Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

State of the art concerning memory acquisition software

A detailed examination of Linux, BSD and Solaris live memory acquisition

Richard Carbone
Certified Hacking Forensic Investigator (EC-Council CHFI)
Certified Incident Handler (SANS)
DRDC Valcartier

Defence R&D Canada – Valcartier
Technical Memorandum
DRDC Valcartier TM 2012-008
March 2012

Canada

State of the art concerning memory acquisition software

A detailed examination of Linux, BSD and Solaris live memory acquisition

Richard Carbone
Certified Hacking Forensic Investigator (EC-Council CHFI)
Certified Incident Handler (SANS)
DRDC Valcartier

Defence R&D Canada – Valcartier

Technical Memorandum
DRDC Valcartier TM 2012-008
March 2012

Principal Author

Original signed by Richard Carbone

Richard Carbone

Programmer/Analyst

Approved by

Original signed by Guy Turcotte

Guy Turcotte

Head/System of Systems Section

Approved for release by

Original signed by Christian Carrier

Christian Carrier

Chief Scientist

Abstract

This technical memorandum examines various software tools that can be used for carrying out forensic memory acquisition against various Linux, BSD, and Solaris x86-based systems. No comparable work could be found in the publicly available literature after an exhaustive survey of the subject matter. This current study is important as these UNIX systems are pervasive in today's modern world and are found in a variety of IT environments ranging from the home to corporate data centres. By addressing the pertinence of x86-based UNIX system memory acquisition the computer forensic investigator will be empowered with the necessary knowledge and techniques required to readily tap into this important avenue of potentially useful evidence. Two tools stand out above the rest, Second Look and Fmem, both of which succeeded in all experiments at capturing the underlying system's memory. Although some of the other tools examined herein had specific strengths, they did not work as expected in all instances.

Résumé

Ce mémorandum technique examine divers outils logiciels qui peuvent être utilisés pour l'acquisition inforensique de la mémoire de divers systèmes Linux, BSD et Solaris basés sur x86. Aucun travail comparable n’a pu être trouvé dans la littérature publique après une enquête exhaustive sur le sujet. Cette étude est pertinents car ces systèmes UNIX sont omniprésents dans notre monde moderne et se retrouvent dans une variété d'environnements informatiques allant de la maison aux centres de données d'entreprise. En abordant l'importance de l’acquisition de la mémoire des systèmes UNIX x86, l'enquêteur inforensique sera doté des connaissances et des techniques nécessaires pour puiser plus facilement dans cette importante source de preuves potentiellement utiles. Deux outils se distinguent, Second Look et Fmem, qui ont chacun réussi à capter la mémoire du système sous-jacent dans toutes les expérimentations. Bien que certains des autres outils examinés aient certains autres atouts spécifiques, ils n’ont pas fonctionné comme prévu dans tous les cas.

This page intentionally left blank.

Executive summary

State of the art concerning memory acquisition software: A detailed examination of Linux, BSD and Solaris live memory acquisition

Carbone, R.; DRDC Valcartier TM 2012-008; Defence R&D Canada – Valcartier; March 2012.

This technical memorandum is the second in a set of three. This memorandum's objective is to examine the technical aspects surrounding the forensic memory acquisition of x86-based UNIX systems. These systems include the various BSD systems (FreeBSD, NetBSD, and OpenBSD, 32 and 64-bit versions, respectively), Solaris (32 and 64-bit versions) and Linux (Red Hat (32-bit only), Fedora Core 15 and Ubuntu 11.04, 32 and 64-bit versions, respectively). The first memorandum focused on memory acquisition from DOS and Windows '9X systems. The third and final memorandum in this series will focus exclusively on memory acquisition from NT-based 32 and 64-bit operating systems ranging from Windows NT up to Windows 7 and Windows Server 2008 R2.

Although much effort has been made these last couple of years by the digital forensics community into techniques concerning Windows memory acquisition, little public literature is available which addresses the issue for UNIX systems. As such, this technical memorandum endeavours to address it specifically.

Based on the work carried out herein, the reader should be able to use the same tools and techniques employed by the author in order to acquire computer memory against the aforementioned systems. However, while conducting memory acquisition experiments against the aforementioned operating systems, important memory-specific limitations were encountered, both the amount of memory supported by these systems and the amount of memory that could be acquired using these tools.

Nevertheless, while each memory acquisition tool examined had specific strengths and weaknesses, two stood above the rest, Second Look and Fmem. Both tools, although Linux specific, worked as expected against modern Linux-based systems.

This work was carried out over a period of several months as part of the Live Computer Forensics project, an agreement between DRDC Valcartier and the RCMP (SRE-09-015, 31XF20). The results of this project will also be of great interest to the Canadian Forces Network Operation Centre (CFNOC) in their mission of securing DND networks and investigating computer incidents.

Sommaire

State of the art concerning memory acquisition software: A detailed examination of Linux, BSD and Solaris live memory acquisition

Carbone, R. ; DRDC Valcartier TM 2012-008 ; R & D pour la défense Canada – Valcartier; mars 2012.

Ce mémorandum technique est le deuxième d'une série de trois. L'objectif de ce mémorandum est d'examiner les aspects techniques liés à l'acquisition inforensique de la mémoire des systèmes UNIX x86. Ces systèmes comprennent les différents systèmes BSD (FreeBSD, NetBSD et OpenBSD, version 32 et 64 bit, respectivement), Solaris (32 et 64 bits) et Linux (Red Hat (32 bits seulement), Fedora Core 15 et Ubuntu 11.04, versions 32 et 64-bit, respectivement). Le premier mémorandum portait sur l'acquisition de la mémoire des systèmes DOS et Windows '9x. Le troisième et dernier mémorandum de cette série se concentrera exclusivement sur l'acquisition de mémoire de systèmes NT 32 bit et les systèmes d'exploitation 64-bits allant de Windows NT à Windows 7 et Windows Server 2008 R2.

Bien que beaucoup d'efforts aient été consacrés par la communauté d'inforensique ces dernières années aux techniques relatives à l'acquisition de la mémoire Windows, peu d'ouvrages publics sont disponibles qui abordent la question pour les systèmes UNIX. À ce titre, ce mémorandum technique s'efforce de répondre spécifiquement à ce besoin.

Basé sur le travail décrit dans ce document, le lecteur devrait être en mesure d'utiliser avec succès les mêmes outils et techniques mises en œuvre par l'auteur pour acquérir la mémoire des systèmes susmentionnés. Cependant, tout en menant des expériences d'acquisition de la mémoire des systèmes d'exploitation susmentionnés, des limites importantes liées à la mémoire ont été découvertes, à la fois quant à la quantité de mémoire prise en charge par ces systèmes et à la quantité de mémoire qui pourrait être acquise en utilisant les différents outils.

Néanmoins, bien que chaque outil d'acquisition de mémoire examiné dispose de forces et faiblesses spécifiques, deux se distinguent du reste, Second Look et Fmem. Ces deux outils, spécifiques à Linux, ont fonctionné comme prévu pour les systèmes Linux récents.

Ce travail a été réalisé sur une période de quelques mois dans le cadre du projet Live Computer Forensics, une entente entre RDDC Valcartier et la GRC (SRE-09-015, 31XF20). Les résultats de ce projet seront aussi d'un grand intérêt pour le Centre d'opérations des réseaux des Forces canadiennes (CORFC) dans leur mission de sécurisation des réseaux du MDN et d'enquêtes sur les incidents informatiques.

Table of contents

List of tables

Acknowledgements

The author would like to thank Mr. Yves van Chestein for peer reviewing this document and providing many useful comments and insight in order to improve it. The author would also like to thank Mr. Martin Salois for translating various portions of the text.

Disclaimer and use policy

The reader should neither construe nor interpret the work described herein by the author as an endorsement of the aforementioned techniques and capacities as suitable for any specific purpose, construed, implied or otherwise.

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

Finally, the author, the Government of Canada, the Minister of National Defence (Canada), the Department of National Defence (Canada) and Defence Research & Development Canada are henceforth absolved of all wrongdoing, whether intentional, unintentional, construed or misunderstood on the part of the reader. If the reader does not agree to these terms then this technical memorandum should be readily returned to the Department of National Defence (Canada). Only if the reader agrees to these terms should he or she continue in reading it beyond this point. It is further assumed by all participants that if the reader has not read said Disclaimer upon reading this technical memorandum 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.

Requirements, assumptions and exclusions

It is assumed that the reader is altogether familiar with digital forensics and the various techniques and methodologies associated thereto. This technical memorandum is not an introduction to digital forensics, its techniques or methodologies. However, this technical memorandum will endeavour to present an adequate technically oriented background to enable the reader to carry out and implement the work and analysis conducted herein.

This present work only examines x86-based UNIX systems. Other memorandums, either written or to be completed, examine the memory acquisition of DOS and Windows '9x systems and modern Windows NT-based operating systems, respectively.

This endeavour has been conducted primarily using a Linux-based system while secondary result validation was carried out using a Windows 7-based system. As such, regardless of the reader's own specific set-up, he reader should arrive at the same overall results as those presented herein, assuming that the guest operating systems are similarly configured and that the same virtualisation technology is used. All operating systems tested upon have been fully virtualised.

The primary Linux system was running Fedora Core 14 64-bit atop a Dell Precision 690 workstation with dual-core Xeon processors (with HyperThreading) providing 8 logical cores in conjunction with 22 GiB RAM and almost 20 TB of disk storage (see Annex A.1 for more details). The validation system was a Windows 7 64-bit system running atop a Dell XPS i7 8-logical core system with 18 GiB and 1.5 TB of disk storage (see Annex A.1 for more details).

All guest operating systems (e.g. BSD, Linux, and Solaris) were tested under Oracle VirtualBox 4.1.0 (Linux and Windows version) with the VirtualBox Extension Pack version 4.1.0 installed. VMware Workstation, another very popular choice for operating system virtualisation, was not used so that the reader would have not to rely on commercially licensed software in order to validate the results obtained by the author.

It is important to emphasize that should memory acquisition of a physical, non-virtualised x86-based system fail, it may be possible to acquire that system's memory using the cold boot attack [1] since the technique is not affected by the underlying operating system. However, the success of the cold boot attack should be considered as experimental at best [2]. As such, the investigator must be realistic in his expectations for acquiring memory using this technique.

Forensically capturing memory

Physically capturing memory under BSD, Linux and Solaris is not particularly difficult. All modern versions of these systems fully support USB mass storage devices. However, correctly recognizing these devices and mounting them may at times be precarious, depending on the underlying operating system. These specifics, however, are too long to fully enumerate here.

Of course, investigators need not be confined only to USB mass storage. Modern BSD and Linux operating systems fully support FireWire standards 1394a and 1394b. However, the level of support offered by the underlying operating system is entirely dependent on its maturity. Although Solaris supports FireWire, its support should not be considered reliable, at least as of the time of this writing, and as such, USB mass storage is strongly suggested over the use of FireWire when working with this specific operating system.

The investigator should bear in mind that older BSD, Linux and Solaris systems might not adequately support USB or FireWire devices as their support for these devices is directly related to the operating system's maturity.

If the investigator finds that USB or FireWire-based mass storage device support is either not available in the operating system or has been disabled in the BIOS, then memory acquisition could be carried out over the network using NFS. Since many of these systems find themselves in networked environments, NFS is a workable solution. It is important to be aware that older versions of NFS have readily attainable upper file size limits. Newer implementations of NFS including versions 3 and 4 support 64-bit file sizes but differing operating systems may impose other unknown constraints.

The use of NFS requires that the both the target system and a remote system be configured to support NFS. Generally, even in systems that do not have NFS enabled by default it is not that difficult to get it working and should not require any operating system reboots. However, operating system variations may again impose differing limitations.

The use of Windows file sharing atop UNIX operating systems is not examined herein although it is possible. The software components required for this may necessitate software or component installation since they are generally not considered an integral part of most UNIX operating systems.

Although swap space is used by most UNIX and UNIX-like systems, it is not acquired within this work or in any of the various experiments. Unlike physical computer memory (RAM), swap space acquisition is carried out in the same manner as any other disk partition (from a live system). As such, in so long as the investigator has root privileges he can readily acquire all of a system's swap space.

About the report

This technical memorandum has been written for the computer forensic investigator who may have to perform a UNIX-based memory acquisition at one time or another in the function of his or her duties. Although information concerning this subject matter can be found in various locations across the Internet, it is generally disconnected and as such, the author has attempted to bring this information together for the reader in a comprehensible manner.

This technical memorandum is not, however, an examination of computer memory analysis. This specific vein of research is outside the scope of this work and warrants an altogether separate technical discussion in order to sufficiently examine the subject matter.

Computer memory volatility

It is important to consider the volatility of computer memory when attempting to acquire it. Furthermore, it does not matter if the computer system is running DOS, UNIX, Windows, or any other operating system. The fact that an individual, in this case a computer forensic investigator, runs a memory acquisition program atop a given computer system changes the state of the underlying system. This is a universal principle and follows through for all cases where a physical intervention is made on against a given information system [41, 42].

In the case of computer memory acquisition, in order to obtain a copy of the system's memory, the investigator must interact with the system (in order to observe it) and then run some program, command, or utility in order to acquire its memory. This process irreversibly changes the running state of the computer system and as such, certain bytes of information containing evidence may be permanently lost. However, it is logical to conclude that the more memory a given system has the less likely this is to occur, as evidence is likely to be spread out across said memory. However, no matter the care and consistency of the steps used by the investigator, some data will be lost with no way of discerning what it was.

Ultimately, however, computer memory acquisitions acquired through diligence should hold up to court-based challenges, in so long as the investigator understands the actions he carried out and their potential impact and can explain the acquisition tool's underlying functionality. This is, of course, where open source software shines in contrast to closed proprietary acquisition software [43].

This page intentionally left blank.

1 Background

1.1 Objective

The objective of this technical memorandum is to determine which memory acquisition software-based tools work under various x86-based UNIX systems including BSD (FreeBSD, NetBSD and OpenBSD, 32 and 64-bit, respectively), Linux (Red Hat 9 (32-bit only), Fedora Core 15 and Ubuntu 11.04, 32 and 64-bit respectively)) and Solaris (32 and 64-bit). Moreover, it will gauge the amount of memory that each operating system supports including how much of that memory can be acquired. Any discovered limitations or caveats are presented to the reader.

1.2 Operating system background

In this section, a brief technical and biographical background of the aforementioned operating systems is examined.

1.2.1 BSD

This subsection examines the most commonly used non-Mac BSD-based operating systems, including issues surrounding memory allocation, detection and acquisition.

1.2.1.1 FreeBSD

FreeBSD is a direct descendant of 386BSD written by Lynne and William Jolitz. Prior to FreeBSD becoming an actual open source initiative, it was originally a collection of unofficial 386BSD patches. However, when the Jolitz's did not agree with the patch maintainers about the future direction of 386BSD the patches were forked along with the base code of Net/2 BSD including some components from 386BSD to form the FreeBSD initiative. The first official FreeBSD release became publicly available via FTP November 1993. It was also commercially obtainable from Walnut Creek on CD. The first release was a result of collaboration between Jordan Hubbard, Nate Williams and Rodney W. Grimes.

Today, FreeBSD is the most widely used of the free and open source BSD operating systems [3]. Currently at stable release version 8.2, FreeBSD is a stable BSD operating system that has many thousands of additional third-party software available from the FreeBSD software repository. While the current stable release of FreeBSD 8.2 was released February 2011, a more avant-gardist version is available, FreeBSD 9. The newer version is not examined herein.

Today, FreeBSD can be found to be in wide use in various networking appliances. Moreover, it supports various computing architectures beyond standard x86. Specifically, it supports not only the x86 32 and 64-bit platforms but even PowerPC (32 and 64-bit) and UltraSPARC architectures. Although other platforms may be supported by FreeBSD, no official distributions are available for them.

Since FreeBSD is in wide use, it is important that computer forensic investigators consider the possibility that he may eventually have to acquire memory from a FreeBSD computer system.

1.2.1.2 NetBSD

Also a direct descendant of 386BSD and Net/2 BSD, the NetBSD initiative was the result of growing developer frustration with the direction of 386BSD and the acceptance of community-based patches by the Jolitz's. The founders of the NetBSD project included Chris Demetriou, Theo de Raadt, Adam Glass and Charles Hannum. The first official NetBSD release became available April 1993 at version 0.8. Its current incarnation is version 5.1, released November 2010.

The key difference between NetBSD and FreeBSD is that the former is a truly multi-platform, multi-architecture operating system. It places emphasis on scalability, quality and security-related features. Moreover, NetBSD has currently been ported to 57 different platforms supporting 15 different processor architectures [23]. Even though it is available for many systems and platforms, it ranks third in terms of BSD usage, behind FreeBSD and OpenBSD, respectively [3].

Thus, due NetBSD's high potential for use on a vast array of computing platforms, it is likely that at some point a computer forensic investigator may have to acquire memory from a NetBSD computer system.

1.2.1.3 OpenBSD

A direct descendant of NetBSD, OpenBSD was forked from NetBSD in 1995 by Theo de Raadt, one of the founding members of NetBSD. Asked to resign from NetBSD, de Raadt started his own BSD initiative based on the available NetBSD source code of that time. The OpenBSD project places a great deal of emphasis on software correctness, resulting in greater security based largely on source code auditing, open and high quality documentation requirements for software and hardware and the use of superior cryptographic technologies. Moreover, OpenBSD has a very restrictive licensing policy that prefers the use of BSD-compatible licenses and rejects all licenses that are not compatible with OpenBSD's desire to maintain openness and transparency. [24]

Currently, OpenBSD has been ported to 17 different platforms, far less than NetBSD but more than FreeBSD. OpenBSD is currently the second most widely used open source BSD operating system [3]. Due to OpenBSD's highly secure nature, many third-party network-based security products are based on it. The very first release of OpenBSD, version 1.2, first became available July 1996. The current stable version, 4.9, was released May 2011.

Due to OpenBSD's proliferation in hardware-centric networking solutions given its ability to satiate niche requirements including secure web, FTP and DNS servers, it follows that while investigating cyber-crime an investigator may eventually be required to acquire the memory from a system running OpenBSD.

1.2.1.4 BSD-specific memory issues

Different BSD operating systems have differing memory capabilities that are important to understand prior to examining the results from the various memory acquisition experiments carried out against these systems herein.

The first issue to examine is BSD's support for PAE. PAE-enabled processors allow 32-bit operating systems to address memory using a 36-bit memory addressing scheme rather than the standard 32-bit scheme [4]. As such, 36-bit memory addressing allows the operating system to address up to a maximum of 62 GiB[1] RAM, many times more memory than the standard 4 GiB RAM permitted by 32-bit memory addressing [7]. However, not all BSD systems support PAE equally.

Consider that while FreeBSD fully supports PAE it is not enabled by default in the distribution release of its kernel [4, 5]. Instead, in order for a FreeBSD system to support PAE its kernel must be recompiled with the appropriate configuration options [4, 5]. OpenBSD, on the other hand does not currently support PAE and it is unknown if it ever will. Although NetBSD supports PAE, it too is disabled by default in its distribution kernel and as such must be recompiled with support enabled [4, 6].

BSD 64-bit memory was also found to be equally inconsistent. Verification of the guest operating system configuration details found in Annex B.2 confirm that both FreeBSD and NetBSD 64-bit operating system kernels fully support 64-bit memory addressing, based on the amounts of memory detected by these operating systems. However, OpenBSD, which currently fails at supporting PAE, also does not support 64-bit memory through its default 64-bit kernel provided with the 64-bit distribution. Thus, in order for OpenBSD to support 64-bit memory addressing its kernel must be recompiled with the appropriate options.

1.2.1.5 BSD memory acquisition

As examined in the previous section, different BSD operating systems support differing amounts of memory. Although all are capable of supporting 64-bit memory addressing, some require kernel recompilation. Moreover, even though PAE support is available for some systems it has not been compiled in with their default kernels.

As such, all memory acquisition experiments conducted herein against all three of the aforementioned BSD operating systems will be done against their default kernels[2]. No kernel recompilation of any kind was carried out.

Under BSD, several accessible system devices (found under */dev*) can be used for memory acquisition. All BSD operating systems examined in this work (OpenBSD, FreeBSD and NetBSD) support */dev/mem* and */dev/kmem* memory devices. These devices are the preferred two

---

1 36-bit memory addressing can access up to a maximum of 64 GiB RAM although the last 2 GiB RAM are reserved while the first 62 GiB RAM can be used for main memory [7].

2 The kernels of FreeBSD and NetBSD have not been recompiled with PAE support for the experiments carried out in this work nor has OpenBSD's 64-bit kernel been recompiled to support 64-bit memory addressing.

methods for acquiring kernel or system memory without having to write specialized software drivers. In order to access directly these devices the investigator must have root privileges. [25, 26 and 27]

However, */dev/mem* and */dev/kmem* are not the same memory device. Device */dev/mem* is a direct interface to physical memory while */dev/kmem* provides access to the kernel's virtual memory address space. However, on systems supporting both it is suggested that */dev/mem* be read from prior to attempting */dev/kmem,* which may not work or be stable in all situations. [28]

Finally, */dev/ksyms*, available under NetBSD and OpenBSD, are representations of the kernel's symbol table and kernel modules[3] and as such cannot be used for memory acquisition[4]. Although according to FreeBSD 8.2 32 and 64-bit Man files, */dev/ksyms* is supported, no trace of it could be found under these systems. This device provides the same functionality as Linux's */proc/ksyms* and */proc/kallsyms* kernel pseudo-files and Solaris' */dev/ksyms*. [16, 22, 25, 26, 27, 30 and 36]

1.2.2 Linux

In this section a brief examination of various Linux operating systems and the various issues surrounding memory allocation, detection, and acquisition will ensue.

1.2.2.1 Red Hat Linux

At one time Red Hat Linux was one of the most popular Linux distributions around and was in head-to-head competition with Caldera Linux[5]. However, Red Hat Linux was finally discontinued in 2004. The last official version released was Red Hat 9 and was freely available for public for download as of March 2003. It could be readily downloaded by anyone with sufficient bandwidth. Red Hat, like Fedora Core, is an RPM-based Linux distribution.

Shortly after the demise of Red Hat Linux 9, Red Hat Inc. came out with a new fee-only version of Red Hat Linux, specifically Red Hat Enterprise Linux. Due to the demise of Red Hat Linux 9, Fedora Core Linux was created as a free replacement for it.

Red Hat Linux has always been 32-bit based and was never offered as a 64-bit distribution. However, version 9 did come bundled with a 32-bit PAE kernel that could support larger memory options (up to 64 GiB RAM). At the time of Red Hat 9's demise, it continued to remain with the 2.4.x generation of Linux kernels. However, due to its popularity it is still likely that some web, FTP, and other network servers may continue to be powered by older Red Hat Linux-based systems such as version 9. As such, it is beneficial that the investigator be aware of both its existence and be knowledgeable in acquiring its memory.

---

3 Only FreeBSD and NetBSD support LKMs [23, 37]. No valid information concerning OpenBSD LKMs can be found. Systems that do not support LKMs load all their compiled-in kernel modules at boot time.

4 It may have some value for manual memory analysis although this is not examined in this work.

5 Caldera Linux is a long-defunct Linux distribution [6].

1.2.2.2 Fedora/Fedora Core Linux

Generating no revenue, Fedora Core Linux is developed in its entirety by the open source community, although it is sponsored by Red Hat Inc. Having taken up the mantle of Red Hat Linux, it continues to be freely available. Moreover, it comes available in both 32-bit and 64-bit flavours. Specifically, the 32-bit version is by default a 32-bit PAE kernel, although the user has the option of installing a 32-but kernel instead. The 64-bit kernel is exclusively bundled with the 64-bit distribution. Moreover, Fedora Core is also a RPM-based distribution.

