to their elevated privileges, have direct access to the kernel's memory [32].

Such rootkits hide processes by manipulating the operating system's list of active processes, changing data inside EPROCESS structures. A process is hidden by unlinking its EPROCESS from the list, connecting the pointers of the previous and of the next EPROCESS in a way that will skip the process that is being hidden. The popular FU rootkit made use of this technique.

Inspired by this and other malware, several anti-memory forensic techniques have been developed. Such techniques can be divided into two categories: anti-acquisition and anti-analysis. Anti-acquisition operates during the memory acquisition process and interfere with the memory scanner. Anti-analysis techniques try to prevent the correct analysis of the memory dump. They perform manipulations to key kernel structures to prevent memory analysis tools from finding specific fundamental kernel variables that are used as a starting point for the analysis [36]. Specifically, list and table walking solutions are likely to miss important information and cannot be relied upon [2].

Of course, the situation is not hopeless. To cope with unlinked processes, one can examine each thread to ensure its corresponding process descriptor (EPROCESS) is appropriately linked. Signature-based scanners have also been developed. They use a set of rules that precisely describe the structure of a system process or thread, respectively. The results of the scanner can be compared with the output of the standard process list in the next step. Differences and anomalies potentially indicate the presence of a malicious program [2]. Note that the Volatility framework has a module (*psscan*) that performs signature-based searches. This module applies an algorithm to locate _EPROCESS structures within a memory image and reveal potential DKOM-related attacks. In addition, various high quality case studies are available which deal with DKOM-based malware to varying degrees using the Volatility framework [4–8].

An alternative technique is proposed in [34, 35]. It uses a combination of scanning and list traversing techniques that rely on the Kernel Processor Control Region (KPCR).

Finally, a new malware technique uses the GPU and DKOM to make conducting forensic analyses more difficult [36]. A prototype malware can execute on the GPU, leaving even less traces of itself than would normally be found.

Modern memory forensic programs, tools and frameworks are signature and structure based. Depending on the type of information sought, the analyst or investigator may use both forms of detection. It is possible for these structures to be modified in the hopes of rendering direct analysis impossible or merely complicating matters for the analyst or investigator.

Haruyama and Suzuki present such a technique [37] to foil Volatility, Mandiant Redline and ResponderPro. Their technique involves locating specific key structures in memory and overwriting them. This does not affect memory image acquisition, only automated analysis. However, nothing stops an analyst or investigator from manually analysing a memory image.

Balzarottia et al. explored the use of running malware from an Intel GPU [36]. If that becomes commonplace, finding actual running malware in a standard memory image will be much harder to detect regardless of the analysis framework used. Of course, much will depend on how the malware is sent onto the video card. Many questions remain, as this is quite new. Balzarottia et al. refer to other GPU malware success stories that may be of interest to the reader. Of course, to run malware on the GPU, some arbitrary program must start the required process and migrate it to the GPU. Thus, the instigating program may leave traces in memory.

At the end of 2012, Milković presented a novel technique to hide data in Windows memory [38]. His project is aptly named "Dementia." Consisting of a user initiated program and kernel driver, it is a proof-of-concept (PoC) that hides data objects in memory. The documentation, presented in the form of a presentation, lacks in-depth specifics. The user and kernel portions of the PoC are apparently stable in 32-bit form but less so for 64-bit versions of Windows. Specifically, the PoC hides data and other artefacts inside of a memory dump during memory acquisition, with the goal of being undetectable by memory forensic frameworks, including Volatility, Memoryze, and likely others. It works by exploiting certain flaws in memory acquisition tools. To what extent data and operating system artefacts can truly be hidden requires additional work and research, although it is unlikely that data and artefacts will be successfully hidden from a diligent and thorough analysis.

Another tool that holds anti-forensic promise is ADD or "Attention Deficit Disorder." Written and developed by Williams and Torres [39], it is a PoC that pollutes computer memory with fake data and information. Currently, it works only against Windows 7 32-bit SP1, but with access to the source code, others can modify it to provide functionality against other versions of Windows. The PoC was first presented at Schmoocon 2014.