Despite being very popular, it continues to remain behind Ubuntu and Mint Linux in terms of its adoption [8]. The distribution is considered to a technology adoption leader as it continuously incorporates new capabilities as they become available in subsequent distribution releases. The release schedule of Fedora is approximately every six months.

The current release is at version 15 and was made publicly available May 2011. The very first version of Fedora was Fedora Core Linux 1 which was released November 2003, just several months before Red Hat 9 became end-of-life. The history of Fedora Linux, developed by the community under the umbrella of the Fedora Project, is somewhat convoluted. It is noteworthy to mention that all versions of Fedora Linux prior to version 7 are known as Fedora Core Linux while those as of version 7 are known as Fedora Linux.

While Red Hat continues to invest and support the Fedora Project, it uses it as a testing ground for assessing technologies that may eventually be incorporated into Red Hat's Enterprise Linux products.

1.2.2.3 Ubuntu Linux

Ubuntu Linux is a Debian-based Linux operating system. The Ubuntu initiative is sponsored by UK technology company Canonical, owned by South African Mark Shuttleworth. Unlike Fedora, which is commercially sponsored but generates no actual revenue Ubuntu generates revenue by providing Ubuntu-related technical support and services. However, the Ubuntu operating system itself is entirely free of charge.

Perhaps due to the Ubuntu philosophy it is currently the most popular Linux desktop in use today [15]. Its first release was October 2004 and has a release schedule approximately every six months. The current release, version 11.04, was released April 2011.

It is available as both a 32-bit and 64-bit operating system and is very desktop friendly, more so than many other Linux distributions. By default, the 32-bit distribution does not provide a PAE-based kernel although compiled versions of PAE kernels are available from the Ubuntu repository for installation. Moreover, while Ubuntu is largely based upon Debian-based packages, some closed source programs and drivers are bundled with it while others are available through its large Debian-based software repository.

The Ubuntu Foundation, founded July 2005, ensures that Ubuntu will continue to remain a well-funded Linux distribution in order for the community to continue developing and supporting it. Ubuntu has also become actively involved in recent cloud computing initiatives by providing specific cloud-based technologies in its latest Ubuntu Server release.

1.2.2.4 Linux-specific memory issues

Unlike the various x86-based BSD operating systems, Linux has been PAE-capable since kernel 2.3.23, released October 1999 [4]. Linux has been supporting PAE for almost 12 years now. As such, it is common to find many 32-bit Linux kernels with PAE support compiled directly in, although this varies widely by distribution and user preferences. Some distributions, running 2.4.x or 2.6.x kernels will by default, install a PAE-based kernel. Sometimes it is set as the default bootable kernel while other times it is not. Moreover, some users prefer the use of PAE kernels while others do not. As such, it is difficult to determine whether a given 32-bit Linux operating system supports PAE in its current running state. However, it does make sense to use PAE-enabled kernels on any system running a 32-bit Linux distribution with close to or more than 4 GiB RAM in order to make the most of available resources given 32-bit memory addressing limitations.

PAE-enabled processors allow 32-bit PAE-capable Linux distributions to use a 36-bit memory-addressing scheme thereby allowing the system to address up to a maximum of 62 GiB[6] RAM [4, 7]. However, if a given distribution does not install a PAE-enabled kernel one is usually available from the distribution's software repository, as was the case with Ubuntu Linux whose PAE kernel and corresponding source code headers had to be manually downloaded, installed and whose boot loader had to be reconfigured. However, these are rather trivial and well documented[7] reconfiguration operations.

All 64-bit Linux kernels support more than 4 GiB RAM, unlike certain BSD distributions that by default do not. Verification of the configuration details for the Linux-based guest operating systems examined herein can be found in Annex B.2. From this, it is clearly demonstrated that PAE-enabled kernels do in fact detect and use memory above the 32-bit memory limit (4 GiB RAM). As such, Linux-based memory allocation is straightforward since the amount of memory supported by 32 and 64-bit Linux systems are relatively uniform.

However, direct memory acquisition under Linux is not as straightforward as it was under BSD. The reasons for this are examined below.

1.2.2.5 Linux memory acquisition

Memory acquisition under Linux is not particularly obvious at first glance. Unlike with BSD and Solaris operating systems, modern Linux systems do not give direct memory access anymore, necessitating the need for memory drivers, including but not limited to Second Look and Fmem.

For a variety of reasons, direct access to physical (*/dev/mem*) and kernel memory (*/dev/kmem* and */proc/kcore*, respectively) has been limited. In 2.6.x kernels, the restriction appears to be caused by the *CONFIG_STRICT_DEVMEM* kernel structure [10, 11, 12 and 13], although why memory access is limited under a 2.4.x kernel (at least for Red Hat 9) is not well understood at this time

6 36-bit memory addressing can access up to a maximum of 64 GiB RAM although the last 2 GiB RAM are reserved while the first 62 GiB RAM can be used for main memory [7].

7 The largest and most popular Linux documentation repository is The Linux Documentation Project (see http://tldp.org/ for more details).

since it does not suffer from this restriction. Nevertheless, this memory restriction limits the extent to which *dd* and *Memdump* can be used for memory acquisition.

Modern 2.6.x kernels no longer have a */dev/kmem* memory device although */dev/mem* continues to be present. The former device was removed from 2.6.x kernels due to its prevalence in Linux-based rootkit attacks. Thus, by removing it rootkits could no longer have immediate and direct access to the kernel's memory address space. [10, 11, 12, 13 and 15]

Even though device */dev/mem* continues to be available under Linux, it is generally not possible to acquire memory beyond the first one megabyte of memory due to the aforementioned reasons, at least under Linux kernel 2.6.x. However, under kernel 2.4.x it is possible to acquire significantly more memory, at least under Red Hat 9, but still less than the total amount actually detected by the operating system. Mileage will undoubtedly vary by kernel, its generation (2.4.x or 2.6.x) and distribution. Linux memory acquisition specifics, as determined through experimentation, are available in Annex C.

However, Linux 2.4.x kernels do support both */dev/kmem* and */dev/mem* memory devices. While support for building memory device */dev/kmem* has been removed from some distribution specific 2.6.x kernels, it is often possible to recompile it in. Unfortunately, a concise list of which distributions permit this device's recompilation is not currently available. Notwithstanding this, for distributions shipping with kernels where this option is removed a publicly available kernel patch reactivates this feature for kernel recompilation [14]. Experimentation conducted by the author confirms that Fedora Linux and Ubuntu systems have their */dev/kmem* device disabled but they can be reactivated upon selecting the appropriate kernel-compilation configuration options.

Device */dev/mem* is a direct interface to physical memory while */dev/kmem* provides access to the kernel's virtual address space [29]. Although they are similar, it is advised that where both are present memory be acquired first from */dev/mem* prior to attempting */dev/kmem*, which should only be used in the event the former fails.

Linux provides several kernel-specific structures for accessing system memory. These include */proc/kcore*, */proc/kallsyms* and */proc/ksyms*. Both */proc/kallsyms* and */proc/ksyms* refer to the same kernel symbol table that is used by both the kernel itself and by various LKM modules. As such, this structure has limited value for memory acquisition[8]. However, */proc/ksyms* exists only under 2.4.x kernels whereas */proc/kallsyms* has superseded the former under 2.6.x kernels. Linux kernel pseudo-files */proc/ksyms* and */proc/kallsyms* provide the same functionality as the BSD and Solaris' */dev/ksyms* device. [16, 22, 30 and 36]

On the other hand, while pseudo-file */proc/kcore* is a representation of physical memory, it is stored using the ELF core file format. As such, memory dumps obtained from this pseudo-file are best left for use with the system debugger, GDB. In instances where */dev/mem* or */dev/kmem* are available, they are preferred over */proc/kcore*. The total physical length of memory from kernel structure */proc/kcore* is the size of detected memory[9] plus 4 KiB[10]. However, */proc/kcore* enables the acquisition, at least to some extent, of hardware-reserved computer memory. [16, 36]

---

8 However, this structure may have significantly more value when conducting a manual memory analysis, which is not examined herein.

9 Detected memory size is based on the *MemTotal* value found in kernel pseudo-file */proc/meminfo*.

Direct access to device and kernel structures generally requires that the investigator have root privileges on the target system.

1.2.3 Solaris

In this section a brief examination of Solaris' background and operating system-related memory allocation, detection and acquisition-based issues are examined.

1.2.3.1 Solaris 10

Solaris, originally SunOS, was a BSD derivative operating system that ran on Sun Microsystems' proprietary hardware. Sun Microsystems, co-founded by Bill Joy, a key figure in the history of the development of the BSD operating system, customized it to run atop Sun's hardware platform. SunOS only became known as Solaris when the operating system reached SunOS version 5.

The very first version of SunOS, released in 1983, replaced its predecessor SUN UNIX 0.7 which was released in 1982. SunOS 4.1.x, released in 1991 was the first version of Solaris to be marketed by Sun, which at that time was also being sold as Solaris version 1.x. Although SunOS 4.1.x was still a BSD-based operating system its successor, Solaris 2 (SunOS 5) was compliant with the new UNIX operating system standard UNIX System V Release 4, developed jointly by Unix Systems Laboratories[11] and Sun Microsystems. The latest version of Solaris is currently at version Solaris 11 Express.

Solaris 10 and Solaris 11 Express are not the same operating system, although they are similar. The former is an operating system which has been around for over 6 six years and should be considered enterprise ready as it is very stable, scalable, robust and offers a Common Criteria certified operating system[12]. On the other hand, Solaris 11 Express, while offering many of the same capabilities as Solaris 10 is not yet ready for the enterprise as it brings into play new technologies that have not yet been thoroughly tested. For these reasons, it was decided from the outset of this work that effort would be made to acquire the memory of a Solaris 10 operating system rather than the memory from a Solaris 11 Express-based system. More specifically, Solaris 10 Release 9/10 has been selected for use herein.

Sun Microsystems, purchased by Oracle Corporation in early 2010, continues to develop and market one of the oldest and most successful commercial UNIX operating systems still in use today. This is the primary reason why the author reasoned it appropriate to conduct memory acquisition experiments against it. However, even though Linux has definitely eaten away at its market-share, it is difficult at this time to determine exactly how much of the commercial UNIX server market still runs on Solaris. Nevertheless, since at one time Solaris was the most commercially successful UNIX platform, investigators must be prepared to deal with it in the event they encounter it.

---

10 4 KiB for ELF data structure overhead.
11 This was a division of Bell Labs.
12 The latest Solaris operating system to pass Common Criteria certification was Solaris 10 Release 11/06 that achieved a rating of EAL4+ ALC_FLR.3 [17].

1.2.3.2 Solaris-specific memory issues

The Solaris operating system has been supporting 64-bit SPARC-based systems for many years, since the release of Solaris 7 in 1998. In contrast, Linux was not available for the x86_64 architecture until 2003 and Microsoft's first attempt at a 64-bit operating system was its release of Windows XP 64-Bit Edition in 2001, originally targeted at the Itanium. [7]

As such, it should come as no surprise that modern Solaris systems have few to no discernible memory issues when running atop a 64-bit kernel as compared to specific BSD distributions. Moreover, Solaris first began supporting PAE on Intel as far back as Solaris 7 [4]. However, Sun Microsystems' first port of Solaris to the Intel architecture was implemented atop Solaris 2.1[13] [19, 20]. In contrast, Linux supported PAE as of 1999 [4].

It is important to note that when installing Solaris 10 atop a virtual machine, at least when using VirtualBox, even if that virtual machine is created as a 32-bit system the Solaris installation program will configure it to run with a 64-bit kernel instead of a PAE-enabled 32-bit kernel. Reconfiguring a Solaris virtual machine to boot a 32-bit PAE-based kernel is not a particularly difficult endeavour; it requires modifying the GRUB boot loader and specifying an appropriate location for the kernel [21]. The exact cause as to why the Solaris installation program defaulted to installing a 64-bit kernel on a virtual machine configured for 32-bit use is not entirely understood at this point. It could be a bug with VirtualBox or with Solaris' hardware detection capabilities. However, verification of the Solaris 32-bit guest operating system found in Annex B.2 clearly demonstrates that a PAE-enabled kernel was in fact booted and functioning correctly due to its ability to detect memory above the 32-bit memory limit (4 GiB RAM). Nevertheless, modern versions of x86-based Solaris do not provide 32-bit kernels, only 32-bit PAE kernels.

It was also found that Solaris reported exactly the same amount of computer memory regardless if the operating system was booted with a 64-bit or 32-bit PAE-enabled kernel. However, a true 32-bit kernel without PAE support will not report more than 4 GiB, regardless if the host operating system is Solaris or not. Since only PAE 32-bit kernel was provided with Solaris 10 x86, only it and a 64-bit kernel could be tested. Running the *prtconf* command enables the root user to accurately determine the operating system's detected memory limit, which for both the 32-bit PAE and 64-bit Solaris kernels was 8,192 MiB RAM. However, based on David W. Noon's C code which he made available online, it turns out that both systems actually support 8,191.559 MiB RAM or 8,388,156 KiB, slightly less than the 8,192 MiB reported by the operating system [39].[14]

13 This does not include Sun Microsystems' first attempt at the Intel architecture when it introduced the Sun386i.

14 Actually, if the host Solaris operating system is 64-bit and more than 4 GiB RAM is present, then the program must be compiled as 64-bit in order to accurately describe the size of operating system detected memory as a 32-bit compile version of the program will always be limited to a maximum size of 4 GiB. However, regardless of the manner in which the program was compiled, 32 or 64-bit, it will always display the correct number of memory pages, at least in tests carried out in this work. [39]

1.2.3.3 Solaris memory acquisition

Memory acquisition under Solaris 10 is relatively straightforward, as would be expected from an enterprise class operating system. Firstly, it requires that the investigator have root privileges. The same standard memory device as found under Linux and BSD, */dev/mem*, is also the standard memory device under Solaris and is the primary device to use when acquiring memory from it. [22, 30]

Under Solaris, as with Linux, kernel-based memory device */dev/kmem* also exists. This device is used to interface directly with the kernel's virtual address space and it is entirely without associated I/O device memory space (hardware-reserved memory). As such, this device is not the preferred method for acquiring Solaris memory since I/O device memory is not included in any dump. [22, 30]

One important difference between Linux and Solaris is that under Solaris, its */proc*-based kernel structure does not include any memory related pseudo-files. Instead, a pseudo-file similar to Linux-based */proc/kcore* kernel structure is found under Solaris as device */dev/allkmem* and it provides direct kernel virtual address space including all I/O device memory space. This device should only be used in the event memory acquisition from */dev/mem* fails. [22, 30]

The Solaris-based memory device */dev/physmem* is a diagnostic driver used by Solaris software-based system troubleshooting suites to test physical memory and is not to be used for memory acquisition, at least not without writing an appropriate software application that can take advantage of this driver's ability to interface directly with physical memory hardware. [30, 31]

It is important for investigators to note that attempting to acquire more than 4 GiB RAM in a 32-bit Solaris environment (with or without PAE) requires that the acquisition program use either a *read()* call or *llseek()* and *read()* calls. The latter two calls must specifically point to memory locations above the 4 GiB memory limitation in order to acquire higher-order memory. Otherwise, the acquisition program must use the *pread64()* call which is a 64-bit file descriptor *read()*-based call. This is especially applicable to the *Memdump* program which uses 32-bit *read()* calls. However, where 64-bit operating systems are concerned and 64-bit *read()* calls are used this is no longer an issue[15]. Specifics are available in Annex C. However, the *Memdump* program has not been modified herein for its use under Solaris 32-bit PAE. [22, 30]

Finally, */dev/ksyms* is a representation of the kernel's symbol table and as such cannot be used for memory acquisition. It is similar in functionality to BSD's */dev/ksyms* and Linux's */proc/ksyms* and */proc/kallsyms* device and kernel structures, respectively. [16, 22, 30 and 36]

1.2.4 A brief note about UNIX memory devices */dev/mem* and */dev/kmem*

UNIX memory device */dev/mem* is found under most major UNIX systems including all BSD, Linux and Solaris operating systems. This memory device is a direct interface to physical

15 Compiling *Memdump* using a 64-bit compiler (at least with GCC and Sun C/C++ native compilers) will automatically use the appropriate 64-bit system call implementation. However, when using a 32-bit compiler, appropriate compiler-based parameters can be used to force 64-bit compilation, but only if this feature is supported by the underlying compiler.

memory including all hardware associated I/O devices and their associated operating system accessible hardware memory. Although memory device */dev/kmem* is similar to /*dev*/*mem*, it does not provide direct physical memory access; instead, it provides an interface to the kernel's virtual address space. It is important to understand, however, that the kernel's virtual address space memory is similar to physical memory except that no hardware device's I/O memory will be accessible through the operating system using this interface.

Moreover, while */dev/mem* represents actual physical byte offsets from physical memory */dev/kmem* does not. Instead, it is representative only of the byte offsets in the kernel's memory space.

Thus, when attempting to acquire computer memory from a UNIX-based system, it is preferred to acquire memory from */dev/mem* prior to endeavouring from */dev/kmem*. Specifically, memory device /*dev*/*kmem* should only be attempted when memory acquisition from */dev/mem* fails or if it is not available on the current system. Moreover, it is important that investigators understand that Linux 2.6.x kernels by default do not support /*dev*/*kmem*. As such, kernel memory interface */proc/kcore* can be attempted if acquisition from /*dev*/*mem* fails.

It is important to understand that Linux pseudo-file memory interface */proc/kcore*, while similar, is not the same as memory device */dev/kmem*. Specifically, the latter provides an interface to physical memory in ELF format while the former provides direct access to all physical memory without associated I/O devices and their memory.

1.3 Memory acquisition software background

1.3.1 DD

The *dd* program is one of the most prevalent UNIX utilities still in use. It is used, among other things, to convert data between ASCII and EBCDIC and perform byte-order conversions (converting data between little and big endian). However, in the realm of digital forensics the *dd* program is most commonly used for disk cloning and preparation. *Dd* is adept because it can read from standard input and write to standard output, enabling it to be readily scripted. The *dd* program is found on every UNIX clone including all BSD, Linux and Solaris systems, to name only a few. As such, it is universal to UNIX.

Despite *dd*'s many uses, it has certain drawbacks. Its most important limitation for forensic work is that it does not perform disk image hashing nor is it capable of recovering data from significantly damaged hard disk drives. For these reasons, other *dd*-based programs have been written over the years to correct for these limitations. These programs include Kurt Garloff's *dd_rescue* and the GNU equivalent *ddrescue* written by Antonio Diaz Diaz. Both programs are well suited for recovering data from ailing disk drives. They are not, however, well suited for memory acquisition.

Two *dd*-like programs well suited for memory acquisition include *dcfldd* written by Nick Harbour and *dc3dd* written by Jesse Kornblum. Both programs provide additional functionality over *dd* including on-the-fly hashing, current operation progress, error log reporting and split output files.

However, *dcfldd* supports additional hashing algorithms and can simultaneously synchronize data writing to multiple output devices.

Although *dc3dd* and *dcfldd* are superior choices over *dd* for memory acquisition, their use was considered inappropriate for this work. This is because ensuring the cross-compilation of these two tools across three BSD operating systems, three Linux operating systems and Solaris and ensuring that all library dependencies are satisfied is well beyond the scope of this document.

The *dd* command used for memory acquisition under all BSD, Linux, and Solaris operating systems examined in this work were of the form:

```
$   dd if=/dev/mem of=mem.dd bs=512 conv=noerror
```

Where /*dev*/*mem* could be any memory-related device or pseudo-file (see sections 1.2.15, 1.2.2.5, 1.2.3.3 and 1.2.4 for more details). The above command causes *dd* to read from the specified input file until there is no more left to read, which in some cases results in very small dumps files while in other cases caused *dd* to go well beyond the system's memory space looping back to the beginning and dumping the same memory over again. Only the root user can acquire memory-related devices in this manner using *dd.*

The reader may question why specific memory page-file[16] sizes or cut-off limits[17] were not specified for the above *dd* command. The author reflected on this but decided in the end that it would be best not to specify any hard upper limits to in order to determine if a given operating system had specific built-in upper memory access limits or whether memory acquisition would simply loop over once the end of memory was reached.

Acquisition-specific memory device details can be found in Section 2.1. All experimental results are found in Annex C.1 while the analyses of these results are found in Section 2.2.1.

1.3.2 Memdump

Written by Wietse Venema as a part of The Coroner's Toolkit, the *Memdump* program dumps memory in a similar fashion to *dd.* It reads system memory according to the default memory page size and writes output to standard output which requires filename piping in order to save the output to a file (and to avoid crashing the command shell).

According to the program's source code, by default, it will read memory from system device */dev/mem*. However, if the appropriate command line parameter is used it will instead read from */dev/kmem*. Nonetheless, as examined in sections 1.2.15, 1.2.2.5, 1.2.3.3 and 1.2.4, reading from device */dev/kmem* is neither suggested nor is it always available,. Instead, */dev/mem* is the preferred method of reading memory on all the aforementioned operating systems examined in this work and as such, will be the default memory device used for the experimentation carried out herein.

---

16 Typical memory page-file sizes are 4 KiB for the x86 architecture.

17 If the system uses 4 KiB page-file size and there is 4 GiB RAM then a count of 1,048,576 can be appended to the above *dd* command to ensure that the program will stop reading beyond this point.

Memdump is more sophisticated than *dd* as it better handles errors. The program is downloaded as source code (C code) and is compiled using the provided *Makefile*. It compiles without difficulty on Linux. Under Solaris, the reader may be required to modify the *makedefs* file and substitute the command *make* with *gmake*, depending on which software compilation suite is in use at the reader's facility. Under BSD, appropriate C *case*-based entries have to be added to the *Makefile* to support newer or additional BSD systems, although this is a straightforward procedure.

Since the *Memdump* program must be compiled for each platform it is to be run against, it is advised that the investigator have a precompiled version of *Memdump* for each of the various UNIX-like systems he may encounter[18]. Fortunately, since the program has few dependencies and is multi-platform it will readily compile using both *GCC* and *GNU Make* or Sun's C/C++ development software suite. If necessary, the *Memdump* program can be introduced onto a suspect system using CD or other read-only media as a statically compiled binary. Of course, compiling the program atop a target system will invariably change its state; however, by exactly how much is not known.

It is important to emphasize that 64-bit versions of *Memdump* will not work on 32-bit systems. Although 32-bit versions will work on 64-bit systems, they will be limited to a maximum memory dump size of 4 GiB RAM. Moreover, only the root user can use *Memdump* to acquire system memory. The *Memdump* tool as run under all BSD, Linux, and Solaris operating systems examined in this work were of the form:

```
$  memdump > memory.img
```

Acquisition-specific memory device details can be found in Section 2.1. All experimental results are found in Annex C.2 while the analysis of these results is found in Section 2.2.2.

1.3.3 Helix 3 Pro Release 3

Originally available as Helix, it was a free Linux live-based bootable CD. Today, only Helix 3 Pro is available. It is no longer freely available and must be purchased. Developed and marketed by E-fense Inc. it is a popular live response CD among digital forensic practitioners. The free version was apparently ended due to the lack of money E-fense generated from on demand support-based fees. The last free version of Helix was Helix 2009 Release 1 and was used as the basis for its commercial release of Helix 3 Pro. The first free version of Helix dates back to 2003, while the final free version was released in 2009. The first commercial release of Helix, renamed Helix 3 Pro, was released as Helix 3 Pro Release 1. Today, Helix 3 Pro is currently at Release 3.

Both Helix and Helix 3 Pro are largely based on Ubuntu 32-bit Linux. Helix and Helix 3 Pro are both a live response and bootable 32-bit Linux CD-based operating systems. They are used primarily for live computer system acquisition including the acquisition of disks and filesystems (including mounted encrypted filesystems) and computer memory. However, it can be booted via CD for post-mortem media acquisition and analysis and it supports a great many filesystems, far more than most standard Linux systems.

---

18 Compile the program statically to link in all required library functions – this will of course increase the size of the executable and the amount of memory it will consume.

Its live response capabilities make Helix 3 Pro ideal for incident handlers and forensic investigators arriving at the scene of an incident. Under Windows, the Helix 3 Pro system not only provides the ability to acquire disk and memory, but also facilitates the capture of various Windows-based volatile data and system information. Helix 3 Pro also offers limited Mac-based capabilities. Moreover, it the only tool examined herein that successfully hashes all memory-specific dump-files. Additionally, it provides the ability to hash acquisition files using four different possible hash algorithms including[19] MD5, SHA-1, SHA-256 and SHA-512. Under Linux, when used for live response, Helix 3 Pro can be used to acquire both system memory and locally attached storage devices.

Although Helix 3 Pro is based on its free predecessor (Helix), they are not the same as there are important differences between them. The commercial version takes advantage of a proprietary acquisition software program called *helix3pro* that can be used to acquire hard disk drives, memory and volatile data (Windows only) against live systems. Moreover, using the Helix 3 Pro *receiver* program it is possible to send all acquisitions to a dedicated acquisition-based system over the network. Of course, acquisitions can be saved to locally attached storage too.

While using Helix 3 Pro, it was not apparent which system device was used for memory acquisition. However, after running the *helix3pro* acquisition program using the Linux *strace* command it was found that memory acquisition was carried out using the */dev/mem* system device through the *open()* system call. Because Helix 3 Pro is 32-bit and uses a 32-bit *open()* system call, memory acquisition may potentially fail beyond the first 4 GiB memory. Experimental result specifics are found in Annex C.3 while the analysis of these results is found in Section 2.2.3. Memory device specifics can be found in Section 2.1.

Under kernel 2.6.x Linux, the *helix3pro* program (the CD must be appropriately mounted) was run as follows:

```
$ cd /media/cdrom/Linux

$ ./helix3pro
```

Unlike the other tools examined herein which acquire memory through */dev/mem* with no detected library dependency issues, Helix 3 Pro encountered dependency issues depending on the underlying Linux operating system. Primarily, because Helix 3 Pro runs atop GTK it absolutely requires that these libraries be installed. The problem with using Helix 3 Pro against 64-bit Linux systems is that although these libraries are already typically installed they are generally 64-bit in nature while those required by Helix 3 Pro must be 32-bit, including required GTK dependencies.

Moreover, by default, most 64-bit Linux systems will not run 32-bit programs out of the box. Thus, depending on the underlying operating system, specific libraries must be installed in order to rectify this. Although this may not be an issue for customized 64-bit Linux systems, it was found to be a problem with the newly installed 64-bit versions of Fedora Core 15 and Ubuntu 11.04 virtual machines. Library dependency specifics can be found Annex D.4.

19 X-Ways also offers file hashing capabilities but this feature was found not to work – see sections 1.3.6 and 2.2.6 for more details

Further complicating matters with Helix 3 Pro is the size of the executable with respect to the amount of main memory required to run it. The executable is just over 30 MiB in size and when loaded into main memory with required libraries can consume several hundred MiB of memory. Thus, using Helix 3 Pro for memory acquisition can also pose a problem as it may potentially overwrite or force evidence residing in memory to be swapped out, more so than any other tool examined in this work.

Tests were made against Red Hat 9 and while nothing specifically indicated that acquisition would not function correctly, too many library dependencies had to be satisfied. However, since no Red Hat 9 software repositories were publicly available, these dependencies could not be adequately satisfied in a timely manner in order to determine definitively if acquisition would have succeeded[20]. Moreover, it was not possible to test Helix 3 Pro against Solaris or BSD since no support is afforded to these operating systems.

1.3.4 Fmem

Developed as part of a master's thesis, *Fmem* is a part of larger software structure, *Foriana*, which is a Linux-based memory analysis framework. The thesis was written by Czech student Ivor Kollár at the Department of Software Engineering while attending the Faculty of Mathematics and Physics of Charles University in Prague in 2010. This is a Linux-specific tool and it must be run as root in order to acquire system memory. Furthermore, according to the documentation provided by his thesis [33] it affords no support for 2.4.x kernels, only 2.6.x generation kernels running atop 32 and 64-bit PC-based computer systems.

However, tests conducted against Red Hat Linux's 2.4.x kernel by the author while attempting to compile the *Fmem* tool resulted in an error message indicating that although the kernel did not currently support loadable kernel modules in its present form. The message was too inconclusive to determine whether compilation would have succeeded against Red Hat 9. While some 2.4.x kernels[21] supports LKMs, recompiling a given kernel in order to verify this is outside the scope of this work and as such, was not carried out. Thus, while the potential for *Fmem* support exists for 2.4.x kernels, this capability will not be determined herein. However, the reader is invited to verify if this is in fact the case.

Since memory analysis is not directly examined in this work, only the *Fmem* portion of *Foriana* is examined. The latest version of *Fmem* is 1.6-1 and was released August 2011. The very first publicly available version of *Fmem* was 0.5.0.

The objective behind this tool is to provide another means of directly interfacing with Linux memory device */dev/mem*, which ordinarily limits access to approximately the first 1 MiB memory, even to the root user. Often, memory beyond this limit is not accessible for unsophisticated tools.

---

20 Older source code files could possibly have been found and recompiled but this would have been a highly time-consuming endeavour. Instead, the author leaves it to the community to determine if memory acquisition using *Fmem* would have been possible.

21 Red Hat 9.0 Linux's kernel can be recompiled with LKM support. However, whether it is supported by *Fmem* is currently unknown.

The tool is compiled as a Linux LKM that is then loaded into kernel space using the *insmod* command. The loading process is automated for the investigator by running the accompanying *run.sh* command. Although the kernel module requires very little in terms of dependencies it is entirely dependent on the current running kernel version. Thus, it cannot be readily used in precompiled form across different Linux kernels. Instead, in order to use the tool, the suspect system must have a functional C compiler and required C library files, both of which are standard on most Linux default installations. However, compiling the tool will invariably change the system's running state. Once it has been ascertained by the investigator that the prerequisites have been satisfied the tool can be compiled on the target system.

Once the LKM is loaded, the module then creates a new memory interface device, */dev/fmem*, which is then read using standard UNIX tools such as *dd*. The LKM requires that a hard memory limit be specified (using the *dd count* parameter) in order to prevent a memory dump from starting over again once the end of physical memory has been reached. The information source used for specifying the upper limit is the *MemTotal* value found in kernel pseudo-file */proc/meminfo*.

The tool was run as follows under all versions of kernel 2.6.x Linux tested herein:

```
$   cd fmem_1.6-1

$   make

$   ./run.sh

$   dd if=/dev/fmem of=memory.img bs=1K count=`cat /proc/meminfo | grep
        MemTotal | awk '{print $2}'`
```

In order for the *Fmem* LKM to compile, it is necessary for the target system to have the kernel header-based source code files for the currently running kernel. Installing it, if it is not already done, requires Internet access (or access to a local network repository), all of which will invariably change the system's state. Compiling the LKM will also change the system's state but to a lesser extent than having to download and install the necessary kernel header-based source code files. Running the command "*./run.sh*" loads[22] the LKM and provides some basic memory address ranges[23], none of which can actually be used. A *Makefile* is provided by *Fmem* for compiling the module.

The author considers this tool to have enormous potential as it was found to be functional on all 2.6.x kernels experimented upon in this work.

Acquisition-specific memory device details are found in Section 2.1. All experimental results are found in Annex C.4 while the analysis of these results is found in Section 2.2.4.

---

22 The LKM can be manually loaded using the *insmod* command.

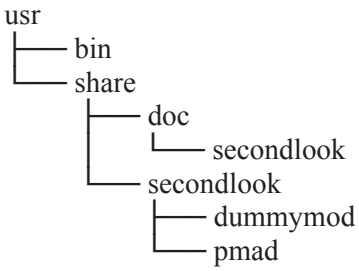
23 The script relies on kernel pseudo-file */proc/mtrr* that is only useful for providing information about Memory Type Range Registers (MTRR). MTRR is a new processor-based feature that provides an improved mechanism for partitioning and managing the system's memory resources. Its use as a substitute for information derived from */proc/iomem* or */proc/meminfo* is not suggested. [38]

1.3.5 Second Look

Developed by PikeWorks Corp., the Second Look toolset is a combination of a memory acquisition tool (consisting of a memory access driver and acquisition script) and memory analysis programs (CLI and GUI). Currently at release version 2011.05, it is provided to the customer as a Debian software package. Although it is packaged using the standard Debian format and was designed to run atop Ubuntu and other Debian-based Linux distributions, it can be made to work atop other distributions including Fedora Core.

The primary component of the toolset is its memory analysis programs[24] that require a Debian-based system for easy installation. Software installation is done in two ways: installing the software package on a supported Debian-based platform or extracting the data files from the Debian archive. The latter installation method was performed herein.

The Debian package is extracted on non-Debian systems using the "*ar -x*" command which results in two main[25] files, *control.tar.gz* and *data.tar.gz*. Only the latter file is important; its contents are extracted using the *gzip* and *tar* commands, successively. Extracting the data files from this archive creates a new subdirectory structure that is as follows:



Within directory structure *./usr/share/secondlook/pmad* resides a simple yet effective memory access driver that must be compiled as an LKM. A *Makefile* is provided therein for compiling the module. This module, as with *Fmem*, requires that the kernel header-based source code files for the running kernel be installed, as without them compiling the module is not possible. Of course, compiling the module invariably changes the system's state but having to download and install the header-based source code files and then compiling the module will change the system even more. Once compiled the module must be loaded into kernel space. The loading of the module and acquisition of memory were carried out as follows against all Linux 2.6.x kernel-based installations in this work:

```
$       cd usr/share/secondlook/pmad

$       make

$       insmod pmad.ko

$       cd ../../../bin
```

24 There is a CLI and GUI-based programs.

25 A third file of no significance, *Debian-control*, is also extracted from the archive and can be safely deleted.

```
$       ./secondlook-memdump.sh memory.img /dev/pmad
```

The author considers this tool to have enormous potential as it was found to be functional against all the 2.6.x kernels experimented upon in this work. However, it was not possible to get the *pmad* LKM to work under Red Hat 9; as such, it is unlikely to work against 2.4.x kernels in so long as LKM module support is disabled by the kernel.

The provided LKM *pmad* is a customized LKM based in part on Dave Anderson's Linux Crash Driver. The LKM itself appears to be re-distributable according to the GNU GPL license under which it is licensed although this interpretation should not be construed as legal advice. The *pmad* LKM is very similar in functionality to the *Fmem* LKM and appears just as effective.

Second Look appears to be marginally more effective than Fmem. This is because the Second Look acquisition script, *secondlook-memdump.sh*, relies exclusively on kernel pseudo-file */proc/iomem* for information as to which memory addresses to dump. See Annex D.3 for an example concerning the use of */proc/iomem*. The script reads the appropriate memory addresses from */proc/iomem* and then instructs the *dd* program to dump memory from these addresses using the newly created */dev/pmad* memory device made available when the *pmad* LKM was loaded into kernel space.

The */proc/iomem* kernel structure provides a complete and detailed list of all I/O hardware memory found throughout the system. This includes both physical RAM and other hardware memory that is at least partially accessible to the operating system (see Annex D.3 for a detailed example). However, a Second Look memory dump-file is always[26] bigger than the physical amount of memory found in the computer system because some hardware device-specific memory is also dumped along with physical RAM.

Hence, through experimentation it was found that not only was the full amount of operating system-detected RAM dumped but in all cases, at least some hardware-reserved RAM (RAM set aside for use by the computer's hardware) was also captured. In other cases, a given system's entire hardware-reserved RAM was captured in its entirety. Moreover, in cases where Second Look successfully captured all hardware-reserved memory it also succeeded in capturing, at least in part, computer hardware-specific memory (non-RAM computer memory found in peripheral hardware and integrated components) from sources including ACPI memory, video card memory and network device buffers, etc. In this manner, a Second Look memory dump is very complete and may include additional sources of information. [40]

This program may eventually set the standard by which Linux-based memory dumps are carried out. Moreover, it appears to work against all 32 and 64-bit 2.6.x kernels making it a potentially indispensable memory acquisition tool.

Acquisition-specific memory device details can be found in Section 2.1. All experimental results are found in Annex C.5 while the analysis of these results is found in Section 2.2.5.

26 Based on conversations with PikeWorks [40] and experimentation carried out in Annex C.5.

1.3.6 X-Ways Capture

X-Ways Software Technology AG of Germany, developer of the famous and versatile WinHex hex-editing program, developed a commercial memory acquisition-based software tool named X-Ways Capture. The tool is designed for investigators requiring the acquisition of both locally attached storage and computer system memory against both Windows and Linux systems. However, since the objective of this work is to evaluate memory acquisition against UNIX-based systems only the Linux acquisition module was tested. Moreover, the tool does not support any BSD or Solaris.

While the utility includes documentation, it is of relatively poor quality and answers few of any questions which the reader may have [32]. The X-Ways Capture tool, at least under Linux, is little else than a fancier version of *dd* and produced results no better than the latter tool.

According to tool traces using the *strace* command, X-Ways Capture works by reading memory directly from */dev/mem* using the *open()* system call. It also queries for and reports about, if so configured, currently mounted partitions and running processes. The information source for the total amount of memory it can acquire is made available to the program by Linux kernel pseudo-file structure */proc/meminfo*[27]. The program can be configured, through its configuration file *capture.ini*, to hash acquired memory and disk images, list currently loaded kernel modules and search for encrypted filesystems using the system loop command (*losetup*). The program works against both 2.4.x and 2.6.x generation kernels and could therefore be tested against Red Hat Linux. However, the tool must be run as root in order to carry out successful storage and memory acquisition.

In tests performed herein, its memory acquisition capabilities are insufficient for a commercial software program. Moreover, even though the tool was configured to hash all acquisition dump-files, no dump-file was ever successfully hashed throughout all of the Linux memory acquisition experiments conducted herein using this tool. Furthermore, process, mount and loaded kernel listings were acquired directly from the system using the *ps* command and kernel structure pseudo-files */proc/partitions*, */proc/mounts* and */proc/modules*, respectively. The program, which could only be tested against Linux, was run as follows:

```
$       ./xwcapture
```

In short, the tool is a disappointment. However, the tool's one redeeming feature is that under all circumstances it will refuse the storing of dump-files, whether derived from storage or memory, to the root filesystem. Instead, dumps must be sent to another disk partition, preferably an empty one, a mounted USB or FireWire mass storage device, or over the network using NFS.

A listing of the configuration file, *capture.ini*, can be found in Annex D.1. Specifics concerning the memory device used for acquisition are found in Section 2.1. All experimental results are found in Annex C.6 while the analysis of these results is found in Section 2.2.6.

27 Pseudo-file */proc/meminfo* reports total system memory using the *MemTotal* value that reports the total amount of operating system addressable physical RAM.

2 Device acquisition summary and experimental results analysis

2.1 Tool device acquisition summary

This section provides a bird's eye view of the specific tools experimented upon in this work with respect to the operating systems they were tested against while taking into account the memory devices used for memory acquisition.

Specifically, the following synopsis is a breakdown of the results obtained in Annex C. The tools are listed in the order in which they were experimented upon as found in Annex C and are listed in the corresponding tables below:

Table 1. Dd versus operating system-specific acquisition memory device.

Operating system	**Memory device**
FreeBSD (32 and 64-bit)	/dev/kmem /dev/mem
NetBSD (32 and 64-bit)	/dev/kmem /dev/mem
OpenBSD (32 and 64-bit)	/dev/kmem /dev/mem
Red Hat 9 Linux	/dev/kmem /dev/mem /proc/kcore
Solaris	/dev/mem /dev/kmem /dev/allkmem
Ubuntu (32-bit, 32-bit PAE and 64-bit)	/dev/mem /proc/kcore
Fedora Core 15 (32-bit, 32-bit PAE and 64-bit)	/dev/mem /proc/kcore

Table 2. Memdump versus operating system-specific acquisition memory device.

Operating system	**Memory device**
FreeBSD (32 and 64-bit)	/dev/mem
NetBSD (32 and 64-bit)	/dev/mem
OpenBSD (32 and 64-bit)	/dev/mem

Red Hat 9 Linux	/dev/mem
Solaris	/dev/mem
Ubuntu (32-bit, 32-bit PAE and 64-bit)	/dev/mem
Fedora Core 15 (32-bit, 32-bit PAE and 64-bit)	/dev/mem

Table 3. Helix 3 Pro Release 3 versus operating system-specific acquisition memory device.

Operating system	**Memory device**
Ubuntu (32-bit, 32-bit PAE and 64-bit)	/dev/mem
Fedora Core 15 (32-bit, 32-bit PAE and 64-bit)	/dev/mem

Table 4. Fmem versus operating system-specific acquisition memory device.

Operating system	**Memory device**
Ubuntu (32-bit, 32-bit PAE and 64-bit)	/dev/fmem
Fedora Core 15 (32-bit, 32-bit PAE and 64-bit)	/dev/fmem

Table 5. Second Look versus operating system-specific acquisition memory device.

Operating system	**Memory device**
Ubuntu (32-bit, 32-bit PAE and 64-bit)	/dev/pmad
Fedora Core 15 (32-bit, 32-bit PAE and 64-bit)	/dev/pmad

Table 6. X-Ways Capture versus operating system-specific acquisition memory device.

Operating system	**Memory device**
Ubuntu (32-bit, 32-bit PAE and 64-bit)	/dev/mem
Fedora Core 15 (32-bit, 32-bit PAE and 64-bit)	/dev/mem

2.2 Results analysis

This section examines in detail the experimental results obtained in Annex C and where possible, attempts to provide meaningful insight into the various results obtained therein. Moreover, the following analysis provides a brief commentary and recommendation for each tool.

2.2.1 Dd

2.2.1.1 Analysis

2.2.1.1.1 BSD analysis

Detailed analyses of experimental results regarding memory acquisition using the UNIX *dd* command have proven interesting and of particular benefit to the various BSD operating systems. The tool succeeded in acquiring the memory of every BSD operating system examined in this work including their 32 and 64-bit counterparts.

Although none of the 32-bit BSD systems examined in this work supported PAE, the success of the various memory acquisitions were met with certain caveats. Specifically, for the FreeBSD and NetBSD 32-bit operating systems, memory acquisition worked up to 4 GiB RAM at which time all additional data in the memory dump-files were repetitions of the first 4 GiB. OpenBSD 32-bit did not suffer from any memory dump-file repetition and appeared to have successfully stopped memory acquisition when the memory dump-file reached 4 GiB in size. Thus, memory acquisition for FreeBSD, NetBSD and OpenBSD (32-bit, respectively) succeeded up to the supported maximum memory size for each operating system 32-bit non-PAE, 4 GiB RAM.

However, analysis of the FreeBSD and NetBSD 64-bit operating systems proved more challenging. For these specific operating systems, memory up to each system's memory detection limit was successfully dumped. Moreover, data written to these dump-files beyond the operating system memory detection limit appeared to contain both hardware-reserved and hardware-specific computer memory. In contrast to its 64-bit cousins, the OpenBSD 64-bit memory acquisition was no different from its 32-bit non-PAE counterparts since they too can only support a maximum of 4 GiB RAM (using its default 64-bit kernel).

For all BSD systems, acquisition succeeded only from memory device */dev/mem*.

2.2.1.1.2 Linux analysis

Analysis for the Linux *dd*-based memory acquisitions was less straightforward. Acquisition partially succeeded for Red Hat Linux 9 which was able to dump approximately 906 and 896 MiB RAM from kernel structure */proc/kcore* and memory device */dev/mem* respectively, at which time both acquisitions ceased for unknown reasons. They likely stopped because the system was no longer able to furnish additional RAM for acquisition. However, the memory that was captured was found to be valid and intact.

Under modern 2.6.x kernels (e.g. Fedora Core 15 and Ubuntu 11.04 (32 and 64-bit, respectively)), acquisition was somewhat more convoluted. For each of these systems, when acquiring memory from memory device */dev/mem*, acquisition was found to be altogether disappointing due to the inability to acquire memory beyond the first one MiB memory.

However, when acquiring memory from kernel structure */proc/kcore*, differences become apparent not only between 32-bit and 64-bit operating systems but even between distributions. Consider that for Ubuntu 32-bit and 32-bit PAE, a partial capture of memory succeeded, up to slightly less than the first 1 GiB RAM for each of the aforementioned systems. Conversely, for Fedora Core 32-bit PAE, Fedora Core 64-bit and Ubuntu 64-bit, the dump files filled up (or nearly filled up, as was the case for Fedora Core 64-bit) the various disk partitions. However, these dumps contained no useful data whatsoever beyond the first MiB.

2.2.1.1.3 Solaris analysis

Memory acquisition using *dd* under Solaris 32-bit PAE and Solaris 64-bit were disappointing and resulted in memory acquisition less than one MiB in size from memory devices */dev/mem*. Acquisition from memory devices */dev/kmem* and */dev/allkmem* proved futile in that they each resulted in zero bytes of data acquired.

2.2.1.2 Recommendations

Dd does not generate memory dump-file hashes or automatically fill up and complete Chain of Custody forms. Thus, if this is a requirement, *dd* should not be used, at least where Linux is concerned; instead, Helix 3 Pro should be used.

The *dd* tool is only capable of acquiring BSD-based memory from memory device */dev/mem* only. For 32-bit non-PAE BSD acquisitions, the investigator should only extract and use the first 4 GiB from the memory dump-file regardless of how large it is. For 64-bit acquisitions, the investigator should extract memory from the dump-file only up to the specified memory detection limit of the underlying operating system. However, *Memdump* is a more reliable memory acquisition tool than *dd* where BSD operating systems are concerned.

Where Linux is concerned, *dd* appears to acquire partial memory dumps from memory devices */dev/mem* and */proc/kcore* while running under 2.4.x kernels. While running under 2.6.x kernels, *dd* is more incoherent. Specifically, memory acquisition from memory device */dev/mem* should altogether be left unused while in use under 2.6.x. Kernels. However, partial success may be possible if memory acquisition is carried out against kernel structure */proc/kcore*. Nonetheless, success will be largely based on architecture and distribution. Specifically, 64-bit Linux distributions appear incapable of acquiring any meaningful amount of data from */proc/kcore*, regardless of distribution. Where 32-bit 2.6.x distributions are concerned, the issue is muddled. Experimentation reveals that specific 32-bit distributions are capable of acquiring partial memory dumps, regardless of PAE use while others altogether fail. Thus, if *dd* is the only tool available for memory acquisition under Linux, it should not be used under 64-bit 2.6.x-based kernels but can be used for partial memory acquisition of 32-bit non-PAE/PAE 2.4.x and 2.6.x kernels. However, overall, *Memdump* appears to be more capable than *dd* for Linux-based memory acquisitions.

As for Solaris, *dd* is altogether incapable of acquiring memory from any of its memory-based devices. Thus, where Solaris is concerned, the *Memdump* memory acquisition tool is to be used in lieu.

2.2.2 Memdump analysis

2.2.2.1 Analysis

Under *Memdump*, the default memory-based acquisition device used was */dev/mem* that was used for all experimentation using *Memdump*.

2.2.2.1.1 BSD analysis

Detailed analyses of experimental results regarding memory acquisition using the *Memdump* tool have proven interesting and of particular benefit to the various BSD operating systems. The tool succeeded in acquiring the memory of every BSD operating system examined in this work including their 32 and 64-bit counterparts.

Although none of the 32-bit BSD systems examined in this work supported PAE, the *Memdump* tool successfully captured, for each of the 32-bit BSD-based operating systems, exactly 4 GiB RAM, up to each system's respective 32-bit memory limit.