Finally, it is important to point out that some malware are exceedingly good at hiding. Regardless of the tool, framework or plugin, some malware will simply evade all but the most thorough analysis. Though not always specifically anti-forensics, their exceptional ability to hide makes them very difficult to corroborate.

2.9 Pagefile wiping

It is also worth noting that Windows provides a pagefile clearing option, effectively wiping out the pagefile's contents [45]. However, this will only occur upon system shutdown. If the power is pulled, there will be no pagefile wiping; the same is true if the system is placed into hibernation mode. Pagefile wiping works even if the pagefile is distributed across multiple partitions or disks. Although it is unlikely to be encountered by investigators, it is something of which they should be aware.

Specifically, it means that post-mortem analysis of the pagefile from such a system will not be possible. Fortunately, this is not the system's default pagefile behaviour. Tests by the author using Windows 7 indicate that Crash-On-Scroll can coexist with pagefile wiping. The Crash-On-Scroll functionality supersedes pagefile wiping, which is not activated when such a deliberate attempt to dump memory is carried out. Upon reboot, the operating system copies the memory dump file out of swap and into *%SystemRoot%*. [3, 40 and 41]

3 Linux memory analysis without framework support

This section briefly examines several important issues concerning Linux memory analysis.

3.1 Issues concerning data carving

Unlike Windows-based memory images, it turns out that data carving is not particularly effective against Linux-based memory images, at least for binary executables. Experimentation has revealed that once a Linux binary, whether an executable or a compiled library file, has been loaded into memory, it loses its ELF header, thereby making its detection and subsequent carving very difficult. Without an ELF header from which to start, data carvers and recovery software will not be able to identify the starting point of a given library or executable in memory. The author attempted ten different memory experiments using both 32 and 64-bit Linux operating systems. Between them, only one ELF-based file was ever recovered. The other recovered files were mostly text-based data files. [42–44]

These same data carving techniques worked moderately well against Windows-based memory images [42–44]. This is because Windows executables and libraries have their PE header loaded into memory.

Thus, unlike work previously carried out by the author [4–8] where copies of malware running in a given memory image could be obtained via carving as this technique will not work for Linux. Fortunately, as of Volatility 2.4, a new plugin, *linux_elf*, can help investigators determine where ELF files reside in memory using alternate means.

3.2 Issues concerning AV analysis

Further complicating Linux-based malware memory analysis is the lack of Linux-specific malware detection through AV scanners. While the various scanners used throughout the Windows reports worked fairly well against both Volatility-dumped and data-carved files, these very same AV scanners (Avast, AVG, BitDefender, ClamAV[4], Comodo, Frisk F-Prot and McAfee) fared poorly against the Linux-based rootkits. [42–44]

Other unpublished tests from the author using Eset and Sophos proved equally disappointing.

In fact, quite the opposite was expected. Since these rootkits were all open sourced PoC, it would have followed that the various scanners would have included some basic signature or heuristic detection capability. After all, these rootkits will inevitably be used as the basis for future evil rootkits. Unfortunately, this was not the case at all.

---

[4] ClamAV was used in some Windows memory malware reports but not others.

3.3 Issues concerning the NSRL

The National Software Reference Library (NSRL) is a standardised and trustworthy source of computer operating system and application file names and hashes (MD5/SHA-1). It is not particularly well suited to Linux-based investigations as there are far too many Linux distributions (hundreds of publicly available distributions are known to exist) to be covered by the NSRL, including all the various kernel versions in use[5]. As such, it does not make sense to rely on the NSRL for file name listings and hashes for comparative purposes against data files recovered from a Linux memory image. [42–44]

3.4 Occasional failures in Linux memory analysis frameworks

On occasion, the main Linux memory analysis frameworks, SecondLook and Volatility are unable to analyse a memory image. The problem is not so much with the frameworks but rather the Linux kernel profile [46], which is necessary for both frameworks to analyse a Linux memory dump. Profiles contain the data structures and debug symbols required for successful analysis and these are unique to a kernel build.