Furthermore, analysis of the FreeBSD and NetBSD 64-bit operating systems demonstrated that *Memdump* was able to acquire each operating system's memory in its entirety. For these 64-bit operating systems, memory up to each system's memory detection limit was successfully dumped. However, the final memory dump-files were larger than the amount of allocated system memory for each the aforementioned 64-bit virtual machines. Specifically, these dump-files appeared to contain both hardware-reserved and hardware-specific computer memory that was located several hundred MiB beyond the 8 GiB memory limit of allocated system memory. Immediately beyond this hardware-specific memory resident data, the remainder of the dump-files were found to be empty but which managed to fill up both 64-bit BSD's system disk partitions. Memory acquisition ceased when the disk partitions filled up.

In contrast to its 64-bit cousins, the OpenBSD 64-bit memory acquisition was no different from its 32-bit non-PAE counterpart since it can only support a maximum of four GiB RAM, with its default 64-bit kernel.

2.2.2.1.2 Linux analysis

Analysis for the Linux *Memdump*-based memory acquisitions was not straightforward. Acquisition partially succeeded for Red Hat Linux 9 in that it tool was able to dump approximately 896 MiB RAM from */dev/mem*, at which time the acquisition ceased for unknown reasons. It likely stopped because the system was no longer able to furnish additional RAM for acquisition. However, the memory that was captured was found to be valid and intact.

Under modern 2.6.x kernels using memory device /*dev*/*mem* (e.g. Fedora Core 15 and Ubuntu 11.04 (32 and 64-bit, respectively)), acquisition was somewhat convoluted. Where 32-bit and 32-bit PAE versions of these operating systems were used, it was found that approximately only the first 1 MiB RAM could be acquired. Where 64-bit versions of these operating systems were used it was found that memory up to approximately 513 MiB RAM could be acquired. It is likely that memory acquisition for these 64-bit operating systems stopped because the system was no longer

able to furnish additional RAM for acquisition. Analysis of the memory dumps from the 32-bit and 32-bit PAE (one MiB dump-file sizes) operating systems reveals that BIOS-based low-memory had been acquired. As for the 64-bit systems (513 MiB dump-file sizes), the memory dumps appeared valid and intact.

However, acquisition against */proc/kcore* was altogether something else. Acquisition against Red Hat 9 succeeded in a partial memory dump approximately 907 MiB in size. Ubuntu 32-bit and 32-bit PAE memory dumps against this pseudo-file memory device succeeded in acquiring dump-files approximately 1 GiB in size that appeared to contain valid memory-based data. However, where Ubuntu 64-bit and Fedora Core 32-bit PAE and 64-bit were concerned, this was altogether another matter. Specifically, these memory dumps carried out against */proc/kcore* resulted in dump-files which went well beyond the amount of allocated virtual machine but which in the end were all found to contain no data other than low-level BIOS data contained within the first MiB therein.

2.2.2.1.3 Solaris analysis

Memory acquisition using *Memdump* under Solaris 32-bit PAE and Solaris 64-bit were successful. The memory dump under Solaris 32-bit PAE resulted in a dump-file 3,583.6 MiB in size while the memory dump conducted under Solaris 64-bit resulted in a dump-file 8191.6 MiB in size. Both dumps are consistent with the information presented in Section 1.2.3.3.

It is not currently understood why the Solaris 32-bit PAE memory dump ended where it did. It is likely that it ended there because at that point, between memory addresses 3,583.6 MiB to 4,096 MiB, hardware-reserved computer memory was occupying that location which Solaris does not make available to non-kernel related subsystems. Memory above 4,096 MiB to 8,192 should have consisted entirely of non-hardware-reserved computer memory as a 32-bit operating system addresses hardware using 32-bit computer addressing, which must be located below the 4 GiB 32-bit memory limit. On the other hand, Solaris 64-bit was entirely successful in dumping its computer memory, with its memory dump-file less than 512 KiB smaller than said virtual machine's allocated memory.

2.2.2.2 Recommendations

Memdump does not generate memory dump-file hashes or automatically fill up and complete Chain of Custody forms. Thus, if this is a requirement, *Memdump* should not be used, at least where Linux is concerned; instead, Helix 3 Pro should be used.

The *Memdump* tool is the preferred method for acquiring 32-bit BSD-based operating system memory against memory device */dev/mem*. For 64-bit acquisitions, the investigator should extract memory from a given 64-bit memory dump-file only up to the specified memory detection limit of the underlying operating system in order to ensure that only valid and intact operating system memory is recovered.

Where Linux is concerned, *Memdump* is the recommended memory acquisition tool to use when only it and *dd* are available. However, if superior tools (e.g. *Fmem*, Second Look) are available then they should be used in lieu. Linux-based 64-bit memory dumps made with *Memdump*

should contain valid and intact data whereas those acquired through *dd* will likely be empty. Moreover, since *Memdump* works well under Linux using its default memory acquisition device /*dev*/*mem*, at least in comparison to *dd*, the investigator does not have to deal with the issue of selecting between */dev/mem* and */dev/kmem* (which does not generally exist under 2.6.x Linux kernels). Unfortunately, *Memdump* is no better than *dd* at acquiring memory under 32-bit and 32-bit PAE 2.6.x-based Linux. Under Red Hat 9, it *fared* about as well as *dd*.

As for Solaris, *Memdump* is the only tool examined herein which is at all capable of acquiring memory from this operating system. The tool is fully capable under Solaris 64-bit but under Solaris 32-bit stops memory acquisition when the hardware-reserved computer memory address range is encountered. This is likely caused by kernel-specific mechanisms that prevents direct user access beyond the aforementioned hardware-reserved computer memory.

2.2.3 Helix 3 Pro Release 3 analysis

2.2.3.1 Linux analysis

Helix 3 Pro Release 3 has the potential for greatness but falls short. It is, however, an efficiently simple and elegant tool to use both for disk and memory acquisition. Featuring a GUI-based acquisition program, it also provides the ability to implement remote network acquisition storage using another Helix-driven system. Although Helix 3 Pro works against both Windows and Mac OS X (with limited functionality), it is not compatible with BSD, Solaris or Linux 2.4.x and earlier operating systems.

Unfortunately, during experimentation it was found that 64-bit operating support was altogether lacking. Since the Helix 3 Pro acquisition program is compiled as a 32-bit application, it requires that 64-bit systems already have the necessary 32-bit GTK libraries installed. Otherwise, they will have to be installed prior to acquisition. Furthermore, most 64-bit Linux systems, by default, do not support 32-bit applications, thereby requiring that 32-bit application compatibility libraries be installed prior to acquisition. Installing these requirements just prior to acquisition will undoubtedly change the system's running significantly. In order to remedy this, E-fense should consider providing a 64-bit compiled version of their acquisition software or statically compiling their 32-bit programs.

However, even once these dependencies are resolved, whether just prior to acquisition or already having been installed for some time, 64-bit memory acquisition cannot acquire more than the first 4 GiB RAM due to the 32-bit nature of the acquisition program and the type of system call used to access the system's memory device. Thus, for 64-bit systems, any memory above this limit will be entirely inaccessible to the acquisition program.

Fortunately, 32-bit Linux operating systems fare better. In so long as the required underlying GTK libraries are installed (most distributions provide and install these libraries by default), the system's memory can be acquired, up to the detected memory limit of the operating system. Moreover, Helix 3 Pro fully supports 32-bit PAE. In tests, it was able to acquire allocated memory up to the operating system's memory detection limit.

Unfortunately, while loading Helix 3 Pro, it can consume upwards of 300 MiB RAM as it has many dependencies that must be loaded into memory prior to acquisition. However, this will have the effect of pushing potentially important evidence already in RAM to the system page-file where it will no longer be acquirable through memory acquisition.

The one truly redeeming feature of Helix 3 Pro is that it is only the tool examined in this work which hashes memory dump-files and can automatically generate Chain of Custody forms. It currently supports the MD5, SHA-1, SHA-256 and SHA-512 hashing algorithms.

2.2.3.2 Recommendations

This tool will only work for 32-bit and 32-bit PAE 2.6.x generation of Linux kernels. It will not work against 64-bit Linux systems beyond the acquisition of the first 4 GiB RAM. As such, Helix 3 Pro Release 3 should only be used against Linux 2.6.x 32-bit-based operating systems.

The three redeeming qualities that this tool provides over its more suggested counterparts (e.g. Second Look and *Fmem*) are that its GUI is easy to navigate and use. Secondly, another Helix-driven system on a networked computer can be used to store acquisitions carried out against one or more suspect systems. Thirdly, it can be configured to automatically perform memory dump-file hashing using a variety of industry-accepted algorithms and generate Chain of Custody forms.

This tool should only be used if *Fmem* or Second Look are unavailable, either because they are not a part of the investigator's toolkit or because either software tool's memory acquisition driver cannot be compiled due to one or more missing dependencies or prerequisites.

2.2.4 Fmem analysis

2.2.4.1 Linux analysis

Analysis reveals *Fmem* to be a highly effective and capable memory acquisition tool. Furthermore, it worked as expected in every experiment. Although it was not specifically designed to run under 2.4.x kernels, it may nevertheless be possible. However, for this to be attempted, the underlying 2.4.x kernel must afford LKM module support, a feature offered by some but not all 2.4.x kernels (including Red Hat 9 which by default has disabled LKM support).

The only downside to this tool is that it requires the investigator to compile the *Fmem* memory acquisition driver prior to dumping memory. This requires the suspect system to already have a suitable C compiler and required library dependencies installed. It is also necessitates that the currently running kernel's header files are also installed.

Most Linux systems already come installed with a C compiler but some systems do not have their kernel header files installed, thereby requiring their installation. In general, installation of these header files should not greatly change the system's state but invariably some changes will be inevitable. Moreover, compiling the kernel-based memory acquisition driver, which is very small, should also not overly change the system's state.

Pre-compilation of the driver for field use is not a realistic endeavour given the large number of potential Linux kernels currently in use. As such, driver compilation should be carried out immediately prior to dumping system memory on the suspect system. If the suspect system has a rootkit or other malware installed, the potential exists that the compilation of the LKM module could be compromised or rendered ineffective. Fortunately, Linux-based malware does not yet appear to have this capability, although that may change in time.

This tool, although freely available to anyone, is as capable as commercially based software Second Look. A comparison of Second Look is found in the next section. Moreover, this tools is superior to Helix 3 Pro with respect to memory acquisitions.

Upon loading the Fmem driver using the provided script, the *dd* command must be used for memory acquisition. When using the *dd* command, the investigator must specify an upper memory limit for the dumping of memory using the command's *count* parameter. This parameter is itself based on the *bs* (byte size) parameter, which is obtained based on the amount of total detected system memory (see Section 1.3.4 for more details). However, since operating system detected memory, at least under Linux, is always less than the amount of physical RAM, based on information provided by kernel structure */proc/meminfo*, no hardware-specific memory will be included in an *Fmem*-based memory dump.

The tool generated no specific errors requiring user intervention. However, like Second Look, it too requires that the investigator be knowledgeable enough to compile the memory acquisition driver and load it into kernel space.

2.2.4.2 Recommendations

This tool does not generate memory dump-file hashes or automatically fill up and complete Chain of Custody forms. Thus, if this is a requirement, this tool should not be used, at least where Linux is concerned; instead, Helix 3 Pro should be used. If a future version of the tool provided hash generation and simplified report generation capabilities, this tool would be vastly improved.

With the exception of Second Look, this is a highly recommended Linux-based memory acquisition tool investigators should consider using against Linux. It is fast, accurate, efficient and trivial to use, assuming that the prerequisites are in place. Moreover, this tool would meet expectations of most investigators. Its most important limitation is that the tool will only work on 2.6.x kernels (2.4.x systems must be confirmed). However, it succeeds at acquiring memory from all tested 32-bit, 32-bit PAE and 64-bit systems Linux systems.

2.2.5 Second Look analysis

2.2.5.1 Linux analysis

Second Look is by far the most advanced memory acquisition tool currently available for Linux-based operating systems. No support, however, is afforded to 2.4.x kernels. Linux 2.4.x kernel support could not be confirmed although it may be possible to support it since LKM module compilation support is available in some 2.4.x kernels.

Its memory acquisition driver is similar to *Fmem*'s, although there are distinct differences between them. What sets Second Look apart from other memory acquisition tools is its use of Linux kernel structure */proc/iomem*, a structure that identifies every single operating system addressable portion of computer memory, including physical RAM and other addressable memory ranges from hardware-specific computer-attached devices. Combining this information along with its memory acquisition driver and *dd*, the Second Look memory acquisition script is able to acquire all detected operating system RAM.

In all cases based upon experimentation when using Second Look in this work, the size of the memory dump-file always exceeded the size of operating system detected computer memory, as based on kernel structure */proc/meminfo*. However, the extent to which memory the memory dump-file size was larger than detected operating system memory was entirely dependent on the kernel architecture. Specifically, experiments confirm that tests carried out against 32-bit PAE and 64-bit Linux kernels not only acquire all operating system detected RAM, but all physical RAM including hardware-reserved computer memory (RAM set aside for hardware use) and what appeared to be at least, in part, a portion of virtual machine hardware device-specific memory. However, for non-PAE 32-bit kernels, only a small portion of additional memory beyond the limit of operating system memory detection was acquired. Whether this small amount of additional memory is hardware-reserved or hardware device-specific memory is not currently known.

The only downside to this tool is that the investigator must compile the Second Look memory acquisition driver prior to dumping memory. This requires that the suspect system already have a suitable C compiler and the required dependencies are installed. It is also necessitates that the currently running kernel's header files be installed.

Fortunately, most systems already come installed with a C compiler. However, some systems do not have their kernel header files installed, thereby requiring their installation. In general, installation of these kernel header files should not largely change the system's state, but some changes will be inevitable, including memory that will be swapped out to page file. Compiling the kernel-based memory acquisition driver, which is very small, should not overtly change the system's state, but again, some changes cannot be avoided. Pre-compilation of the driver for field use is not a realistic endeavour given the large number of potential Linux kernels that may be encountered. As such, driver compilation should be carried out immediately prior to dumping system memory.

The tool generated no specific errors requiring user intervention. However, like *Fmem*, it too requires that the investigator be knowledgeable enough to compile the memory acquisition driver and load it into kernel space. That said, it is a trivial software tool to use and deploy.

Unlike other commercial software tools examined in this work (Helix 3 Pro and X-Ways Capture) Second Look also comes bundled with a memory analysis toolkit providing both a GUI and CLI interface which installs with very little effort atop Ubuntu 11.04 32-bit. In-depth examination of this, however, is outside the scope of this specific work.

Another point that is worth improving is providing a built-in (through the memory acquisition script) capability to perform on the fly memory dump-file memory hashing.

2.2.5.2 Recommendations

With the exception of *Fmem*, this is a highly recommended Linux-based memory acquisition tool investigators should consider using against Linux. It is fast, accurate, efficient and trivial to use, assuming that the prerequisites are in place. Moreover, this tool is far above the capabilities of Helix 3 Pro, *dd*, *Memdump*, and X-Ways Capture.

However, it will only work on 2.6.x generation of Linux kernels (testing of 2.4.x LKM-enabled kernels is required). Moreover, it successfully acquires memory from 32-bit, 32-bit PAE and 64-bit systems.

When conducting a memory acquisition, if memory dump-file hashing is required, an external tool such as *md5sum*, *md5deep*, *sha1sum*, etc., will have to be used, as this tool does not in any way conduct file hashing. Moreover, this tool cannot be used to generate automatic Chain of Custody forms.

2.2.6 X-Ways Capture analysis

2.2.6.1 Linux analysis

Experimental results have revealed that based exclusively on surface level examinations it appeared as though the X-Ways Capture tool worked correctly. However, this examination was based solely on the memory dump-file size. Detailed analyses revealed that all of the memory dumps files obtained through memory acquisition using this tool contained virtually no data other than binary zero. Only approximately the first MiB of each memory dump contained low-level BIOS data. Beyond, this point, the memory dumps files were effectively devoid of anything else.

The tool not only failed to function as expected but moreover, there was no single instance where it worked correctly. As such, this tool must be deemed a complete failure. Moreover, for a commercially developed disk and memory acquisition-based software tool, a great deal more was expected.

2.2.6.2 Recommendations

Under no circumstances is it suggested to use X-Ways Capture for memory acquisition. All experiments carried out using this tool were a total failure. The tool's disk acquisition capabilities were not examined in this work and as such, no specific comment is provided herein.

3 Conclusion and final tool assessment

Where Solaris x86 memory acquisitions are concerned, *Memdump* is the tool of choice. It works as expected for Solaris 64-bit systems. However, when 32-bit systems are encountered, memory acquisition will transpire up to the memory address where hardware-reserved computer memory is located, often found between 3.3 to 3.5 GiB RAM. Experimentation thus far has clearly demonstrated that memory acquired using this tool is valid and intact.

BSD x86-based systems are also best served by using *Memdump*. All 32-bit systems encountered through experimentation in this work, memory acquisition occurred without issue and was able to fully acquire all memory for each operating system, FreeBSD, NetBSD and OpenBSD up to each system's 32-bit memory limit (4 GiB RAM) for each case.

Where BSD 64-bit systems were concerned, memory acquisition was less straightforward. Under OpenBSD 64-bit, the maximum detected and supported memory size was 4 GiB RAM even though a 64-bit kernel was running. This is a known issue with the default OpenBSD 64-bit kernel, which incidentally is the most likely to be encountered by investigators. For this specific operating system, the memory dump conducted using *Memdump* was acquired without difficulty and it can acquire up to 4 GiB RAM. As for FreeBSD and NetBSD 64-bit operating systems, *Memdump* will successfully acquire all the operating system's memory. With FreeBSD and OpenBSD 64-bit systems, once all the operating system's memory is acquired, up to detected memory limit, the dump-file will continue filling up with binary zero until the partition where the dump-file is located either fills up or the investigator stops the program's execution. Extraction of memory from these dump-files is done by copying out the acquired data up to each operating system's detected memory limit. This operation can be readily carried out using the UNIX *dd* command.

As for Linux, while each of the programs examined herein is applicable to Linux, two tools stand out above the rest. Specifically, for circa 2.6.x kernels (including 32-bit, 32-bit PAE and 64-bit), Second Look is the primary tool of interest. However, *Fmem* has shown itself to be equally capable. The primary difference between the two tools is that Second Look, due to its use of kernel structure */proc/iomem* as its source of system memory addresses, is able to acquire some memory from hardware-specific computer-attached devices whereas *Fmem* does not. This, in of itself, should not be a limiting factor for choosing to use either tool. Specifically, investigators should consider using Second Look if there is the potential for analysing memory dumps as Second Look is bundled with a moderately capable memory analysis system based atop an X Windows GUI. Otherwise, investigators may choose to use *Fmem* at their leisure.

For 2.4.x-based Linux kernels, the investigator has the option to use either *dd* or *Memdump*. With either tool, it is not typically possible to acquire operating system memory against these systems beyond approximately 900 MB RAM. RAM above this limit is non-acquirable, regardless of the tool and use of specific memory devices. LKM-enabled 2.4.x-based kernels may possibly function with additional tools including Helix 3 Pro, Fmem and Second Look, although this must be confirmed.

In conclusion, UNIX-based live memory forensics acquisition is not a straightforward endeavour. Nevertheless, capable tools exist for the aforementioned operating systems and through their use,

as applied to the appropriate operating system they are best suited for, investigators will be able to acquire memory consistently and reproducibly.

References

[1] Halderman, J. Alex, Schoen, Seth D., Heninger, Nadia, et al. Lest We Remember: Cold Boot Attacks on Encryption Keys. Research paper. Published in Proceedings 2008 USENIX Security Symposium. February 2008. Princeton University. http://citp.princeton.edu/pub/coldboot.pdf.

[2] Carbone, Richard. An in-depth analysis of the cold boot attack: Can it be used for sound forensic memory acquisition? Technical memorandum. TM-2010-296. Defence R&D Canada – Valcartier. January 2011. http://pubs.drdc.gc.ca/PDFS/unc105/p534323_A1b.pdf.

[3] BSD Certification Group. BSD Usage Survey. Survey. BSD Certification Group. October 2005. http://www.bsdcertification.org/downloads/pr_20051031_usage_survey_en_en.pdf.

[4] Wikipedia. Physical Address Extension. Online encyclopaedic article. Wikimedia Foundation Inc. July 2011. http://en.wikipedia.org/wiki/Physical_Address_Extension.

[5] The FreeBSD Documentation Project. FreeBSD Handbook. Guide. Release 8.2. The FreeBSD Documentation Project. 2011. http://www.freebsd.org/doc/handbook/kernelconfig-config.html.

[6] Migeon, Jean-Yves. PAE support for native i386. Blog post. Blog.netbsd.org. July 2010. http://blog.netbsd.org/tnf/entry/pae_support_for_native_i386.

[7] Wikipedia. 64-bit. Online encyclopaedic article. Wikimedia Foundation Inc. July 2011. http://en.wikipedia.org/wiki/64-bit.

[8] Wikipedia. Fedora (operating system). Online encyclopaedic article. Wikimedia Foundation Inc. August 2011. http://en.wikipedia.org/wiki/Fedora_(operating_system).

[9] Wikipedia. Ubuntu (operating system). Online encyclopaedic article. Wikimedia Foundation Inc. August 2011. http://en.wikipedia.org/wiki/Ubuntu_(operating_system).

[10] JL. JL's stuff: /dev/crash Driver. Blog. Gleeda.blogspot.com. August 2009. Gleeda.blogspot.com. http://gleeda.blogspot.com/2009/08/devcrash-driver.html.

[11] Van de Ven, Arjan. X86: introduce /dev/mem restrictions with a config option. LWN.net. Jan 2008. http://lwn.net/Articles/267427/.

[12] Lineberry, Anthony. Malicious Code Injection via /dev/mem. Technical report. Anthony Lineberry. March 2009.

[13] Rintel, Lubomir. Bug 492803 – Please disable CONFIG_STRICT_DEVMEM. Bug report. Redhat Bugzilla. Unknown date. https://bugzilla.redhat.com/show_bug.cgi?id=492803.

[14] Engelhardt, Jan. LKML: Arjan van de Ve: Re: [PATCH] make /dev/kmem a config option. Blog. Linux Kernel Mailing List. February 2008. https://lkml.org/lkml/2008/2/10/328.

[15] Ubuntu. Security/Features – Ubuntu Wiki. Informational web site. Ubuntu. June 2011. https://wiki.ubuntu.com/Security/Features.

[16] Man Pages Project. Proc(5) Man Page. Linux/UNIX man page. Man Pages Project. October 2010.

[17] Common Criteria. Certified Products: Operating systems. Informational web listing. Common Criteria. Unknown date. http://www.commoncriteriaportal.org/products/.

[18] Wikipedia. Sun386i. Online encyclopaedic article. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Sun386i.

[19] Wikipedia. Solaris (operating system). Online encyclopaedic article. Wikimedia Foundation Inc. August 2011. http://en.wikipedia.org/wiki/Solaris_(operating_system).

[20] Bezroukov, Nikolai. Solaris History. Informational web site. Softpanorama.org. May 2011. http://www.softpanorama.org/Solaris/solaris_history.shtml.

[21] Oracle Corporation. Solaris 10 10/09 Release Notes. Product documentation. Oracle Corporation. 2010. http://download.oracle.com/docs/cd/E19253-01/821-0381/ghjhr/index.html.

[22] Oracle Corporation. Mem(7D) Man Page. Solaris man page. Oracle Corporation. February 2002.

[23] Wikipedia. NetBSD. Online encyclopaedic article. Wikimedia Foundation Inc. July 2011. http://en.wikipedia.org/wiki/Netbsd.

[24] Wikipedia. OpenBSD. Online encyclopaedic article. Wikimedia Foundation Inc. July 2011. http://en.wikipedia.org/wiki/Openbsd.

[25] FreeBSD. Mem(4) Man Page. Man page. FreeBSD Kernel Interface Manual. FreeBSD. October 2004.

[26] NetBSD. Mem(4) Man Page. Man page. NetBSD/i386 Kernel Interface Manual. NetBSD. June 1993.

[27] OpenBSD. Mem(4) Man Page. Man page. OpenBSD Programmer's Manual (i386). OpenBSD. May 2007.

[28] Farmer, Dan and Venema, Wietse. Forensic Discovery. Book. First edition. ISBN-13 No.: 978-0201634976. January 2005. http://www.porcupine.org/forensics/forensic-discovery/.

[29] Man Pages Project. Mem(4) Man Page. Linux/UNIX man page. Man Pages Project. November 1992.

[30] Oracle Corporation. Man pages section 7: Device and Network Interfaces. Reference manual. Part No.: 819-2254-33. Oracle Corporation. 2010. http://download.oracle.com/docs/cd/E19082-01/819-2254/.

[31] Oracle Corporation. Physmem(7D) Man Page. Solaris man page. Oracle Corporation. November 2006.

[32] X-Ways Software Technology AG. X-Ways Capture. Product documentation. Version 1.2. Unknown date. X-Ways Software Technology AG.

[33] Kollár, Ivor. Forensic RAM dump image analyser. Master's thesis. Charles University in Prague, Faculty of Mathematics and Physics, Department of Software engineering. 2010.

[34] Wikipedia. Caldera OpenLinux. Online encyclopaedic article. Wikimedia Foundation Inc. September 2011. http://en.wikipedia.org/wiki/Caldera_OpenLinux.

[35] The Knotter. System memory dumps on Linux. Informational web site. May 2010. http://www.theknotter.net/system-memory-dumps-on-linux/.

[36] Kernel developers. THE /proc FILESYSTEM. Linux kernel documentation. June 2009. www.kernel.org/doc/Documentation/filesystems/proc.txt.

[37] Wikipedia. Loadable kernel module. Online encyclopaedic article. Wikimedia Foundation Inc. September 2011. http://en.wikipedia.org/wiki/Loadable_kernel_module.

[38] Gentoo Wiki. MTRR. Informational web site. Gentoo Wiki. July 2011. http://en.gentoo-wiki.com/wiki/MTRR.

[39] Noon, David W. In response to - Why does code fail to find *exact* amount of RAM?? Informational web site. Mombu.com. http://www.mombu.com/programming/hpux/t-why-does-code-fail-to-find-exact-amouut-of-ram-1573802.html.

[40] Tappert, Andrew, and Carbone, Richard. Technical exchange with Pike Works for Second Look. Technical electronic mail-based exchange. October 2011.

[41] Wikipedia. Observer effect (information technology). Online encyclopaedia. Wikimedia Foundation. April 2012. http://en.wikipedia.org/wiki/Observer_effect_(information_technology).

[42] Wikipedia. Observer effect (physics). Online encyclopaedic article. Wikimedia Foundation Inc. October 2012. http://en.wikipedia.org/wiki/Observer_effect_(physics).

[43] Carbone,Richard and Charpentier, Robert. Life-Cycle Support for Information Systems Based on Free and Open Source Software. Technical paper. Document No.: I-136. Presented to: 11th International Command and Control Research and Technology Symposium, Cambridge UK. September 2006.

This page intentionally left blank.

Annex A Details concerning computer systems used for experimentation

A.1 Dedicated virtualization experimentation workstation

In order to create various VirtualBox virtual machines and carry out memory acquisition against said systems a dedicated host computer platform was needed. Its specifications are as follows below in Table 7:

Table 7. Dedicated computer system for carrying out memory acquisitions against BSD, Linux and Solaris based operating systems.

Computer model	Dell Precision 690 Workstation
Processors	Dual Xeon 3.20 GHz w/HyperThreading (8 logical processors)
Physical RAM	22.00 GiB RAM
Swap	None
Operating System	Linux Fedora Core 14, 64-bit
Virtualization Software	Oracle VirtualBox 4.1.0 with Extension Pack
Linux kernel	Kernel 2.6.35.12-90.fc14.x86_64 #1 SMP
Graphics adapter	NVidia GeForce GTX 460
Graphics driver	NVidia driver 270.41.06
Monitors	1) Dell E196FP LCD display (19”) 2) BenQ FP992 LCD display (19”)
Floppy	1.44 MB floppy drive
USB	8 USB ports
Keyboard	USB US English keyboard
Mouse	USB optical mouse
FireWire	2 FireWire ports (no attached devices)
CD Drive	Hitachi CD-RW drive
DVD Drive	Philips CD-RW/DVD-RW drive
Hard drives	1) 1.5 TB Seagate 7,200 RPM SATA drive (system disk) 2) 3x 2 TB Hitachi 7,200 SATA drives in RAID 5 configuration with one spare yielding 4 TB disk space (software RAID

	connected to Vantec SATA controller) 3) 8x 2 TB Seagate 7,200 RPM SATA drive in RAID 5 configuration with no spare yielding 14 TB disk space (software RAID connected to Vantec SATA controller)
Host adapters	2x Vantec PCI Express E-SATA host adapter
Sound card	Sigma Tel HD sound card
Network cards	1) Broadcom NetXtreme Gigabit Ethernet 2) 1394 Net Adapter

A.2 Virtualization validation system

In order to validate the results obtained using the Dell Precision 690 workstation (see Table 7) the following validation system and was configured as shown below in Table 8:

Table 8. Result validation computer system for assessing memory acquisition outcomes.

Computer model	Dell Studio XPS Desktop 9100
Processors	i7 Quad-core 2.80 GHz w/HyperThreading (8 logical processors)
Physical RAM	18.00 GiB RAM
Swap	36.00 GB on exFAT partition atop PCI Express RevoDrive SSD
Operating System	Windows 7 64-bit Service Pack 1
Virtualization Software	Oracle VirtualBox 4.1.0 with Extension Pack
Graphics adapter	ATI Radeon 5670 HD
Graphics driver	ATI driver version 8.831.2.0
Monitors	1) Philips 19” 2) Philips 19”
Floppy	None
USB	7 USB ports
Keyboard	USB US English keyboard
Mouse	Dell USB optical mouse
FireWire	1 FireWire ports (no attached devices)
Optical Drives	1) LG Blu-Ray (RO)/DVD/CD drive 2) Plextor Blu-Ray RW

Hard drives	1) 2 x 750 MB Seagate 7,200 RPM SATA drive in RAID 1 connected to Intel ICHR8 SATA RAID controller) 2) PCI Express RevoDrive SSD 100 GB
Host adapters	None
Sound card	Creative Labs PCI Express Sound Blaster X-I sound card
Network cards	1) Broadcom NetXtreme Gigabit Ethernet 2) RealTek RTL8168D Gigabit Ethernet

Annex B Details concerning VirtualBox and operating system configuration for experimentation

B.1 VirtualBox hardware-specific configurations

This sub-annex examines the various hardware-specific configurations for the virtual machines built using Oracle VirtualBox.

B.1.1 VirtualBox configurations for OpenBSD

The following technical details are specific to the configuration of the OpenBSD-based operating systems (32 and 64-bit) experimented upon in this work.

Table 9. OpenBSD VirtualBox virtual machine-specific configuration details.

VirtualBox configuration	OpenBSD (32-bit)	OpenBSD (64-bit)
VM built using VirtualBox version	4.1.0	4.1.0
Operating system version	4.9	4.9
VirtualBox PAE/NX enabled	Yes	Yes
VirtualBox IO APIC enabled	Yes	Yes
VirtualBox allocated memory	8,388,608 KiB RAM	8,388,608 KiB RAM
VirtualBox no. allocated processors	2 processors	2 processors
VirtualBox floppy drives allocated	None	None
VirtualBox optical drives allocated	1 drive	1 drive
VirtualBox no. monitors allocated	1 monitor	1 monitor
VirtualBox allocated video memory	128 MiB	128 MiB
VirtualBox 3D acceleration enabled	Yes	Yes
VirtualBox 2D acceleration enabled	No	No
VirtualBox network adapter enabled	Yes/default adapter	Yes/default adapter
VirtualBox sound adapter enabled	Yes/default adapter	Yes/default adapter
VirtualBox serial ports enabled	No	No
VirtualBox USB enabled	Yes	Yes
VirtualBox USB 2.0 (EHCI) enabled	Yes	Yes

B.1.2 VirtualBox configurations for NetBSD

The following technical details are specific to the configuration of the NetBSD-based operating systems (32 and 64-bit) experimented upon in this work.

Table 10. NetBSD VirtualBox virtual machine-specific configuration details.

VirtualBox configuration	NetBSD (32-bit)	NetBSD (64-bit)
VM built using VirtualBox version	4.1.0	4.1.0
Operating system version	5.1	5.1
VirtualBox PAE/NX enabled	Yes	Yes
VirtualBox IO APIC enabled	Yes	Yes
VirtualBox allocated memory	8,388,608 KiB RAM	8,388,608 KiB RAM
VirtualBox no. allocated processors	2 processors	2 processors
VirtualBox floppy drives allocated	None	None
VirtualBox optical drives allocated	1 drive	1 drive
VirtualBox no. monitors allocated	1 monitor	1 monitor
VirtualBox allocated video memory	128 MiB	128 MiB
VirtualBox 3D acceleration enabled	Yes	Yes
VirtualBox 2D acceleration enabled	No	No
VirtualBox network adapter enabled	Yes/default adapter	Yes/default adapter
VirtualBox sound adapter enabled	Yes/default adapter	Yes/default adapter
VirtualBox serial ports enabled	No	No
VirtualBox USB enabled	Yes	Yes
VirtualBox USB 2.0 (EHCI) enabled	Yes	Yes

B.1.3 VirtualBox configurations for FreeBSD

The following technical details are specific to the configuration of the FreeBSD-based operating systems (32 and 64-bit) experimented upon in this work.

Table 11. FreeBSD VirtualBox virtual machine-specific configuration details.

VirtualBox configuration	FreeBSD (32-bit)	FreeBSD (64-bit)
VM built using VirtualBox version	4.1.0	4.1.0
Operating system version	8.2	8.2
VirtualBox PAE/NX enabled	Yes	Yes
VirtualBox IO APIC enabled	Yes	Yes
VirtualBox allocated memory	8,388,608 KiB RAM	8,388,608 KiB RAM
VirtualBox no. allocated processors	2 processors	2 processors
VirtualBox floppy drives allocated	None	None
VirtualBox optical drives allocated	1 drive	1 drive
VirtualBox no. monitors allocated	1 monitor	1 monitor
VirtualBox allocated video memory	128 MiB	128 MiB
VirtualBox 3D acceleration enabled	Yes	Yes
VirtualBox 2D acceleration enabled	No	No
VirtualBox network adapter enabled	Yes/default adapter	Yes/default adapter
VirtualBox sound adapter enabled	Yes/default adapter	Yes/default adapter
VirtualBox serial ports enabled	No	No
VirtualBox USB enabled	Yes	Yes
VirtualBox USB 2.0 (EHCI) enabled	Yes	Yes

B.1.4 VirtualBox configurations for Ubuntu Linux

The following technical details are specific to the configuration of the Ubuntu-based operating systems (32-bit, 32-bit PAE and 64-bit) experimented upon in this work.

Table 12. Ubuntu VirtualBox virtual machine-specific configuration details.

VirtualBox configuration	Ubuntu 11.04 (32-bit)	Ubuntu 11.04 (32-bit PAE)	Ubuntu 11.04 (64-bit)
VM built using VirtualBox version	4.1.0	4.1.0	4.1.0
Operating system version	11.04	11.04	11.04
VirtualBox PAE/NX enabled	Yes	Yes	Yes

VirtualBox IO APIC enabled	Yes	Yes	Yes
VirtualBox allocated memory	8,388,608 KiB RAM	8,388,608 KiB RAM	8,388,608 KiB RAM
VirtualBox no. allocated processors	2 processors	2 processors	2 processors
VirtualBox floppy drives allocated	None	None	None
VirtualBox optical drives allocated	1 drive	1 drive	1 drive
VirtualBox no. monitors allocated	1 monitor	1 monitor	1 monitor
VirtualBox allocated video memory	128 MiB	128 MiB	128 MiB
VirtualBox 3D acceleration enabled	Yes	Yes	Yes
VirtualBox 2D acceleration enabled	No	No	No
VirtualBox network adapter enabled	Yes/default adapter	Yes/default adapter	Yes/default adapter
VirtualBox sound adapter enabled	Yes/default adapter	Yes/default adapter	Yes/default adapter
VirtualBox serial ports enabled	No	No	No
VirtualBox USB enabled	Yes	Yes	Yes
VirtualBox USB 2.0 (EHCI) enabled	Yes	Yes	Yes

B.1.5 VirtualBox configurations for Red Hat Linux

The following technical details are specific to the configuration of the Red Hat Linux-based operating system (32-bit PAE) experimented upon in this work.

Table 13. Red Hat Linux VirtualBox virtual machine-specific configuration details.

VirtualBox configuration	Red Hat (32-bit PAE)
VM built using VirtualBox version	4.1.0
Operating system version	9.0
VirtualBox PAE/NX enabled	Yes
VirtualBox IO APIC enabled	Yes
VirtualBox allocated memory	8,388,608 KiB RAM
VirtualBox no. allocated processors	2 processors

VirtualBox floppy drives allocated	None
VirtualBox optical drives allocated	1 drive
VirtualBox no. monitors allocated	1 monitor
VirtualBox allocated video memory	128 MiB
VirtualBox 3D acceleration enabled	Yes
VirtualBox 2D acceleration enabled	No
VirtualBox network adapter enabled	Yes/default adapter
VirtualBox sound adapter enabled	Yes/default adapter
VirtualBox serial ports enabled	No
VirtualBox USB enabled	Yes
VirtualBox USB 2.0 (EHCI) enabled	Yes

B.1.6 VirtualBox configurations for Fedora Core

The following technical details are specific to the configuration of the Fedora Core-based operating systems (32-bit PAE and 64-bit) experimented upon in this work.

Table 14. Fedora Core VirtualBox virtual machine-specific configuration details.

VirtualBox configuration	Fedora Core (32-bit PAE)	Fedora Core (64-bit)
VM built using VirtualBox version	4.1.0	4.1.0
Operating system version	15	15
VirtualBox PAE/NX enabled	Yes	Yes
VirtualBox IO APIC enabled	Yes	Yes
VirtualBox allocated memory	8,388,608 KiB RAM	8,388,608 KiB RAM
VirtualBox no. allocated processors	2 processors	2 processors
VirtualBox floppy drives allocated	None	None
VirtualBox optical drives allocated	1 drive	1 drive
VirtualBox no. monitors allocated	1 monitor	1 monitor
VirtualBox allocated video memory	128 MiB	128 MiB
VirtualBox 3D acceleration enabled	Yes	Yes
VirtualBox 2D acceleration enabled	No	No

VirtualBox network adapter enabled	Yes/default adapter	Yes/default adapter
VirtualBox sound adapter enabled	Yes/default adapter	Yes/default adapter
VirtualBox serial ports enabled	No	No
VirtualBox USB enabled	Yes	Yes
VirtualBox USB 2.0 (EHCI) enabled	Yes	Yes

B.1.7 VirtualBox configurations for Solaris

The following technical details are specific to the configuration of the Solaris-based operating systems (32-bit PAE and 64-bit) experimented upon in this work.

Table 15. Solaris VirtualBox virtual machine-specific configuration details.

VirtualBox configuration	Solaris (32-bit PAE)	Solaris (64-bit)
VM built using VirtualBox version	4.1.0	4.1.0
Operating system version	Release 9/10	Release 9/10
VirtualBox PAE/NX enabled	Yes	Yes
VirtualBox IO APIC enabled	Yes	Yes
VirtualBox allocated memory	8,388,608 KiB RAM	8,388,608 KiB RAM
VirtualBox no. allocated processors	2 processors	2 processors
VirtualBox floppy drives allocated	None	None
VirtualBox optical drives allocated	1 drive	1 drive
VirtualBox no. monitors allocated	1 monitor	1 monitor
VirtualBox allocated video memory	128 MiB	128 MiB
VirtualBox 3D acceleration enabled	Yes	Yes
VirtualBox 2D acceleration enabled	No	No
VirtualBox network adapter enabled	Yes/default adapter	Yes/default adapter
VirtualBox sound adapter enabled	Yes/default adapter	Yes/default adapter
VirtualBox serial ports enabled	No	No
VirtualBox USB enabled	Yes	Yes
VirtualBox USB 2.0 (EHCI) enabled	Yes	Yes

B.2 VirtualBox guest operating system details

This sub-annex examines both the virtualized hardware and operating system specifics for each virtual machine built using Oracle VirtualBox.

B.2.1 Guest operating system details for OpenBSD

The following table focuses on the detected hardware for the various OpenBSD virtualised guest operating systems (32 and 64-bit) experimented upon in this work.

Table 16. OpenBSD guest virtualized hardware and operating system details.

Operating system details	OpenBSD (32-bit)	OpenBSD (64-bit)
Operating system version	4.9	4.9
Operating system kernel version	OpenBSD openbsd4.9_32bit 4.9 GENERIC.MP#794 i386	OpenBSD openbsd4.9_64bit 4.9 GENERIC.MP#819 amd64
Memory detected	3,669,532 KiB	3,668,928 KiB
Processors detected	2 processors	2 processors
Hard drives detected	1 drive	1 drive
Optical drives detected	1 drive	1 drive
Network adapter functioning correctly	Yes	Yes
USB detected	Yes	Yes
VirtualBox Guest Additions installed	No	No
Version	N/A	N/A

B.2.2 Guest operating system details for NetBSD

The following table focuses on the detected hardware for the various NetBSD virtualised guest operating systems (32 and 64-bit) experimented upon in this work.

Table 17. NetBSD guest virtualized hardware and operating system details.

Operating system details	NetBSD (32-bit)	NetBSD (64-bit)
Operating system version	5.1	5.1
Operating system kernel version	NetBSD 5.1 (GENERIC) #0: Sun Nov 7 14:39:56 UTC 2010	NetBSD 5.1 (GENERIC) #0: Sat Nov 6 13:19:33 UTC 2010

Memory detected	3,669,564 KiB	8,388,156 KiB
Processors detected	2 processors	2 processors
Hard drives detected	1 drive	1 drive
Optical drives detected	1 drive	1 drive
Network adapter functioning correctly	Yes	Yes
USB detected	Yes	Yes
VirtualBox Guest Additions installed	No	No
Version	N/A	N/A

B.2.3 Guest operating system details for FreeBSD

The following table focuses on the detected hardware for the various FreeBSD virtualised guest operating systems (32 and 64-bit) experimented upon in this work.

Table 18. FreeBSD guest virtualized hardware and operating system details.

Operating system details	FreeBSD (32-bit)	FreeBSD (64-bit)
Operating system version	8.2	8.2
Operating system kernel version	FreeBSD 8.2-RELEASE #0: Fri Feb 18 02:24:46 UTC 2011	FreeBSD 8.2-RELEASE #0: Thu Feb 17 02:41:51 UTC 2011
Memory detected	3,657,120 KiB	8,375,004 KiB
Processors detected	2 processors	2 processors
Hard drives detected	1 drive	1 drive
Optical drives detected	1 drive	1 drive
Network adapter functioning correctly	Yes	Yes
USB detected	Yes	Yes
VirtualBox Guest Additions installed	No	No
Version	N/A	N/A

B.2.4 Guest operating system details for Ubuntu Linux

The following table focuses on the detected hardware for the various Ubuntu virtualised guest operating systems (32-bit, 32-bit PAE and 64-bit) experimented upon in this work.

Table 19. Ubuntu guest virtualized hardware and operating system details.

Operating system details	Ubuntu (32-bit)	Ubuntu (32-bit PAE)	Ubuntu (64-bit)
Operating system version	11.04	11.04	11.04
Operating system kernel version	2.6.38-10-generic #46-Ubuntu SMP Tue Jun 28 15:05:41 UTC 2011 i686 i686 i386 GNU/Linux	2.6.38-10-generic-pae #46-Ubuntu SMP Tue Jun 28 16:54:49 UTC 2011 i686 i686 i386 GNU/Linux	2.6.38-10-generic #42-Ubuntu SMP Mon Apr 11 03:31:24 UTC 2011 x86_64 x86_64 x86_64 GNU/Linux
Memory detected	3,613,268 KiB	8,265,044 KiB	8,194,124 KiB
Processors detected	2 processors	2 processors	2 processors
Hard drives detected	1 drive	1 drive	1 drive
Optical drives detected	1 drive	1 drive	1 drive
Network adapter functioning correctly	Yes	Yes	Yes
USB detected	Yes	Yes	Yes
VirtualBox Guest Additions installed	Yes	Yes	Yes
Version	4.1.0	4.1.0	4.1.0

B.2.5 Guest operating system details for Red Hat Linux

The following table focuses on the detected hardware for the following Red Hat Linux virtualised guest operating system (32-bit PAE) experimented upon in this work.

Table 20. Red Hat guest virtualized hardware and operating system details.

Operating system details	Red Hat Linux (32-bit PAE)
Operating system version	9.0
Operating system kernel version	2.4.20-8bigmem #1 SMP Thu Mar 13 17:32:29 EST 2003 i686 i686 i386 GNU/Linux
Memory detected	8,253,495 KiB
Processors detected	2 processors
Hard drives detected	1 drive
Optical drives detected	1 drive

Network adapter functioning correctly	Yes
USB detected	Inconsistent
VirtualBox Guest Additions installed	Yes
Version	4.1.0

B.2.6 Guest operating system details for Fedora Core Linux

The following table focuses on the detected hardware for the various Fedora Core virtualised guest operating systems (32-bit PAE and 64-bit) experimented upon in this work.

Table 21. Fedora Core guest virtualized hardware and operating system details.

Operating system details	Fedora Core (32-bit PAE)	Fedora Core (64-bit)
Operating system version	15	15
Operating system kernel version	2.6.38.8-35.fc15.i686.PAE #1 SMP Wed Jul 6 14:29:06 UTC 2011 i686 i686 i386 GNU/Linux	2.6.38.8-35.fc15.x86_64 #1 SMP Wed Jul 6 13:58:54 UTC 2011 x86_64 x86_64 x86_64 GNU/Linux
Memory detected	8,266,212 KiB	8,194,780 KiB
Processors detected	2 processors	2 processors
Hard drives detected	1 drive	1 drive
Optical drives detected	1 drive	1 drive
Network adapter functioning correctly	Yes	Yes
USB detected	Yes	Yes
VirtualBox Guest Additions installed	Yes	Yes
Version	4.1.0	4.1.0

B.2.7 Guest operating system details for Solaris

The following table focuses on the detected hardware for the various Solaris virtualised guest operating systems (32-bit PAE and 64-bit) experimented upon in this work.

Table 22. Solaris guest virtualized hardware and operating system details.

Operating system details	Solaris 32-bit PAE	Solaris 64-bit
Operating system version	10/09	10/09

Operating system kernel version	Generic_142910-17 32-bit	Generic_142910-17 64-bit
Memory detected as per [39]	8,388,156 KiB	8,388,156 KiB
Processors detected	2 processors	2 processors
Hard drives detected	1 drive	1 drive
Optical drives detected	1 drive	1 drive
Network adapter functioning correctly	Yes	Yes
USB detected	Yes	Yes
VirtualBox Guest Additions installed	Yes	Yes
Version	4.1.0	4.1.0

Annex C Experimental results

C.1 Dd

In this Annex, the experimental results obtained using the *dd* tool are examined. This tool was used against all of the various 32 and 64-bit UNIX operating systems examined herein.