Even if a memory acquisition and kernel profile generation[6] succeed without issue, there is no guarantee that the framework will successfully analyse the memory image. While this is more of a problem with Volatility than SecondLook, it does happen to both. The author has tried analyses against well over a hundred Linux systems consisting of Fedora, Ubuntu, OpenSUSE, Debian, ArchLinux and Red Hat Enterprise Linux. From these systems, after correctly generating profiles manually [46, 47] for both Volatility and SecondLook, the failure rates were still at 19% and 3%, respectively.

---

[5] A full listing of which Linux distributions are supported by a given version of the NSRL can be found in its “*nsrlprod.txt*” file.

[6] SecondLook can pull many thousands of prebuilt kernel profiles from an online repository specifically for SecondLook clients. These profiles can also be successfully used in lieu of Volatility-specific profiles [45].

4 Wrap-up and discussion

This report provides guidance and things to watch out for when conducting memory forensics, regardless of the framework in use or if a manual approach must be taken.

Although many memory analysis issues have yet to be solved or even completely understood with respect to their consequences, the domain of computer memory analysis has greatly advanced these last six years. The Volatility memory forensics project has contributed immensely to this effort and, without it, forensic investigators, analysts and researchers would be too reliant on closed source proprietary systems. With Volatility, and to a lesser extent Rekall, we can better understand memory analysis and better validate our results. While Volatility may not have all the features of its commercial heavyweight counterparts, it is not far behind and, generally, much more flexible.

In some ways, memory analysis is more mature than memory acquisition. All that is required is a nearly complete and intact image and sufficient computing resources for the analysis framework to perform its magic under keen guidance.

At the same time, Linux memory analysis has also progressed much these last few years as well. But, as it is completely different from Windows, it faces a unique set of challenges, some of which have been highlighted in this report. It is the author's hope to continue researching Linux memory analysis in a future report.

This page intentionally left blank.

References

[1] Sutherland, Iain; Evans, Joe; Tryfonas, Theodore and Blyth, Andrew. Acquiring volatile operating system data tools and techniques. Technical paper. ACM SIGOPS Operating Systems Review, Volume 42, Issue 3, Pages 65–73. April 2008. Last Accessed: November 2015. http://dl.acm.org/citation.cfm?id=1368516.

[2] Vömel, Stefan and Freiling, Felix C. A survey of main memory acquisition and analysis techniques for the windows operating system. Technical paper. Digital Investigation, Volume 8, Issue 1, Pages 3–22. July 2011. Last Accessed: November 2015. http://ac.els-cdn.com/S1742287611000508/1-s2.0-S1742287611000508-main.pdf?_tid=6e4b831a-37c6-11e5-98fc-00000aacb35d&acdnat=1438376289_962ea4932c8bc01de07d603dff0b5bb1.

[3] Ligh, Michael Hale et al. The Art of Memory Forensics. Book. First edition. Wiley Publishing. 2014. ISBN-13: 9781118825099.

[4] Carbone, Richard. Malware memory analysis for non-specialists: Investigating a publicly available memory image of the Zeus Trojan horse. Technical memorandum. DRDC Valcartier TM 2013-018. Defence R&D Canada. April 2013. Last Accessed: November 2015. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc166/p801024_A1b.pdf.

[5] Carbone, Richard. Malware memory analysis for non-specialists: Investigating publicly available memory images for Prolaco and SpyEye. Technical memorandum. DRDC Valcartier TM 2013-155. Defence R&D Canada. October 2013. Last Accessed: November 2015. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc166/p801025_A1b.pdf.

[6] Carbone, Richard. Malware memory analysis for non-specialists: Investigating publicly available memory image 0zapftis (R2D2). Technical memorandum. DRDC Valcartier TM 2013-177. Defence R&D Canada. October 2013. Last Accessed: November 2015. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc166/p800963_A1b.pdf.

[7] Carbone, Richard. Malware memory analysis for non-specialists: Investigating publicly available memory image for the Stuxnet worm. Scientific report. DRDC-RDDC-2013-R1. Defence R&D Canada. November 2013. Last Accessed: November 2015. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc165/p538612_A1b.pdf.

[8] Carbone, Richard. Malware memory analysis for non-specialists: Investigating publicly available memory image for the Tigger Trojan horse. Scientific Report. DRDC-RDDC-2013-R28. DRDC. June 2014. Last Accessed: November 2015. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc166/p800199_A1b.pdf.

[9] Carbone, Richard. File recovery and data extraction using automated data recovery tools: A balanced approach using Windows and Linux when working with an unknown disk image and filesystem. Technical memorandum. DRDC Valcartier TM 2009-161. Defence R&D Canada. January 2013. Last Accessed: November 2015. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc150/p531895_A1b.pdf.

[10] Ligh, Michael Hale. FAQ: Frequently Asked Questions – Deprecated. FAQ. Volatility Foundation. Last modified: September 12, 2012. Last Accessed: November 2015. https://code.google.com/p/volatility/wiki/FAQ21.

[11] Cohen, Michael. Windows Virtual Address Translation and the Pagefile. Online informational web page. Rekall. 2014. Last Accessed: November 2015. http://www.rekall-forensic.com/posts/2014-10-25-pagefile.html.

[12] Gruhn, Michael. Windows NT pagefile.sys Virtual Memory Analysis. Technical paper. Department of Computer Science, IT Security Infrastructure, Friedrich-Alexander University Erlangen-Nürnberg. 2014. Last Accessed: November 2015. https://www1.cs.fau.de/filepool/gruhn/pagefile.pdf.

[13] Ligh, Michael Hale. VolatilityRoadmap: Volatility Roadmap. Online informational web page. Volatility Foundation. Last modified: November 5, 2013. Last Accessed: November 2015. https://code.google.com/p/volatility/wiki/VolatilityRoadmap.

[14] Bradley, Jessica R. and Garfinkel, Simson L. Bulk Extractor 1.4: User Manual. User manual. Version 1.4. Digitalcorpora.org. March 2015. Last Accessed: November 2015. http://digitalcorpora.org/downloads/bulk_extractor/BEUsersManual.pdf.

[15] Iqbal, Hameed. Forensic Analysis of Physical Memory and Page File. Master's thesis. Department of Computer Science and Media Technology, Gjøvik University College. 2009. Last Accessed: November 2015. http://brage.bibsys.no/xmlui/bitstream/handle/11250/143807/Hameed%2bIqbal.pdf?sequence=1&isAllowed=y.

[16] Epifani, Mattia. Tor Forensics on Windows OS. Presentation. SANS EU Digital Forensics Summit, Prague. October 5, 2014. Last Accessed: November 2015. https://digital-forensics.sans.org/summit-archives/dfirprague14/Tor_Forensics_On_Windows_OS_Mattia_Epifani.pdf.

[17] Trivedi, Nisarg. Study on Pagefile.sys in Windows System. Technical paper. IOSR Journal of Computer Engineering (IOSR-JCE), e-ISSN: 2278-0661, p- ISSN: 2278-8727, Volume 16, Issue 2, Ver. V (Mar-Apr. 2014), PP 11-16. April 2014. Last Accessed: November 2015. http://www.iosrjournals.org/iosr-jce/papers/Vol16-issue2/Version-5/C016251116.pdf.

[18] Dias, Ricardo. Intelligence-Driven Incident Response with YARA. SANS Gold paper. SANS.org. October 2014. Last Accessed: November 2015. http://www.sans.org/reading-room/whitepapers/forensics/intelligence-driven-incident-response-yara-35542.