C.1.1 FreeBSD

Table 23. Experimental memory dump results for FreeBSD 32-bit using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,744,890,880 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/kmem
Time	0 second
Dump size	0 byte
Number of 7, 8, 16 and 32-bit strings	N/A – unable to acquire
Highest 7, 8, 16 and 32-bit string byte offset	N/A – unable to acquire
Device	/dev/mem
Time	1,396 seconds
Dump size	10,392,633,344 bytes / filled up disk partition
Number of 7, 8, 16 and 32-bit strings	47,116,348 / 292,006,990 / 13,525 / 10,341
Highest 7, 8, 16 and 32-bit string byte offset	10,374,183,003 / 10,374,183,003 / 10,248,630,264 / 10,352,714,388
Memory analysis commentary	The tool appears to have correctly captured the first 4 GiB RAM from device */dev/mem*. Memory acquisition from device */dev/kmem* failed. Memory acquired from */dev/mem* actually captured 4 GiB, up to the 32-bit memory limitation inherent in a 32-bit operating

	system. Although the memory dump size was much larger than 4 GiB, the first 3,571 MiB of detected system RAM was captured. This was then followed by hardware-reserved computer memory (RAM set aside for hardware use) found between 3,571 MiB and 4,096 MiB. The next 4 GiB RAM was a repetition of the first 4 GiB. In addition, the next memory segment over 8 GiB was again a partial repetition of the first 4 GiB RAM, until the memory dump-file filled up the disk partition.

Table 24. Experimental memory dump results for FreeBSD 64-bit using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,576,004,096 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/kmem
Time	0 second
Dump size	0 byte
Number of 7, 8, 16 and 32-bit strings	N/A - unable to acquire memory
Highest 7, 8, 16 and 32-bit string byte offset	N/A - unable to acquire memory
Device	/dev/mem
Time	870 seconds
Dump size	9,663,676,416 bytes / filled up disk partition
Number of 7, 8, 16 and 32-bit strings	1,309,892 / 13,800,449 / 3,165 / 2,515
Highest 7, 8, 16 and 32-bit string byte offset	9,126,745,773 / 9,126,805,474 / 8,711,590,584 / 7,347,200,000
Memory analysis commentary	The tool appears to have correctly captured the first 8 GiB RAM from device */dev/mem*. Memory acquisition from device */dev/kmem* failed. Memory acquired from */dev/mem* actually appeared to have captured the detected 8,178 MiB RAM. Although the memory dump size was larger than this by well over another 1 GiB it

	was found that memory captured between 8,178 MiB and 8,192 MiB RAM contained what appeared to be hardware-reserved computer memory (RAM set aside for hardware use). All memory above the allocated 8 GiB RAM appeared to be virtual machine hardware device-specific memory (non-RAM hardware memory). However, additional in-depth verification is necessary.

C.1.2 NetBSD

Table 25. Experimental memory dump results for NetBSD 32-bit using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,757,633,536 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/kmem
Time	0 second
Dump size	0 byte
Number of 7, 8, 16 and 32-bit strings	N/A - unable to acquire memory
Highest 7, 8, 16 and 32-bit string byte offset	N/A - unable to acquire memory
Device	/dev/mem
Time	827 seconds
Dump size	10,556,828,032 bytes / filled up disk partition
Number of 7, 8, 16 and 32-bit strings	787,289 / 16,422,908 / 1,814 / 1,255
Highest 7, 8, 16 and 32-bit string byte offset	10,559,291,229 / 10,563,698,688 / 10,505,008,698 / 10,531,933,824
Memory analysis commentary	The tool appears to have correctly captured the first 4 GiB RAM from device */dev/mem*. Memory acquisition from device */dev/kmem* failed. Memory acquired from */dev/mem* actually captured 4 GiB, up

	to the 32-bit memory limitation inherent in a 32-bit operating system. Although the memory dump size was much larger than 4 GiB, the first 3,583 MiB of detected system RAM was captured. This was then followed by hardware-reserved computer memory (RAM set aside for hardware use) found between 3,583 MiB and 4,096 MiB. The next 4 GiB RAM was a repetition of the first 4 GiB. In addition, the next memory segment over 8 GiB was again a partial repetition of the first 4 GiB RAM, until the memory dump-file filled up the disk partition.

Table 26. Experimental memory dump results for NetBSD 64-bit using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,589,471,744 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/kmem
Time	0 second
Dump size	0 byte
Number of 7, 8, 16 and 32-bit strings	N/A - unable to acquire memory
Highest 7, 8, 16 and 32-bit string byte offset	N/A - unable to acquire memory
Device	/dev/mem
Time	683 seconds
Dump size	10,556,828,032 bytes / filled up disk partition
Number of 7, 8, 16 and 32-bit strings	299,574 / 20,351,704 / 339 / 306
Highest 7, 8, 16 and 32-bit string byte offset	9,126,778,436 / 9,126,778,436 / 9,118,537,174 / 9,085,660,816
Memory analysis commentary	The tool appears to have correctly captured the first 8 GiB RAM from device */dev/mem*. Memory acquisition from device */dev/kmem* failed. Memory acquired from */dev/mem* actually appears to have captured the detected 8,191 MiB RAM. Although the memory

	dump size was larger than this by well over another 1 GiB it was found that memory captured between 8,191 MiB and 8,192 MiB RAM contained what appeared to be hardware-reserved computer memory (RAM set aside for hardware use). All memory above the allocated 8 GiB RAM appeared to be virtual machine hardware device-specific memory (non-RAM hardware memory). However, additional in-depth verification is necessary.

C.1.3 OpenBSD

Table 27. Experimental memory dump results for OpenBSD 32-bit using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,756,982,272 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/kmem
Time	0 second
Dump size	0 byte
Number of 7, 8, 16 and 32-bit strings	N/A - unable to acquire memory
Highest 7, 8, 16 and 32-bit string byte offset	N/A - unable to acquire memory
Device	/dev/mem
Time	450 seconds
Dump size	4,294,967,292 bytes
Number of 7, 8, 16 and 32-bit strings	138,095 / 1,414,626 / 659 / 272
Highest 7, 8, 16 and 32-bit string byte offset	4,294,967,285 / 4,294,967,284 / 4,294,902,744 / 3,756,925,372
Memory analysis commentary	The tool appears to have correctly captured the first 4 GiB RAM from device */dev/mem*. Memory acquisition from device */dev/kmem* failed.

	Memory acquired from */dev/mem* actually captured 4 GiB, up to the 32-bit memory limitation inherent in a 32-bit operating system. Although detected system memory was 3,582 MiB RAM, the memory dump size was 4 GiB in size. This difference in size was attributable to what appeared to be hardware-reserved computer memory (RAM set aside for hardware use) between 3,582 MiB and 4,096 MiB RAM.

Table 28. Experimental memory dump results for OpenBSD 64-bit using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,756,982,272 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/kmem
Time	0 second
Dump size	0 byte
Number of 7, 8, 16 and 32-bit strings	N/A - unable to acquire memory
Highest 7, 8, 16 and 32-bit string byte offset	N/A - unable to acquire memory
Device	/dev/mem
Time	103 seconds
Dump size	4,294,967,292 bytes
Number of 7, 8, 16 and 32-bit strings	176,770 / 1,623,895 / 630 / 303
Highest 7, 8, 16 and 32-bit string byte offset	4,294,967,285 / 4,294,967,284 / 4,294,902,744 / 3,755,839,952
Memory analysis commentary	The tool appears to have correctly captured the first 4 GiB RAM from device */dev/mem*. Memory acquisition from device */dev/kmem* failed. Memory acquired from */dev/mem* actually captured 4 GiB, not what would be expected from a 64-bit operating system that has been allocated 8 GiB RAM. Although detected system memory is 3,582 MiB RAM, the memory dump size was 4 GiB in size. This difference in size was attributable to what appeared to be

	hardware-reserved computer memory (RAM set aside for hardware use) between 3,582 MiB and 4,096 MiB RAM.

C.1.4 Red Hat Linux

Table 29. Experimental memory dump results for Red Hat Linux using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,451,578,880 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/proc/kcore
Time	7 seconds
Dump size	950,738,944 bytes
Number of 7, 8, 16 and 32-bit strings	5,342,509 / 14,085,705 / 4,648 / 12,783
Highest 7, 8, 16 and 32-bit string byte offset	950,734,712 / 950,734,712 / 948,255,466 / 939,438,228
Device	/dev/mem
Time	8 seconds
Dump size	939,524,096 bytes
Number of 7, 8, 16 and 32-bit strings	5,420,859 / 15,549,449 / 4,689 / 13,161
Highest 7, 8, 16 and 32-bit string byte offset	939,484,154 / 939,516,268 / 909,028,468 / 939,434,132
Device	/dev/kmem
Time	0 second
Dump size	0 byte
Number of 7, 8, 16 and 32-bit strings	N/A - unable to acquire memory
Highest 7, 8, 16 and 32-bit string byte offset	N/A - unable to acquire memory
Memory analysis commentary	The tool appears to have correctly captured the first 906 MiB and 896 MiB RAM from devices */proc/kcore* and */dev/mem*,

	respectively. Memory acquisition from device */dev/kmem* failed. Although the operating system detected far more memory than the size of either dump-file, 906 MiB and 896 MiB, respectively, the operating system could not furnish more memory for acquisition for unknown reasons. Both dump-files appeared to contain valid and intact computer memory. Neither dump-file contained what appeared to be hardware-reserved computer memory (RAM set aside for hardware use) or virtual machine hardware device-specific memory (non-RAM hardware memory). Memory from */proc/kcore* was in ELF format while memory from */dev/mem* was not.

C.1.5 Ubuntu Linux

Table 30. Experimental memory dump results for Ubuntu Linux 32-bit using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,699,986,432 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/proc/kcore
Time	9 seconds
Dump size	1,065,349,120 bytes
Number of 7, 8, 16 and 32-bit strings	2,145,703 / 6,575,304 / 535 / 125
Highest 7, 8, 16 and 32-bit string byte offset	1,016,082,403 / 1,059,073,292 / 942,558,044 / 942,490,484
Device	/dev/mem
Time	<1 second
Dump size	1,052,672 bytes
Number of 7, 8, 16 and 32-bit strings	2,042 / 17,154 / 11 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,052,521 / 1,052,668 / 984,054 / 0

Memory analysis commentary	The tool appears to have correctly captured the first 1,015 MiB and 1 MiB RAM from devices */proc/kcore* and */dev/mem*, respectively. Although the operating system detected far more memory than the size of either dump-file, 1,015 MiB and 1 MiB, respectively, the operating system could not furnish more memory for acquisition for unknown reasons. The former dump-file contained what appeared to be valid and intact memory while the latter only contained low-memory BIOS-related data. Moreover, neither dump-file contained what appeared to be hardware-reserved computer memory (RAM set aside for hardware use) or virtual machine hardware device-specific memory (non-RAM hardware memory). Furthermore, memory from */proc/kcore* was in ELF format while memory from */dev/mem* was not. Finally, the memory dump-file captured from */dev/mem* was too small to be of any use.

Table 31. Experimental memory dump results for Ubuntu Linux 32-bit PAE using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,463,405,056 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/proc/kcore
Time	10 seconds
Dump size	1,069,543,424 bytes
Number of 7, 8, 16 and 32-bit strings	279,869 / 7,608,348 / 534 / 117
Highest 7, 8, 16 and 32-bit string byte offset	1,066,417,677 / 1,066,418,163 / 948,012,468 / 945,870,096
Device	/dev/mem
Time	<1 second
Dump size	1,052,672 bytes
Number of 7, 8, 16 and 32-bit	2035 / 17136 / 11 / 0

strings	
Highest 7, 8, 16 and 32-bit string byte offset	1,052,663 / 1,052,659 / 984,054 / 0
Memory analysis commentary	The tool appears to have correctly captured the first 1,019 MiB and 1 MiB RAM from devices */proc/kcore* and */dev/mem*, respectively. Although the operating system detected far more memory than the size of either dump-file, 1,019 MiB and 1 MiB, respectively, the operating system could not furnish more memory for acquisition for unknown reasons. The former dump-file contained what appeared to be valid and intact memory while the latter only contains low-memory BIOS-related data. Moreover, neither dump-file contained what appeared to be hardware-reserved computer memory (RAM set aside for hardware use) or virtual machine hardware device-specific memory (non-RAM hardware memory). Furthermore, memory from */proc/kcore* was in ELF format while memory from */dev/mem* was not. Finally, the memory dump-file captured from */dev/mem* was too small to be of any use.

Table 32. Experimental memory dump results for Ubuntu Linux 64-bit using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,390,782,976 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/proc/kcore
Time	205 seconds
Dump size	14,421,532,672 bytes / filled up disk partition
Number of 7, 8, 16 and 32-bit strings	5 / 64 / 0 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,204 / 6,980 / 0 / 0
Device	/dev/mem
Time	<1 second

Dump size	1,052,672 bytes
Number of 7, 8, 16 and 32-bit strings	1,985 / 16,541 / 12 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,049,359 / 1,052,648 / 1,030,690 / 0
Memory analysis commentary	The tool succeeded in acquiring memory from both */proc/kcore* and */dev/mem*. While the memory dump originating from */proc/kcore* managed to fill up the disk partition it contained almost nothing at all, only a few snippets of meaningless data. The dump-file resulting from */dev/mem* only contained what appeared to be low-memory BIOS-related data. Either way, both dumps contained too little data to be of any use. Also, memory from */proc/kcore* was in ELF format while memory from */dev/mem* was not.

C.1.6 Fedora Core Linux

Table 33. Experimental memory dump results for Fedora Core Linux 32-bit PAE using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,464,601,088 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/proc/kcore
Time	275 seconds
Dump size	11,296,238,592 bytes / filled up disk partition
Number of 7, 8, 16 and 32-bit strings	5 / 60 / 0 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,204 / 7076 / 0 / 0
Device	/dev/mem
Time	<1 second
Dump size	1,052,672 bytes
Number of 7, 8, 16 and 32-bit strings	3,897 / 26,130 / 17 / 1

Highest 7, 8, 16 and 32-bit string byte offset	1,052,613 / 1,052,667 / 1,030,690 / 425,124
Memory analysis commentary	The tool succeeded in acquiring memory from both */proc/kcore* and */dev/mem*. While the memory dump originating from */proc/kcore* managed to fill up the disk partition it contained almost nothing at all, only a few snippets of meaningless data. The dump-file resulting from */dev/mem* only contained what appeared to be low-memory BIOS-related data. Either way, both dumps contained too little data to be of any use. In addition, memory from */proc/kcore* was in ELF format while memory from */dev/mem* was not.

Table 34. Experimental memory dump results for Fedora Core Linux 64-bit using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,391,454,720 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/proc/kcore
Time	35 seconds
Dump size	9,130,041,344 bytes
Number of 7, 8, 16 and 32-bit strings	5 / 70 / 0 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,204 / 7,052 / 0 / 0
Device	/dev/mem
Time	<1 second
Dump size	1,052,672 bytes
Number of 7, 8, 16 and 32-bit strings	3,688 / 25,675 / 16 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,052,625 / 1,052,666 / 984,054 / 0
Memory analysis commentary	The tool succeeded in acquiring memory from both */proc/kcore* and */dev/mem*. While the memory dump originating from */proc/kcore* managed to fill up the disk partition it contained almost nothing at all, only a few snippets of meaningless data.

	The dump-file resulting from */dev/mem* only contained what appeared to be low-memory BIOS-related data. Either way, both dumps contained too little data to be of any use. In addition, memory from */proc/kcore* was in ELF format while memory from */dev/mem* was not.

C.1.7 Solaris

Table 35. Experimental memory dump results for Solaris 32-bit PAE using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,589,471,744 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	<1 second
Dump size	651,264 bytes
Number of 7, 8, 16 and 32-bit strings	1,732 / 9,627 / 9 / 3
Highest 7, 8, 16 and 32-bit string byte offset	523,764 / 523,764 / 514,782 / 422,792
Device	/dev/kmem
Time	0 second
Dump size	0 byte
Number of 7, 8, 16 and 32-bit strings	N/A - unable to acquire memory
Highest 7, 8, 16 and 32-bit string byte offset	N/A - unable to acquire memory
Device	/dev/allkmem
Time	0 second
Dump size	0 byte
Number of 7, 8, 16 and 32-bit strings	N/A - unable to acquire memory
Highest 7, 8, 16 and 32-bit	N/A - unable to acquire memory

string byte offset	
Memory analysis commentary	Memory acquisition from devices */dev/kmem* and */dev/allkmem* are met with disappointment as it was not possible to acquire any memory using this method. Only a very small amount of memory was acquired from device */dev/mem*, with a size of less than 1 MiB. This particular memory dump-file was found to contain too little information to be of any use.

Table 36. Experimental memory dump results for Solaris 64-bit using dd.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,589,471,744 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	<1 second
Dump size	651,264 bytes
Number of 7, 8, 16 and 32-bit strings	2,112 / 11,514 / 9 / 3
Highest 7, 8, 16 and 32-bit string byte offset	523,764 / 523,764 / 514,782 / 422,792
Device	/dev/kmem
Time	0 second
Dump size	0 byte
Number of 7, 8, 16 and 32-bit strings	N/A - unable to acquire memory
Highest 7, 8, 16 and 32-bit string byte offset	N/A - unable to acquire memory
Device	/dev/allkmem
Time	0 second
Dump size	0 byte
Number of 7, 8, 16 and 32-bit strings	N/A - unable to acquire memory

Highest 7, 8, 16 and 32-bit string byte offset	N/A - unable to acquire memory
Memory analysis commentary	Memory acquisition from devices */dev/kmem* and */dev/allkmem* was not possible. Only a very small amount of memory was acquired from device */dev/mem*, with a size of less than 1 MiB. This particular memory dump-file was found to contain too little information to be of any use.

C.2 Memdump

In this Annex, the experimental results obtained using the *Memdump* tool are examined. This tool was used against all of the various 32 and 64-bit UNIX operating systems examined herein.

C.2.1 FreeBSD

Table 37. Experimental memory dump results for FreeBSD 32-bit using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,744,890,880 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	67 seconds
Dump size	4,294,967,269 bytes
Number of 7, 8, 16 and 32-bit strings	6,450,625 / 42,385,742 / 27,169 / 99,625
Highest 7, 8, 16 and 32-bit string byte offset	4,294,967,285 / 4,294,967,284 / 4,294,902,774 /3,691,408,980
Memory analysis commentary	The tool succeeded not only in capturing all memory detected by the operating system but all computer memory up to the operating system's 4 GiB 32-bit operating system memory limit. The first 3,571 MiB RAM appears to be valid and intact while the remaining captured memory between 3,571 MiB and 4,096 MiB RAM appears to be hardware-reserved computer memory (RAM set aside for hardware use).

Table 38. Experimental memory dump results for FreeBSD 64-bit using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,576,004,096 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No

Generate Chain of Custody form	Yes
Device	/dev/mem
Time	615 seconds
Dump size	9,663,676,416 bytes / filled up disk partition
Number of 7, 8, 16 and 32-bit strings	1,471,903 / 16,055,249 / 3,473 / 7,612
Highest 7, 8, 16 and 32-bit string byte offset	9,126,745,798 / 9,126,805,474 / 8,524,989,958 / 8,524,411,200
Memory analysis commentary	The tool appeared to have correctly captured memory up to limit of virtual machine allocated memory (8 GiB). Although another 1 GiB of memory-related data was captured it appeared to be neither part of the system's actual memory (8 GiB), based on advanced pattern analysis. Moreover, this additional 1 GiB of data neither appeared to contain hardware-reserved computer memory (RAM set aside for hardware use) or virtual machine hardware device-specific memory (non-RAM hardware memory). As such, it can be inferred that this additional 1 GiB of data contained no computer memory whatsoever. The source and significance of the data contained therein is currently unknown.

C.2.2 NetBSD

Table 39. Experimental memory dump results for NetBSD 32-bit using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,757,633,536 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	109 seconds
Dump size	4,294,967,269 bytes
Number of 7, 8, 16 and 32-bit strings	129,793 / 5,014,443 / 372 / 225
Highest 7, 8, 16 and 32-bit string byte offset	4,294,967,285 / 4,294,967,284 / 4,294,902,774 / 3,757,709,128

Memory analysis commentary	The tool succeeded not only in capturing all memory detected by the operating system but all computer memory up to the operating system's 4 GiB 32-bit operating system memory limit. The first 3,583 MiB RAM appeared to be valid and intact while the remaining captured memory between 3,583 MiB and 4,096 MiB RAM appeared to be hardware-reserved computer memory (RAM set aside for hardware use).

Table 40. Experimental memory dump results for NetBSD 64-bit using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,589,471,744 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	310 seconds
Dump size	10,565,632,000 bytes / filled up disk partition
Number of 7, 8, 16 and 32-bit strings	261,577 / 20,207,062 / 340 / 252
Highest 7, 8, 16 and 32-bit string byte offset	9,126,778,437 / 9,126,778,437 / 9,118,537,174 / 9,085,660,816
Memory analysis commentary	The tool appeared to have correctly captured memory up to limit of virtual machine allocated memory (8 GiB). Although more than another 1 GiB of memory-related data was captured it appeared to be neither part of the system's actual memory (8 GiB), based on advanced pattern analysis. Moreover, this additional data neither appeared to contain hardware-reserved computer memory (RAM set aside for hardware use) or virtual machine hardware device-specific memory (non-RAM hardware memory). As such, it can be inferred that this additional data contained no computer memory whatsoever. The source and significance of the data contained therein is currently unknown.

C.2.3 OpenBSD

Table 41. Experimental memory dump results for OpenBSD 32-bit using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,756,982,272 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	101 seconds
Dump size	4,294,967,269 bytes
Number of 7, 8, 16 and 32-bit strings	139,074 / 1,250,813 / 666 / 269
Highest 7, 8, 16 and 32-bit string byte offset	4,294,967,285 / 4,294,967,284 / 4,294,902,774 / 3,756,925,372
Memory analysis commentary	The tool succeeded not only in capturing all memory detected by the operating system but all computer memory up to the operating system's 4 GiB memory limit. The first 3,583 MiB RAM appeared to be valid and intact while the remaining captured memory between 3,583 MiB and 4,096 MiB RAM appeared to be hardware-reserved computer memory (RAM set aside for hardware use).

Table 42. Experimental memory dump results for OpenBSD 64-bit using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,756,982,272 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	195 seconds
Dump size	4,294,967,269 bytes
Number of 7, 8, 16 and 32-bit	175,744 / 1,403,859 / 625 / 299

strings	
Highest 7, 8, 16 and 32-bit string byte offset	4,294,967,285 / 4,294,967,284 / 4,294,902,774 / 3,755,839,952
Memory analysis commentary	The tool succeeded not only in capturing all memory detected by the operating system but all computer memory up to the operating system's 4 GiB memory limit. The first 3,583 MiB RAM appeared to be valid and intact while the remaining captured memory between 3,583 MiB and 4,096 MiB RAM appeared to be hardware-reserved computer memory (RAM set aside for hardware use).

C.2.4 Red Hat Linux

Table 43. Experimental memory dump results for Red Hat Linux using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,451,578,880 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	4 seconds
Dump size	939,524,096 bytes
Number of 7, 8, 16 and 32-bit strings	5,282,949 / 12,907,323 / 4,585 / 11,623
Highest 7, 8, 16 and 32-bit string byte offset	939,484,154 / 939,516,268 / 917,787,880 / 939,434,132
Memory analysis commentary	Although the memory dump was only 896 MiB in size, it appeared to be an intact memory dump. Furthermore, although it was only a partial representation of the full amount of memory it may nevertheless be of some use. Although the operating system detected far more memory than 896 MiB, the operating system could not furnish more memory for acquisition for unknown reasons. The dump-file appeared to be valid and intact. Moreover, the dump-file appeared to contain neither hardware-reserved computer memory nor virtual machine hardware device-specific memory (non-RAM hardware memory).

C.2.5 Ubuntu Linux

Table 44. Experimental memory dump results for Ubuntu Linux 32-bit using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,699,986,432 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	2 second
Dump size	1,052,672 bytes
Number of 7, 8, 16 and 32-bit strings	2,042 / 17,154 / 11 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,052,521 / 1,052,668 / 984,054 / 0
Memory analysis commentary	The tool was entirely incapable of acquiring any memory over the first approximate 1 MiB RAM and as such the memory dump was highly incomplete and therefore of very limited use.