[19] Martin, James E. Detection of Data Hiding in Computer Forensics. Presentation. NebraskaCERT Conference. August, 2008. Last Accessed: November 2015. http://www.certconf.org/presentations/2008/files/C1.pdf.

[20] Nicholas P. Maclean. Acquisition and analysis of windows memory. Master’s thesis, University of Strathclyde. 2006. Last Accessed: November 2015. http://4tphi.net/fatkit/papers/NickMaclean2006.pdf.

[21] Kornblum, Jesse D. Using Every Part of the Buffalo in Windows Memory Analysis. Technical paper. Digital Investigation, Volume 4, Issue 1, March 2007, Pages 24–29. March 2007. Last Accessed: November 2015. http://www.sciencedirect.com/science/article/pii/S1742287607000047.

[22] Petroni, Nick et al. FATKit: A Framework for the Extraction and Analysis of Digital Forensic Data from Volatile System Memory. Journal of Digital Investigations, 3(4), 2006. December 2006. Last Accessed: November 2015. http://ac.els-cdn.com/S1742287606001228/1-s2.0-S1742287606001228-main.pdf?_tid=c8473936-81a9-11e5-a99b-00000aacb35d&acdnat=1446500371_ca5b2bd55975d53544d4cbae92c3b75a.

[23] Matonis, Michael. Page_brute (software tool). Readme.md. January 5, 2014. Last Accessed: November 2015. https://github.com/matonis/page_brute.

[24] Lee, Seokhee et al. Windows Pagefile Collection and Analysis for a Live Forensics Context. Technical paper. Center for Information Security Technologies, Korea University, Seoul, Korea and Department for Electronics for Automation, University of Brescia, Via Branze 38, Brescia, Italy. December 2007. Last Accessed: November 2015. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4426211&tag=1.

[25] Perry, Brandon. Analyzing the Windows pagefile.sys from GNU/Linux. Forensics blog. Brandon Perry. October 4, 2011. Last Accessed: November 2015. http://volatile-minds.blogspot.ca/2011/10/analyzing-windows-pagefilesys-from.html.

[26] Stimson, Jared M. FORENSIC ANALYSIS OF WINDOW’S VIRTUAL MEMORY INCORPORATING THE SYSTEM’S PAGEFILE. Master's thesis. Naval Postgraduate School, Monterey, California. December 2008. Last Accessed: November 2015. http://calhoun.nps.edu/bitstream/handle/10945/3714/08Dec_Stimson.pdf?sequence=1&isAllowed=y.

[27] Haist, Robert. Restoring Windows CMD sessions from pagefile.sys. Forensics blog. Robert Haist. December 2013. Last Accessed: November 2015. http://blog.roberthaist.com/2013/12/restoring-windows-cmd-sessions-from-pagefile-sys-2/.

[28] Robertson, Chad. Indicators of compromise in memory forensics. SANS Gold paper. SANS.org. February 2013. Last Accessed: November 2015. http://www.sans.org/reading-room/whitepapers/forensics/indicators-compromise-memory-forensics-34162.

[29] Garner, George. Getting Started with KnTDD. Software manual. GMG Systems Inc. 2013.

[30] HBGary. Responder Field Edition: Complete Windows Memory Investigation Suite. Information pamphlet. HBGary Inc. Unknown date. Last Accessed: November 2015. http://lib.store.yahoo.net/lib/easycdduplication/responderfield.pdf.

[31] Vidas, Timothy. The Acquisition And Analysis Of Random Access Memory. Technical paper. Journal of Digital Forensic Practice, Volume 1, Number 1, 2006. Last Accessed: November 2015. http://users.ece.cmu.edu/~tvidas/papers/JDFP06.pdf.

[32] Florio, Elia. When Malware Meets Rootkits. White Paper. Symantec Security Response. Dublin. 2005. Last Accessed: November 2015. https://www.symantec.com/avcenter/reference/when.malware.meets.rootkits.pdf.