Table 45. Experimental memory dump results for Ubuntu Linux 32-bit PAE using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,463,405,056 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	2 second
Dump size	1,052,672 bytes
Number of 7, 8, 16 and 32-bit strings	2,035 / 17,136 / 11 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,052,663 / 1,052,659 / 984,054 / 0

Memory analysis commentary	The tool was entirely incapable of acquiring any memory over the first approximate 1 MiB RAM and as such the memory dump was highly incomplete and therefore of very limited use.

Table 46. Experimental memory dump results for Ubuntu Linux 64-bit using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,390,782,976 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	20 second
Dump size	537,989,120 bytes
Number of 7, 8, 16 and 32-bit strings	2,761 / 21,312 / 23 / 0
Highest 7, 8, 16 and 32-bit string byte offset	537,989,109 / 537,989,108 / 537,924,598 / 0
Memory analysis commentary	Although the memory dump is approximately 513 MiB in size, it was an incomplete memory dump. Nevertheless, the tool did succeed in capturing some memory above the first 1 MiB RAM. The exact nature of this data and its actual representation of computer memory cannot be determined at this time. Moreover, the number of data strings found in memory is unusually few considering the size of memory dump-file acquired. As such, it is unlikely that this memory dump-file will be of particular use to investigators.

C.2.6 Fedora Core Linux

Table 47. Experimental memory dump results for Fedora Core Linux 32-bit PAE using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,464,601,088 bytes
Tool performs hash verification	No
Must be run as root	Yes

Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	<1 second
Dump size	1,052,672 bytes
Number of 7, 8, 16 and 32-bit strings	3,884 / 26,642 / 17 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,055,220 / 1,055,442 / 1,033,340 / 0
Memory analysis commentary	The tool was entirely incapable of acquiring any memory over the first approximate 1 MiB RAM and as such the memory dump was highly incomplete and therefore of very limited use.

Table 48. Experimental memory dump results for Fedora Core Linux 64-bit using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,391,454,720 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	30 seconds
Dump size	537,989,120 bytes
Number of 7, 8, 16 and 32-bit strings	6,524 / 40,308 / 26 / 0
Highest 7, 8, 16 and 32-bit string byte offset	537,989,109 / 537,989,108 / 537,924,598 / 0
Memory analysis commentary	Although the memory dump is approximately 513 MiB in size, it was an incomplete memory dump. Nevertheless, the tool did succeed in capturing some memory above the first 1 MiB RAM. The exact nature of this data and its actual representation of computer memory cannot be determined at this time. Moreover, the number of data strings found in memory is unusually few considering the size of memory dump-file acquired. As such, it is unlikely that this memory dump-file will be of particular use to investigators.

C.2.7 Solaris

Table 49. Experimental memory dump results for Solaris 32-bit PAE using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,589,471,744 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	156 seconds
Dump size	3,757,635,536 bytes
Number of 7, 8, 16 and 32-bit strings	1,342,456 / 12,370,785 / 4,572 / 668
Highest 7, 8, 16 and 32-bit string byte offset	3,757,633,532 / 3,757,633,532 / 3,757,426,644 3,747,103,596
Memory analysis commentary	The tool appeared to have correctly captured the system's memory. Although the 32-bit *Memdump* program should be able to address up to 4 GiB RAM the remaining non-captured memory is most likely hardware-reserved computer memory (RAM set aside for hardware use) which the operating system does not make available to non-kernel subsystems. As such, this memory dump appears to have been successful thereby making it appropriate for use against Solaris-based systems.

Table 50. Experimental memory dump results for Solaris 64-bit using Memdump.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,589,471,744 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	258 seconds
Dump size	8,589,471,744 bytes

Number of 7, 8, 16 and 32-bit strings	4,329,837 / 36,611,894 / 209,107 / 10,708
Highest 7, 8, 16 and 32-bit string byte offset	8,587,330,404 / 8,587,374,536 / 8,586,702,680 / 8,587,236,572
Memory analysis commentary	The tool appeared to have fully acquired the operating system's detected amount of physical computer memory without any noticeable issue or specific error. This memory dump appears to have been successful thereby making it appropriate for use against Solaris-based systems.

C.3 Helix 3 Pro Release 3

In this Annex, the experimental results obtained using the Helix 3 Pro Release 3 are examined. This tool was used against all of the various 32 and 64-bit UNIX operating systems examined herein.

C.3.1 Ubuntu Linux

Table 51. Experimental memory dump results for Ubuntu Linux 32-bit using Helix 3 Pro Release 3.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,699,986,432 bytes
Tool performs hash verification	Yes – MD5 and SHA1 tested and verified
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	268 seconds
Dump size	3,699,986,432 bytes
Number of 7, 8, 16 and 32-bit strings	43,348,919 / 256,486,121 / 11 / 0
Highest 7, 8, 16 and 32-bit string byte offset	3,699,986,392 / 3,699,986,424 / 984,054 / 0
Memory analysis commentary	The tool appears to have successfully captured the computer system's memory (RAM). Additional analysis revealed that the computer memory dump-file was valid and intact. However, the tool was only able to acquire memory up to the operating system's memory detection limit. As such, it appeared as if the hardware-reserved computer memory (RAM set aside for hardware use) found between 3,528 MiB and 4,096 MiB could not be acquired and was determined to be filled in with binary zeros. It was determined that the tool consumed upwards of 300 MiB RAM with its dependencies which were largely loaded into system memory. This provides a suitable explanation for the large number of strings found in the memory dump-file.

Table 52. Experimental memory dump results for Ubuntu Linux 32-bit PAE using Helix 3 Pro Release 3.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,463,405,056 bytes
Tool performs hash verification	Yes – MD5 and SHA1 tested and verified
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	375 seconds
Dump size	8,463,405,056 bytes
Number of 7, 8, 16 and 32-bit strings	56,046,917 / 374,423,544 / 3,943 / 0
Highest 7, 8, 16 and 32-bit string byte offset	8,459,252,608 / 8,459,252,722 / 5,602,291,184 / 0
Memory analysis commentary	The tool appears to have successfully acquired the computer system's memory (RAM) up to the operating system's memory detection limit. The tool, which is 32-bit in nature, was operational in a PAE-based environment, unlike its partial functionality in 64-bit environments. In this experiment, based on advanced analyses, the computer memory dump-file appeared valid and intact. However, captured computer system memory found between 8,071 MiB and 8,192 MiB could not be acquired, memory which is generally reserved for hardware-reserved computer memory (RAM set aside for hardware use). Memory in the dump-file between these locations was instead filled with binary zeros. It was determined that the tool consumed upwards of 300 MiB RAM with its dependencies which were largely loaded into system memory. This provides a suitable explanation for the large number of strings found in the memory dump-file.

Table 53. Experimental memory dump results for Ubuntu Linux 64-bit using Helix 3 Pro Release 3.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,390,782,976 bytes
Tool performs hash verification	Yes – MD5 and SHA1 tested and verified

Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	345 seconds
Dump size	8,390,782,976 bytes
Number of 7, 8, 16 and 32-bit strings	47,386 / 37,802,021 / 12 / 0
Highest 7, 8, 16 and 32-bit string byte offset	4,297,740,584 / 4,297,740,584 / 1,030,690 / 0
Memory analysis commentary	The tool appears to have only partially succeeded in capturing the computer system's memory (RAM) even if the memory dump-file size was almost the same size as the operating system's detected memory limit. Additional analysis revealed that only a portion of the computer memory dump-file was in fact valid and intact. The tool was only able to acquire memory up to the operating system's 32-bit memory limitation imposed by the system because the acquisition tool was 32-bit in nature. Moreover, the difference in size between the memory dump-file and the amount of memory allocated to the virtual machine was approximately 120 MiB, likely representing at least, in part, the hardware-reserved computer memory (RAM set aside for hardware use). It was determined that the tool consumed upwards of 300 MiB RAM with its dependencies which were largely loaded into system memory. Furthermore, valid and intact memory appears to have stopped at around 4 GiB due to the aforementioned memory limitation. The rest of the dump-file has been filled with binary zeros. This, and the fact that the tool was likely to have, in part, been loaded with its dependencies into memory above 4 GiB by the VMM satisfactorily explains the reason why far less memory-based strings were detected in this specific memory dump. This is, of course, in comparison to experiments conducted on 32-bit operating systems.

C.3.2 Fedora Core Linux

Table 54. Experimental memory dump results for Fedora Core Linux 32-bit PAE using Helix 3 Pro Release 3.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,464,601,088 bytes
Tool performs hash verification	Yes – MD5 and SHA1 tested and verified
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	315 seconds
Dump size	8,464,601,088 bytes
Number of 7, 8, 16 and 32-bit strings	52,985,802 / 404,239,412 / 29,465 / 0
Highest 7, 8, 16 and 32-bit string byte offset	8,459,252,723 / 8,459,252,723 / 7,569,422,304 / 0
Memory analysis commentary	The tool appears to have successfully acquired the computer system's memory (RAM) up to the operating system's detection limit. The tool, which is 32-bit in nature, is in fact fully functional in a PAE environment, unlike the partial functionality displayed in 64-bit environments. In this experiment, based on advanced analyses, the memory dump-file appeared valid and intact. However, operating system memory found between 8,072 MiB and 8,192 MiB could not be acquired, memory which is generally reserved for hardware-reserved computer memory (RAM set aside for hardware use). It was determined that the tool consumed upwards of 300 MiB RAM with its dependencies which were largely loaded into system memory. This provides a suitable explanation for the large number of strings found in the memory dump-file.

Table 55. Experimental memory dump results for Fedora Core Linux 64-bit using Helix 3 Pro Release 3.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,391,454,720 bytes
Tool performs hash verification	Yes – MD5 and SHA1 tested and verified
Must be run as root	Yes
Kernel error when running tool	No
Generate Chain of Custody form	Yes
Device	/dev/mem
Time	427 seconds
Dump size	8,391,454,720 bytes
Number of 7, 8, 16 and 32-bit strings	4,242,417 / 37,812,464 / 16 / 0
Highest 7, 8, 16 and 32-bit string byte offset	4,297,740,584 / 4,297,740,584 / 984,054
Memory analysis commentary	The tool appears to have only partially captured the computer system's memory (RAM) even though the memory dump-file size was almost the same size as the amount of memory allocated to the virtual machine. Additional analysis revealed that only a portion of the computer memory dumped was in fact valid and intact. Moreover, the tool was only able to acquire memory up to the operating system's 32-bit memory detection limit imposed by the system due to the use of the 32-bit acquisition tool. Furthermore, the difference in size between the memory dump-file and the amount of memory allocated to the virtual machine was approximately 189 MiB, likely representing at least in part hardware-reserved computer memory (RAM set aside for hardware use). It was determined that the tool consumed upwards of 300 MiB RAM with its dependencies which were largely loaded into system memory. Valid and intact memory appears to have stopped at around 4 GiB due to the aforementioned memory limitation. The rest of the dump-file has been filled with binary zeros. This, and the fact that the tool was likely to have, in part, been loaded with its dependencies into memory above 4 GiB by the VMM satisfactorily explains why far less memory-based strings were detected in this memory dump. This is, of course, in comparison to experiments conducted on 32-bit operating systems.

C.4 Fmem

In this Annex, the experimental results obtained using the *Memdump* tool are examined. This tool was used against all of the various 32 and 64-bit UNIX operating systems examined herein.

C.4.1 Ubuntu Linux

Table 56. Experimental memory dump results for Ubuntu Linux 32-bit using Fmem.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,699,986,432 bytes
Tool performs hash verification	No
Must be run as root	Yes
Kernel error when running tool	No
Did module load?	Yes
Kernel error when running tool	No
Device	/dev/fmem
Time	155 seconds
Dump size	3,699,986,432 bytes
Number of 7, 8, 16 and 32-bit strings	2,517,949 / 14,270,563 / 5,236 / 4,514
Highest 7, 8, 16 and 32-bit string byte offset	3,699,916,674 / 3,699,933,180 / 3,699,613,578 / 3,699,797,748
Memory analysis commentary	The tool was able to acquire all detected operating system memory (RAM). However, memory above this limit up to the 4 GiB memory limit imposed by non-PAE systems, specifically 568 MiB RAM, could not be acquired. This non-acquired memory likely represented the system's hardware-reserved computer memory (RAM set aside for hardware use). Captured memory appeared valid and intact.

Table 57. Experimental memory dump results for Ubuntu Linux 32-bit PAE using Fmem.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,463,405,056 bytes
Tool performs hash verification?	No
Must be run as root?	Yes

Kernel error when running tool?	No
Did module load?	Yes
Kernel error when running tool?	No
Device	/dev/fmem
Time	126 seconds
Dump size	8,463,405,056 bytes
Number of 7, 8, 16 and 32-bit strings	795,316 / 30,502,599 / 2,180 / 445
Highest 7, 8, 16 and 32-bit string byte offset	8,053,438,596 / 8,329,908,356 / 8,187,187,762 / 8,052,637,032
Memory analysis commentary	The tool was able to fully acquire all detected operating system memory (RAM) and therefore fully supports 32-bit PAE-enabled memory. Interestingly, the difference between the size of the actual memory dump-file and the size of allocated memory to the virtual machine is approximately 121 MiB. This difference likely represented the system's hardware-reserved computer memory (RAM set aside for hardware use). Captured memory appeared valid and intact.

Table 58. Experimental memory dump results for Ubuntu Linux 64-bit using Fmem.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,390,782,976 bytes
Tool performs hash verification?	No
Must be run as root?	Yes
Kernel error when running tool?	No
Did module load?	Yes
Kernel error when running tool?	No
Device	/dev/fmem
Time	80 seconds
Dump size	8,390,782,976 bytes
Number of 7, 8, 16 and 32-bit strings	1,665,934 / 35,555,070 / 411 / 1,676
Highest 7, 8, 16 and 32-bit string byte offset	8,390,659,630 / 8,390,660,081 / 8,386,375,156 / 8,390,395,104
Memory analysis commentary	The tool was able to fully acquire all detected operating system memory (RAM) and therefore fully supports 64-bit memory.

	Interestingly, the difference between the size of the actual memory dump-file and the size of allocated memory to the virtual machine is approximately 190 MiB. This difference likely represented the system's hardware-reserved computer memory (RAM set aside for hardware use). Captured memory appeared valid and intact.

C.4.2 Fedora Core Linux

Table 59. Experimental memory dump results for Fedora Core Linux 32-bit PAE using Fmem.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,464,601,088 bytes
Tool performs hash verification?	No
Must be run as root?	Yes
Kernel error when running tool?	No
Did module load?	Yes
Kernel error when running tool?	No
Device	/dev/fmem
Time	117 seconds
Dump size	8,464,601,088 bytes
Number of 7, 8, 16 and 32-bit strings	1,499,431 / 33,177,767 / 301 / 1,142
Highest 7, 8, 16 and 32-bit string byte offset	8,187,218,104 / 8,329,908,356 / 8,0517,355,92 / 8,052,964,892
Memory analysis commentary	The tool was able to fully acquire all detected operating system memory (RAM) and therefore fully supports 32-bit PAE-enabled memory. Interestingly, the difference between the size of the actual memory dump-file and the size of allocated memory to the virtual machine is approximately 120 MiB. This difference likely represented the system's hardware-reserved computer memory (RAM set aside for hardware use). Captured memory appeared valid and intact.

Table 60. Experimental memory dump results for Fedora Core Linux 64-bit using Fmem.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,391,454,720 bytes
Tool performs hash verification?	No

Must be run as root?	Yes
Kernel error when running tool?	No
Did module load?	Yes
Kernel error when running tool?	No
Device	/dev/fmem
Time	101 seconds
Dump size	8,391,454,720 bytes
Number of 7, 8, 16 and 32-bit strings	566,027 / 9,511,239 / 2,012 / 690
Highest 7, 8, 16 and 32-bit string byte offset	8,391,452,368 / 8,391,454,616 / 8,391,452,314 / 8,387,829,696
Memory analysis commentary	The tool was able to fully acquire all detected operating system memory (RAM) and therefore fully supports 64-bit memory. Interestingly, the difference between the size of the actual memory dump-file and the size of allocated memory to the virtual machine is approximately 190 MiB. This difference likely represented the system's hardware-reserved computer memory (RAM set aside for hardware use). Captured memory appeared valid and intact.

C.5 Second Look

In this Annex, the experimental results obtained using the *Memdump* tool are examined. This tool was used against all of the various 32 and 64-bit UNIX operating systems examined herein.

C.5.1 Ubuntu Linux

Table 61. Experimental memory dump results for Ubuntu Linux 32-bit using Second Look.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,699,986,432 bytes
Tool performs hash verification?	No
Must be run as root?	Yes
Kernel error when running tool?	No
Device	/dev/pmad
Time	8 seconds
Dump size	3,758,030,848 bytes
Number of 7, 8, 16 and 32-bit strings	2,400,397 / 15,455,898 / 3,095 / 3,756
Highest 7, 8, 16 and 32-bit string byte offset	3,758,025,721 / 3,758,026,740 / 3,757,887,528 / 3,757,958,148
Memory analysis commentary	The tool was able to fully acquire all detected operating system memory (RAM) and then a bit more. The difference in memory between the amount of memory detected by the operating system and the dump-file's size is approximately 55 MiB, corresponding to only a portion of the available hardware-reserved computer memory (RAM set aside for hardware use). The remaining non-acquired memory, approximately 513 MiB RAM, corresponds to the remaining hardware-reserved computer memory.

Table 62. Experimental memory dump results for Ubuntu Linux 32-bit PAE using Second Look.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,463,405,056 bytes
Tool performs hash verification?	No
Must be run as root?	Yes
Kernel error when running tool?	No

Device	/dev/pmad
Time	20 seconds
Dump size	9,126,805,504 bytes
Number of 7, 8, 16 and 32-bit strings	2,533,892 / 27,042,485 / 3,133 / 3,798
Highest 7, 8, 16 and 32-bit string byte offset	9,126,805,118 / 9,126,805,491 / 9,126,500,992 / 9,125,425,508
Memory analysis commentary	The tool was able to fully acquire not only all of the operating system's detected memory (RAM), but all of its hardware-reserved computer memory (RAM set aside for hardware use) and what appears to be at least, in part, a portion of the system's hardware device-specific memory (non-RAM hardware memory). Captured memory was found to be intact and valid.

Table 63. Experimental memory dump results for Ubuntu Linux 64-bit using Second Look.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,390,782,976 bytes
Tool performs hash verification?	No
Must be run as root?	Yes
Kernel error when running tool?	No
Device	/dev/pmad
Time	21 seconds
Dump size	9,126,805,504 bytes
Number of 7, 8, 16 and 32-bit strings	2,588,760 / 34,529,493 / 2,884 / 3,330
Highest 7, 8, 16 and 32-bit string byte offset	9,126,787,968 / 9,126,801,589 / 9,126,436,752 / 9,126,424,576
Memory analysis commentary	The tool was able to fully acquire not only all of the operating system's detected memory (RAM), but all of its hardware-reserved computer memory (RAM set aside for hardware use) and what appears to be at least, in part, a portion of the system's hardware device-specific memory (non-RAM hardware memory). Captured memory was found to be intact and valid.

C.5.2 Fedora Core Linux

Table 64. Experimental memory dump results for Fedora Core Linux 32-bit PAE using Second Look.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,464,601,088 bytes
Tool performs hash verification?	No
Must be run as root?	Yes
Kernel error when running tool?	No
Device	/dev/pmad
Time	9,126,805,504 bytes
Dump size	62 seconds
Number of 7, 8, 16 and 32-bit strings	2,945,815 / 31,581,317 / 14,150 / 3,886
Highest 7, 8, 16 and 32-bit string byte offset	9,126,805,493 / 9,126,805,493 / 9,124,866,676 / 9,126,547,772
Memory analysis commentary	The tool was able to fully acquire not only all of the operating system's detected memory (RAM), but all of its hardware-reserved computer memory (RAM set aside for hardware use) and what appears to be at least, in part, a portion of the system's hardware device-specific memory (non-RAM hardware memory). Captured memory was found to be intact and valid.

Table 65. Experimental memory dump results for Fedora Core Linux 64-bit using Second Look.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,391,454,720 bytes
Tool performs hash verification?	No
Must be run as root?	Yes
Kernel error when running tool?	No
Device	/dev/pmad
Time	35 seconds
Dump size	9,126,805,504 bytes
Number of 7, 8, 16 and 32-bit strings	3,837,410 / 50,693,083 / 17,082 / 3,208
Highest 7, 8, 16 and 32-bit string byte offset	9,126,787,968 / 9,126,801,589 / 9,126,436,752 / 9,126,424,576

Memory analysis commentary	The tool was able to fully acquire not only all of the operating system's detected memory (RAM), but all of its hardware-reserved computer memory (RAM set aside for hardware use) and what appears to be at least, in part, a portion of the system's hardware device-specific memory (non-RAM hardware memory). Captured memory was found to be intact and valid.

C.6 X-Ways Capture

In this Annex, the experimental results obtained using the *Memdump* tool are examined. This tool was used against all of the various 32 and 64-bit UNIX operating systems examined herein.

C.6.1 Ubuntu Linux

Table 66. Experimental memory dump results for Ubuntu Linux 32-bit using X-Ways Capture.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	3,699,986,432 bytes
Tool performs hash verification?	Yes / Non-functional
Must be run as root?	Yes
Kernel error when running tool?	No
Device	/dev/mem
Time	20 seconds
Dump size	3,700,424,704 bytes
Number of 7, 8, 16 and 32-bit strings	1,991 / 16,757 / 12 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,048,565 / 1,048,564 / 1,030,690
Memory analysis commentary	Although the tool's dump-file size was slightly larger than the amount of detected operating system memory (RAM), it was found through analysis that the tool did not succeed in capturing memory beyond the first MiB RAM. The rest of the dump-file consisted solely of binary zeros. As such, this memory dump has too little data to be of any use.

Table 67. Experimental memory dump results for Ubuntu Linux 32-bit PAE using X-Ways Capture.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,463,405,056 bytes
Tool performs hash verification?	Yes / Non-functional
Must be run as root?	Yes
Kernel error when running tool?	No
Device	/dev/mem

Time	32 seconds
Dump size	8,464,105,472 bytes
Number of 7, 8, 16 and 32-bit strings	1,985 / 16,844 / 12 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,048,565 / 1,048,564 / 1,030,690 / 0
Memory analysis commentary	Although the tool's dump-file size was slightly larger than the amount of detected operating system memory (RAM), it was found through analysis that the tool did not succeed in capturing memory beyond the first MiB RAM. The rest of the dump-file consisted solely of binary zeros. As such, this memory dump has too little data to be of any use.

Table 68. Experimental memory dump results for Ubuntu Linux 64-bit using X-Ways Capture.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,390,782,976 bytes
Tool performs hash verification?	Yes / Non-functional
Must be run as root?	Yes
Kernel error when running tool?	No
Device	/dev/mem
Time	28 seconds
Dump size	8,391,753,728 bytes
Number of 7, 8, 16 and 32-bit strings	1,974 / 16,486 / 12 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,048,565 / 1,048,564 / 1,030,690 / 0
Memory analysis commentary	Although the tool's dump-file size was slightly larger than the amount of detected operating system memory (RAM), it was found through analysis that the tool did not succeed in capturing memory beyond the first MiB RAM. The rest of the dump-file consisted solely of binary zeros. As such, this memory dump has too little data to be of any use.

C.6.2 Fedora Core Linux

Table 69. Experimental memory dump results for Fedora Core Linux 32-bit PAE using X-Ways Capture.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,464,601,088 bytes
Tool performs hash verification?	Yes / Non-functional
Must be run as root?	Yes
Kernel error when running tool?	No
Device	/dev/mem
Time	31 seconds
Dump size	8,465,154,048 bytes
Number of 7, 8, 16 and 32-bit strings	3,845 / 26,347 / 17 / 0
Highest 7, 8, 16 and 32-bit string byte offset	1,048,565 / 1,048,564 / 1,030,690 / 0
Memory analysis commentary	Although the tool's dump-file size was slightly larger than the amount of detected operating system memory (RAM), it was found through analysis that the tool did not succeed in capturing memory beyond the first MiB RAM. The rest of the dump-file consisted solely of binary zeros. As such, this memory dump has too little data to be of any use.