[33] Butler, Jamie. Fu.cpp source code. Source code. Codeforge.com. December 2003. Last Accessed: November 2015. http://www.codeforge.com/read/3533/fu.cpp__htmlhttp://www.rootkit.com/project.php?id=12.

[34] Zhang R, Wang L, Zhang S. Windows Memory Analysis Based on KPCR. Technical paper. Proceedings of the Fifth International Conference on Information Assurance and Security. 2009. Last Accessed: November 2015. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5284273.

[35] Zhang, S et al. Exploratory Study on Memory Analysis of Windows 7 Operating System. Technical paper. Proceedings of the Third International Conference on Advanced Computer Theory and Engineering (ICACTE). 2010. Last Accessed: November 2015. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5579832.

[36] Balzarottia, Davide et al. The Impact of GPU-Assisted Malware on Memory Forensics: A Case Study. Technical paper. Digital Investigation. Volume 14. Supplement 1. 2015. Last Accessed: November 2015. http://ac.els-cdn.com/S1742287615000559/1-s2.0-S1742287615000559-main.pdf?_tid=8617ce3a-81aa-11e5-ae27-00000aab0f6b&acdnat=1446500689_05b7ca8f6427c2ce0eb2d80057d489e2.

[37] Haruyama, Takahiro and Suzuki, Hiroshi. One-byte Modification for Breaking Memory Forensic Analysis. Presentation. Presented at BlackHat Netherlands, 2012. 2012. Last Accessed: November 2015. https://media.blackhat.com/bh-eu-12/Haruyama/bh-eu-12-Haruyama-Memory_Forensic-Slides.pdf.

[38] Milković, Luka. Defeating Windows memory forensics 29c3. Presentation. Presented at Chaos Communication Congress in Hamburg, 2012. December 2012. Last Accessed: November 2015. http://code.google.com/p/dementia-forensics/downloads/detail?name=Defeating%20Windows%20memory%20forensics.pdf.

[39] Williams, Jake and Torres, Alissa. ADD -- Complicating Memory Forensics Through Memory Disarray. Presentation. Schmoocon 2014. 2014. Last Accessed: November 2015. http://www.mediafire.com/view/h7bmcscbtyaeb6r/ADD_Shmoocon.pdf.

[40] Scroll lock. Wikipedia, The Free Encyclopedia. Last modified: August 16, 2015. Last Accessed: November 2015. https://en.wikipedia.org/wiki/Scroll_lock.

[41] Microsoft Corporation. Forcing a System Crash from the Keyboard. Online informational web page. Microsoft Corporation. Unknown date. Last Accessed: November 2015. https://msdn.microsoft.com/en-us/library/windows/hardware/ff545499(v=vs.85).aspx.

[42] Carbone, Richard. Malware memory analysis of the IVYL Linux rootkit: Investigating a publicly available Linux rootkit using the Volatility memory analysis framework. Scientific Report. DRDC. DRDC-RDDC-2015-R060. April 2015. Last Accessed: November 2015. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc189/p801882_A1b.pdf.

[43] Carbone, Richard. Malware memory analysis of the Jynx2 Linux rootkit (Part 1): Investigating a publicly available Linux rootkit using the Volatility memory analysis framework. Scientific Report. Defence R&D Canada. DRDC-RDDC-2014-R176. October 2014. Last Accessed: November 2015. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc197/p800829_A1b.pdf.

[44] Carbone, Richard. Memory analysis of the KBeast Linux rootkit: Investigating publicly available Linux rootkit using the Volatility memory analysis framework. Scientific Report. DRDC. DRDC-RDDC-2015-R064. June 2015. Last Accessed: November 2015. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc199/p801869_A1b.pdf.

[45] Microsoft Corporation. How to Clear the Widows Page File at Shutdown. Article ID: 314834. Microsoft support article. Microsoft Corp. July 2010. Last Accessed: November 2015. https://support.microsoft.com/en-ca/kb/314834.

[46] Jamie Levy. Linux – Linux Profiles. Documentation web page. Volatility Foundation. 2015. Last Accessed: November 2016. https://github.com/volatilityfoundation/volatility/wiki/Linux.

[47] Raytheon. SecondLook User Manual. User Manual. Raytheon. 2014.

[48] DFRWS. DFRWS 2005 Forensics Challenge. Forensics challenge. DFRWS. Last accessed: 2005. http://old.dfrws.org/2005/challenge/.

[49] National Software Reference Library. Wikipedia, The Free Encyclopedia. January 2015. Last accessed: January 2016. https://en.wikipedia.org/wiki/National_Software_Reference_Library.

[50] LexisNexis. NIST (NSRL) Filter. How-to. LexisNexis. 2016. http://help.lexisnexis.com/litigation/ac/law/law_6/nist(nsrl)_filter.htm.

[51] AccessData Group LLC. AccessData KFF Installation Guide. Technical guide. AccessData Group LLC. Last accessed: 2013. https://ad-pdf.s3.amazonaws.com/Kff%20Install%20Guide.pdf.

[52] Dias, Ricardo. Intelligence-Driven Incident Response with YARA. SANS Gold whitepaper. SANS.org. Last accessed: October 2014. https://www.sans.org/reading-room/whitepapers/forensics/intelligence-driven-incident-response-yara-35542.

[53] Carbone, Richard, Salois, Martin and Bean, Christina. An in-depth analysis of the cold boot attack: Can it be used for sound forensic memory acquisition? Technical Memorandum. DRDC Valcartier TM 2010-296. Defence R&D Canada. January 2011. Last accessed: November 2015. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc188/p534323_A1b.pdf.

Bibliography

Adair, S; Harstein, B.; Richard, M. and Ligh, M. Hale. Tools And Techniques For Fighting Malicious Code: Malware Analysts Cookbook And DVD. Book. Wiley & Sons. 2010. ISBN-13: 978-8126529261.

Carbone, R. and Bourdon-Richard, S. The definitive guide to Linux-based live memory acquisition tools: An addendum to "State of the art concerning memory acquisition software: A detailed examination of Linux, BSD and Solaris live memory acquisition". Technical Memorandum. Defence R&D Canada. DRDC Valcartier TM 2012-319. September 2013.

Carbone, R. and Rheaume, F. Malware memory analysis for non-specialists - Hints and tipcs for Windows users. Scientific letter. DRDC. DRDC-RDDC-2014-L207. September 2014.

Carbone, R.; Bean, C. and Salois, M. An in-depth analysis of the cold boot attack: Can it be used for sound forensics memory acquisition? Technical Memorandum. Defence R&D Canada. DRDC Valcartier TM 2010-209. January 2011.

Carvey, H. Windows Forensic Analysis (DVD Toolkit). Book. Syngress Publishing. Edition 2E. 2009. ISBN-13: 978-1-59749-422-9.

Charland, P; Nadeau, F. and Carbone, R. Remote Memory Acquisition. Scientific Letter. DRDC. DRDC-RDDC-2015-L137. May 2015.

Kornblum, J. and Torres, A. Memory Forensics In-Depth. Training course. SANS. 2014.

Ligh, M. Hale; Case, A.; Levy, J. and Walters, A. Art of Memory Forensics. Book. Wiley & Sons. 1st Edition. 2014. ISBN-13: 978-1-118-82509-9.

Malin, C. H.; Casey, E. and Aquilina, J. M. Malware Forensics Field Guide for Windows Systems: Digital Forensics Field Guides. Book. Elsevier. 2012. ISBN-978-1-59749-472-4.

Malin, C. H.; Casey, E. and Aquilina, J. M. Malware Forensics Field Guide for Linux Systems: Digital Forensics Field Guides. Book. Elsevier. 2012. ISBN-978-1-59749-470-0.

Sikorshi, M and Honig, A. Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. Book. No Starch Press. 2012. ISBN-13: 978-1593272906.

List of symbols/abbreviations/acronyms/initialisms

ADD	Attention Deficit Disorder
APT	Advanced Persistent Threat
AV	Antivirus (or Anti-Virus)
BSOD	Blue Screen Of Death
CAF	Canadian Armed Forces
CCN	Credit Card Number
CFNOC	Canadian Forces Network Operation Centre
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DFRWS	Digital FoRensic WorkShop
DKOM	Direct Kernel Object Manipulation
ELF	Executable and Linkable Format
FTK	Forensic ToolKit
GB	Gigabyte (1×10^9 bytes)
GiB	GiB (2^{30} bytes)
GPU	Graphics (or Graphical) Processing Unit
IP	Internet Protocol
KFF	Known File Filter
KPCR	Kernel Processor Control Region
MD5	Message Digest 5
NIST	National Institute of Standards and Technology
NSRL	National Software Reference Library
OpenCL	Open Computing Language
PE	Portable Executable
PoC	Proof of Concept
RAM	Random Access Memory
RCMP	Royal Canadian Mounted Police
SHA-1	Secure Hash Algorithm-1

SIN	Social Insurance Number
SP1 / SP2	Service Pack 1 / 2
SSN	Social Security Number
USB	Universal Serial Bus
VMEM	Virtual Memory VMware file

DOCUMENT CONTROL DATA

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g., Centre sponsoring a contractor's report, or tasking agency, are entered in Section 8.)

DRDC – Valcartier Research Centre
Defence Research and Development Canada
2459 route de la Bravoure
Quebec (Quebec) G3J 1X5
Canada

2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.)

UNCLASSIFIED

2b. CONTROLLED GOODS

(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC DECEMBER 2012

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Windows memory analysis issues and Linux memory analysis footnotes

4. AUTHORS (last name, followed by initials – ranks, titles, etc., not to be used)

Carbone, R.

5. DATE OF PUBLICATION (Month and year of publication of document.)

January 2017

6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)

32

6b. NO. OF REFS (Total cited in document.)

65

7. DESCRIPTIVE NOTES (The category of the document, e.g., technical report, technical note or memorandum. If appropriate, enter the type of report, e.g., interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Scientific Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)

005A PASS RCMP "Live Forensics" Collaboration

9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DRDC-RDDC-2017-R004

10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

The purpose of this work was to identify and document the various issues that in the opinion of the author remain concerning Windows memory analysis. Minor Linux specific memory analysis issues are also discussed. Too often, publicly available memory analysis specific case studies, analyses, books, guides and how-to gloss over analysis problems including current limitations, pitfalls and caveats. Finding documentation discussing these issues is problematic as no single useful source could be found after multiple searches. Because of this, and based on the work already conducted by the author in his public and private memory analysis case studies, this report highlights and examines the more important remaining memory analysis issues. The author proposes a very short manual methodology for analysing damaged or corrupted memory images.

Ces travaux visaient à cerner et à documenter différents problèmes qui subsistent, selon l’auteur, au sujet de l’analyse de la mémoire de Windows. On discute aussi de problèmes mineurs liés à l’analyse de la mémoire dans Linux. Trop souvent, les études de cas spécifiques, les analyses, les livres, les guides et les savoir-faire d’analyse de la mémoire offerts au public font abstraction des problèmes d’analyse, y compris les limites, les pièges et les mises en garde actuels. Il est difficile de trouver de la documentation qui aborde ces problèmes, car il a été impossible de trouver une seule source utile après de multiples recherches. Par conséquent, et à la lumière des travaux déjà effectués par l’auteur dans ses études de cas d’analyse de mémoire publiques et privées, le présent rapport souligne et examine les principaux problèmes qui persistent dans l’analyse de la mémoire. Enfin, l’auteur propose une méthodologie manuelle très courte permettant d’analyser les images mémoire endommagées ou altérées.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g., Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Forensics, Computer memory; Linux; Memory analysis; Memory forensics; Windows