Table 70. Experimental memory dump results for Fedora Core Linux 64-bit using X-Ways Capture.

Virtual machine allocated RAM	8,589,934,592 bytes
Detected memory	8,391,454,720 bytes
Tool performs hash verification?	Yes / Non-functional
Must be run as root?	Yes
Kernel error when running tool?	No
Device	/dev/mem
Time	45 seconds
Dump size	8,391,753,728 bytes
Number of 7, 8, 16 and 32-bit strings	3,689 / 25,616 / 17 / 0
Highest 7, 8, 16 and 32-bit	1,048,565 / 1,048,564 / 1,030,690 / 0

string byte offset	
Memory analysis commentary	Although the tool's dump-file size was slightly larger than the amount of detected operating system memory (RAM), it was found through analysis that the tool did not succeed in capturing memory beyond the first MiB RAM. The rest of the dump-file consisted solely of binary zeros. As such, this memory dump has too little data to be of any use.

Annex D Tool configuration files

D.1 X-Ways Capture Linux configuration file

The configuration file, *capture.ini*, used throughout this work for memory acquisition of Linux systems using the X-Ways Capture Linux utility, is as follows:

```
[steps]
AppendIni
#GetUserDate
#GetUserTime
#Ask "Please enter a comment"
#Ask Please enter a comment
#Ask Please enter three characters: ???
+DumpPhysicalMemory
+DumpProcessList
+AppendToLog /proc/partitions
+AppendToLog /proc/mounts
+AppendToLog /proc/modules
ATACheck
EncryptionCheckFile /proc/mounts
+EncryptionCheckFile /proc/modules
+EncryptionCheckProcessList
-PhysicalImaging
-LogicalBackup

[ListProcessesCommand]
ps -Aef

[SearchFileForEncryption]
/proc/mounts
/dev/loop

[SearchFileForEncryption]
/proc/modules
aes
anubis
arc4
blowfish
cast5
cast6
crc32c
crypto_null
deflate
des
khazad
```

```
michael_mic
serpent
sha256
sha512
tea
twofish
wp512
cryptoloop
dm_crypt

[SearchProcessListForEncryption]
loop

[settings]
language=English
#language=German
PromptForOutputPath
#UserShouldAcknowledge
DateFormat=dd/mm/yyyy
#DateFormat=dd.mm.yyyy
LogInfoMsgs
LogHints
LogWarnings
LogErrors
LogResults
PrintInfoMsgs
PrintHints
PrintWarnings
PrintErrors
PrintResults
ImageSegmentSize=2000
PhysicalImageFormat=raw
#PhysicalImageFormat=e01-compressed
#PhysicalImageFormat=e01-uncompressed
+PhysicalImageCalcHash=md5
+PhysicalImageCalcHash=sha-1
+PhysicalImageCalcHash=sha-256
#PhysicalImageCalcHash=none
FileSplitSize=2147483647

[ExcludeDevicesFromPhysicalImaging]
ram
md

[ExcludeFromLogicalBackup]
/dev
/sys
/proc
```

[LinuxExcludeFromLogicalBackupFS]
smbfs
nfs
coda
iso9660
udf

[user]
name=IAMNOTSURE
date=71
key=4RWR454324343F24

The changes made to this configuration file by the author include adding the *[user]* section, which provides license validation for the tool. In addition, the *[ListProcessesCommand]* section has been modified from *ps -A* to *ps -Aef*. In the *[settings]* section, the calculate hash settings for MD5, SHA-1 and SHA-256 have been uncommented and set to force using the "+" symbol in order for all these algorithms to be used for hashing the memory dump-file.

Additional changes made to the configuration file, specifically in the *[steps]* section. The following changes ensure the inclusion and execution of specific functionality which are as follows:

+DumpPhysicalMemory
+DumpProcessList
+AppendToLog /proc/partitions
+AppendToLog /proc/mounts
+AppendToLog /proc/modules
+EncryptionCheckFile /proc/modules
+EncryptionCheckProcessList

However, certain features in the *[steps]* section were configured to be ignored and include:

-PhysicalImaging
-LogicalBackup

D.2 Sample Linux /proc/meminfo

Below is an example of the Linux pseudo-file */proc/meminfo* structure from a Linux kernel 2.6.x system. Here, kB is equivalent to KiB.

MemTotal:	16466128 kB
MemFree:	15141644 kB
Buffers:	96004 kB
Cached:	667440 kB
SwapCached:	0 kB
Active:	386528 kB
Inactive:	504876 kB

```
Active(anon):       128920 kB
Inactive(anon):     126584 kB
Active(file):       257608 kB
Inactive(file):     378292 kB
Unevictable:        0 kB
Mlocked:            0 kB
SwapTotal:          0 kB
SwapFree:           0 kB
Dirty:              76 kB
Writeback:          0 kB
AnonPages:          128060 kB
Mapped:             52412 kB
Shmem:              127528 kB
Slab:               52820 kB
Sreclaimable:       29480 kB
Sunreclaim:         23340 kB
KernelStack:        2792 kB
PageTables:         13340 kB
NFS_Unstable:       0 kB
Bounce:             0 kB
WritebackTmp:       0 kB
CommitLimit:        8233064 kB
Committed_AS:       1005472 kB
VmallocTotal:       34359738367 kB
VmallocUsed:        362440 kB
VmallocChunk:       34359316476 kB
HardwareCorrupted:  0 kB
AnonHugePages:      20480 kB
HugePages_Total:    0
HugePages_Free:     0
HugePages_Rsvd:     0
HugePages_Surp:     0
Hugepagesize:       2048 kB
DirectMap4k:        75816 kB
DirectMap2M:        16699392 kB
```

D.3 Sample Linux */proc/iomem* with respect to Second Look memory acquisition script

The Linux kernel pseudo-file */proc/iomem* structure is used by the kernel to map the location and addresses of the various forms of memory provided to the system by the assortment of computer equipment found connected to it. For example, consider that a portion of video card memory is addressable by the operating system. This memory is found in the following example below labelled as *nvidia* (seen highlighted in red below). This video card includes 1 GiB DDR RAM although only a small fraction of this memory is actually available to the operating system.

Specifically, the operating system only sees 33,554,431 bytes of this video memory[28]. The rest is reserved by the video device for its own use, never to be directly seen or used by the operating system.

In this same way, physical computer memory consisting of DIMMs or SIMMs is similarly available to the operating system (seen highlighted in blue below). It is available at different offsets throughout the computer system's addressable I/O range. However, in adding up the various *System RAM* memory ranges (seen highlighted in cyan below) the example below it becomes apparent that this example system is equipped with 16 GiB RAM.

The Second Look memory acquisition script, *secondlook-memdump.sh*, works by reading the pseudo-file */proc/iomem* structure and based on the information obtained therein, feeds it to the *dd* command as corresponding byte offset parameters. In this manner, the script is certain to acquire all physical computer memory.

Below is an example pseudo-file */proc/iomem* structure from a Linux kernel 2.6.x system. All memory offsets are in hexadecimal.

```
00000000-0000ffff : reserved
00010000-0009afff : System RAM
0009b000-0009ffff : RAM buffer
000a0000-000bffff : PCI Bus 0000:00
000c0000-000effff : PCI Bus 0000:00
000f0000-000fffff : PCI Bus 0000:00
  000f0000-000fffff : reserved
00100000-cfe0abff : System RAM
  01000000-0147dfad : Kernel code
  0147dfae-01b412ff : Kernel data
  01c34000-01daffff : Kernel bss
cfe0ac00-cfe0cbff : ACPI Non-volatile Storage
cfe0cc00-cfe0ebff : RAM buffer
cfe0ec00-cfe5cbff : reserved
cfe5cc00-cfe5ebff : ACPI Tables
cfe5ec00-dfffffff : reserved
  cff00000-dfffffff : PCI Bus 0000:00
    db5fb800-db5fbfff : 0000:0c:0a.0
      db5fb800-db5fbfff : FireWire_ohci
    db5fc000-db5fffff : 0000:0c:0a.0
    db600000-db6fffff : PCI Bus 0000:0b
      db6f0000-db6fffff : 0000:0b:00.0
        db6f0000-db6fffff : tg3
    db700000-db7fffff : PCI Bus 0000:06
    db800000-db8fffff : PCI Bus 0000:0c
    db900000-dbffffff : PCI Bus 0000:01
      db900000-dbafffff : PCI Bus 0000:05
```

28 Memory address dc000000 to ddffffff (3,690,987,520 – 3,724,541,951) yields 33,554,431 or 1FFFFFF addressable bytes of video memory.

```
        db9eb000-db9ebfff : 0000:05:05.0
          db9eb000-db9ebfff : aic7xxx
        db9ec000-db9effff : 0000:05:0b.0
          db9ec000-db9effff : mpt
        db9f0000-db9fffff : 0000:05:0b.0
          db9f0000-db9fffff : mpt
        dba00000-dba1ffff : 0000:05:05.0
      dbb00000-dbefffff : PCI Bus 0000:02
        dbb00000-dbcfffff : PCI Bus 0000:04
          dbbfbf80-dbbfbfff : 0000:04:00.0
          dbbfbf80-dbbfbfff : sata_sil24
          dbbfc000-dbbfffff : 0000:04:00.0
          dbbfc000-dbbfffff : sata_sil24
          dbc00000-dbc7ffff : 0000:04:00.0
        dbd00000-dbefffff : PCI Bus 0000:03
          dbdff800-dbdfffff : 0000:03:00.0
          dbdff800-dbdfffff : ahci
          dbe00000-dbe0ffff : 0000:03:00.0
    dc000000-dfefffff : PCI Bus 0000:07
      dc000000-ddffffff : 0000:07:00.0
        dc000000-ddffffff : nvidia
      dfdfc000-dfdfffff : 0000:07:00.1
        dfdfc000-dfdfffff : ICH HD audio
      dfe00000-dfe7ffff : 0000:07:00.0
    dfffc000-dfffffff : 0000:00:1b.0
      dfffc000-dfffffff : ICH HD audio
e0000000-efffffff : PCI MMCONFIG 0000 [bus 00-ff]
  e0000000-efffffff : reserved
f0000000-feffffff : reserved
  f0000000-fe000000 : PCI Bus 0000:00
    f0000000-fbffffff : PCI Bus 0000:07
      f0000000-f7ffffff : 0000:07:00.0
      f8000000-fbffffff : 0000:07:00.0
    fc000000-fc0fffff : PCI Bus 0000:01
      fc000000-fc0fffff : PCI Bus 0000:05
        fc000000-fc0fffff : 0000:05:0b.0
    fc100000-fc1003ff : 0000:00:1f.2
      fc100000-fc1003ff : ahci
  fec00000-fec003ff : IOAPIC 0
  fec80000-fec803ff : IOAPIC 1
  fed00000-fed003ff : HPET 0
  fed20000-fed9ffff : PCI Bus 0000:00
  fee00000-fee00fff : Local APIC
ff97c000-ff97ffff : PCI Bus 0000:00
ff980800-ff980bff : PCI Bus 0000:00
  ff980800-ff980bff : 0000:00:1d.7
    ff980800-ff980bff : ehci_hcd
ffb00000-ffffffff : reserved
```

100000000-42fffffff : System RAM

D.4 Helix 3 Pro library dependencies for running atop 64-bit Linux systems

The following two sub-annexes detail the specific software dependencies required to allow both the underlying 64-bit Linux systems run 32-bit binaries and satisfy various software dependencies required for running Helix 3 Pro Release 3.

D.4.1 Library dependencies for newly installed Fedora Core 15 64-bit

Under a newly installed default Fedora Core 15 64-bit Linux installation, the following dependencies are required to allow both the execution of 32-bit binaries and satisfy the various required dependencies for running Helix 3 Pro Release 3:

glibc
nss-softokn-freebl
gtk2
atk
audit-libs
avahi-libs
cairo
cups-libs
dbus-libs
expat
fontconfig
freetype
gamin
gdk-pixbuf2
glib2
gnutls
jasper-libs
keyutils-libs
krb5-libs
libX11
libXau
libXcomposite
libXcursor
libXdamage
libXext
libXfixes
libXft
libXinerama
libXrandr
libXrender
libcom_err
libgcc

libgcrypt
libgpg-error
libjpeg-turbo
libpng
libselinux
libstdc++
libtasn1
libthai
libtiff
libxcb
pango
pixman
zlib

D.4.2 Library dependencies for newly installed Ubuntu 11.04 64-bit

Under a newly installed default Ubuntu 11.04 64-bit Linux installation, the following dependencies are required to allow both the execution of 32-bit binaries and satisfy the various required dependencies for running Helix 3 Pro Release 3:

ia32-libs
lib32asound2
lib32bz2-1.0
lib32gcc1
lib32ncurses5
lib32ncursesw5
libstdc++6
lib32vl-0
lib32z1
libc6-i386

Bibliography

Open Source University, Red Hat Academy. Device Memory Buffers and /proc/iomem – Red Hat Academy 2.0. Informational web site. Red Hat Enterprise Linux 4 training course. Red Hat Inc. https://osu.redhat.com/content/courses/rha130-4/section_0002/tag_lessons/section_0002/section_0001/tag_resource/section_0003?set_language=en.

Oracle Corporation. Man pages section 7: Device and Network Interfaces. Reference manual. Part No.: 819-2254-33. Oracle Corporation. 2010. http://download.oracle.com/docs/cd/E19082-01/819-2254/book-info/index.html.

Oracle Corporation. Oracle Solaris Tunable Parameters Reference Manual. Reference manual. Part No.: 817-0404-19. Oracle Corporation. 2010. http://download.oracle.com/docs/cd/E19253-01/817-0404/book-info/index.html.

Oracle Corporation. Oracle VM VirtualBox User Manual. Guide. Version 4.1.0. Oracle Corporation. 2011. http://download.virtualbox.org/virtualbox/UserManual.pdf.

Oracle Corporation. Oracle Solaris 11 Express Frequently Asked Questions. FAQ. Oracle Corporation. November 2010. http://www.oracle.com/technetwork/server-storage/solaris11/overview/faqs-oraclesolaris11express-185609.pdf.

PikeWorks Corp. Second Look: Linux Memory Forensics User Guide. Product documentation. Version 2011.05. May 2011. PikeWorks Corp.

Venema, Wietse. Memdump Source Code. Source code. The Coroner's Toolkit. January 2004. http://www.porcupine.org/forensics/tct.html.

Venema, Wietse. Memdump(1) Man Page. Linux/UNIX man page. The Coroner's Toolkit. January 2004. http://www.porcupine.org/forensics/tct.html.

Weise, Joel, and Powell, Brad. Using Computer Forensics When Investigating System Attacks. Sun Blue Print (white paper). Sun Microsystems. Document No.: 819-2262. April 2005.

Wikipedia. 386BSD. Online encyclopaedic article. Wikimedia Foundation Inc. July 2011. http://en.wikipedia.org/wiki/386BSD.

Wikipedia. 64-bit. Online encyclopaedic article. Wikimedia Foundation Inc. July 2011. http://en.wikipedia.org/wiki/64-bit.

Wikipedia. Berkeley Software Distribution. Online encyclopaedic article. Wikimedia Foundation Inc. July 2011. http://en.wikipedia.org/wiki/Berkeley_Software_Distribution.

Wikipedia. BSD/OS. Online encyclopaedic article. Wikimedia Foundation Inc. May 2011. http://en.wikipedia.org/wiki/BSD/OS.

Wikipedia. Comparison of BSD operating systems. Online encyclopaedic article. Wikimedia Foundation Inc. July 2011. http://en.wikipedia.org/wiki/Comparison_of_BSD_operating_systems.

Wikipedia. Dd (Unix). Online encyclopaedic article. Wikimedia Foundation Inc. August 2011. http://en.wikipedia.org/wiki/Dd_(Unix).

Wikipedia. Fedora (operating system). Online encyclopaedic article. Wikimedia Foundation Inc. August 2011. http://en.wikipedia.org/wiki/Fedora_(operating_system).

Wikipedia. FreeBSD. Online encyclopaedic article. Wikimedia Foundation Inc. August 2011. http://en.wikipedia.org/wiki/FreeBSD.

Wikipedia. Linux. Online encyclopaedic article. Wikimedia Foundation Inc. August 2011. http://en.wikipedia.org/wiki/Linux.

Wikipedia. NetBSD. Online encyclopaedic article. Wikimedia Foundation Inc. July 2011. http://en.wikipedia.org/wiki/Netbsd.

Wikipedia. OpenBSD. Online encyclopaedic article. Wikimedia Foundation Inc. July 2011. http://en.wikipedia.org/wiki/Openbsd.

Wikipedia. Oracle Corporation. Online encyclopaedic article. Wikimedia Foundation Inc. August 2011. http://en.wikipedia.org/wiki/Oracle_Corporation.

Wikipedia. Physical Address Extension. Online encyclopaedic article. Wikimedia Foundation Inc. July 2011. http://en.wikipedia.org/wiki/Physical_Address_Extension.

Wikipedia. Red Hat Linux. Online encyclopaedic article. Wikimedia Foundation Inc. June 2011. http://en.wikipedia.org/wiki/Red_Hat_Linux.

Wikipedia. Solaris (operating system). Online encyclopaedic article. Wikimedia Foundation Inc. August 2011. http://en.wikipedia.org/wiki/Solaris_(operating_system).

Wikipedia. SunOS. Online encyclopaedic article. Wikimedia Foundation Inc. May 2011. http://en.wikipedia.org/wiki/SunOS.

Wikipedia. Ubuntu (operating system). Online encyclopaedic article. Wikimedia Foundation Inc. August 2011. http://en.wikipedia.org/wiki/Ubuntu_(operating_system).

Wikipedia. Unix System Laboratories. Online encyclopaedic article. Wikimedia Foundation Inc. June 2011. http://en.wikipedia.org/wiki/Unix_System_Laboratories.

Wikipedia. UNIX System V. Online encyclopaedic article. Wikimedia Foundation Inc. April 2011. http://en.wikipedia.org/wiki/UNIX_System_V.

Wikipedia. X86_64. Online encyclopaedic article. Wikimedia Foundation Inc. August 2011. http://en.wikipedia.org/wiki/X86_64.

Wilder, David. LKCD Installation and Configuration. Tutorial. IBM Inc. 2002. http://lkcd.sourceforge.net/doc/lkcd_tutorial.pdf.

X-Ways Software Technology AG. X-Ways Capture. Product documentation. Version 1.2. Unknown date. X-Ways Software Technology AG.

List of symbols/abbreviations/acronyms/initialisms

2D	Two Dimensional
3D	Three Dimensional
ACPI	Advanced Configuration and Power Interface
ASCII	American Standard Code for Information Interchange
ATI	Array Technologies Inc.
BIOS	Basic Input/Output System
BSD (386BSD / OpenBSD / NetBSD / FreeBSD)	Berkeley Software Distribution
CD	Compact Disc
CLI	Command Line Interface
DDR	Double Data Rate
DIMM	Dual Inline Memory Module
DND	Department of National Defence
DNS	Domain Name System
DOS	Disk Operating System
DRDC	Defence R&D Canada
DVD	Digital Video Disc / Digital Versatile Disc
E-SATA	External-Serial Advanced Technology Attachment
EAL	Evaluation Assurance Level
EHCI	Enhanced Host Control Interface
ELF	Executable and Linkable Format
EST	Eastern Standard Time
EBCDIC	Extended Binary Coded Decimal Interchange Code
exFAT	Extended File Allocation Table
FOSS	Free and Open Source Software
FTP	File Transfer Protocol
GCC	GNU C Compiler
GB	Gigabyte
GDB	GNU Debugger

GiB	Gibibyte (2^{30} bytes)
GHz	Gigahertz
GIMP	GNU Image Manipulation Program
GNOME	GNU Network Object Model Environment
GPL	GNU Public License
GRUB	Grand Unified Bootloader
GTK	Gimp ToolKit
GUI	Graphical User Interface
HD	High Definition
I/O	Input/Output
IT	Information Technology
KiB	Kibibyte (2^{10} bytes)
LCD	Liquid Crystal Display
LKM	Loadable Kernel Module
Mac OS X	Mac Operating System X
MB	Megabyte (10^{6} bytes)
MD5	Message Digest 5
MiB	Mebibyte (2^{20} bytes)
MP (e.g. GENERIC.MP)	Multi-Processor
N/A	Not Available
NFS	Network FileSystem
NT	New Technology
PAE	Physical Address Extension
PCI	Peripheral Component Interconnect
PowerPC	Performance Optimization With Enhanced RISC – Performance Computing
R2	Release 2
RAID	Redundant Array of Inexpensive Disks
RAM	Random Access Memory
RO	Read-Only
RPM	Red Hat Package Manager
RW	Read/Write
SATA	Serial Advanced Technology Attachment

SHA-1/256/512	Secure Hash Algorithm-1/256/512
SIMM	Single Inline Memory Module
SMP	Symmetric Multi-Processing
SPARC	Scalable Processor ARChitecture
SSD	Solid State Drive
SunOS	Sun Operating System
TB	Terabyte (10^{12} bytes)
UltraSPARC	Ultra Scalable Processor ARChitecture
USB	Universal Serial Bus
UTC	Coordinated Universal Time
VM	Virtual Machine
VMM	Virtual Memory Manager
x86	Intel compatible 8x86-based processor
x86_64	Intel compatible 8x86-based 64-bit processor

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified.)

1. ORIGINATOR (Name and address of the organization preparing the document.)

Defence R&D Canada – Valcartier
2459 Pie-XI Blvd North
Quebec (Quebec)
G3J 1X5 Canada

2. SECURITY CLASSIFICATION
(Overall security classification of the document, including special warning terms if applicable.)

UNCLASSIFIED
(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC JUNE 2010

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.)

(U) State of the art concerning memory acquisition software: A detailed examination of Linux, BSD and Solaris live memory acquisition

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.)

March 2012

6. NO. OF PAGES (Including Annexes, Appendices and DCD sheet.)

130

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.)

Technical Memorandum

8a. 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)

31XF20 « MOU RCMP "Live Forensics" »

8b. CONTRACT NO. (If appropriate, the applicable number under which the document was written)

9a. ORIGINATOR'S DOCUMENT NUMBER (Official document number by which the document is identified by the originating activity. Number must be unique to this document.)

TM 2012-008

9b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned to this document either by the originator or the sponsor.)

10. DOCUMENT AVAILABILITY (Any limitation on further distribution of the document, other than those imposed by security classification.)

(X) Unlimited distribution
( ) Distribution limited to defence departments
( ) Distribution limited to defence contractors
( ) Distribution limited to government
( ) Distribution limited to Defence R&D Canada
( ) Controlled by Source

11. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (10). However, where further distribution (beyond the audience specified in (10) is possible, a wider announcement audience may be selected.)

Unlimited

12. ABSTRACT (Brief and factual summary of the document. 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), or (U). May be in English only).

(U) This technical memorandum examines various software tools that can be used for carrying out forensic memory acquisition against various Linux, BSD, and Solaris x86-based systems. No comparable work could be found in the publicly available literature after an exhaustive survey of the subject matter. This current study is important as these UNIX systems are pervasive in today's modern world and are found in a variety of IT environments ranging from the home to corporate data centres. By addressing the pertinence of x86-based UNIX system memory acquisition the computer forensic investigator will be empowered with the necessary knowledge and techniques required to readily tap into this important avenue of potentially useful evidence. Two tools stand out above the rest, Second Look and Fmem, both of which succeeded in all experiments at capturing the underlying system's memory. Although some of the other tools examined herein had specific strengths, they did not work as expected in all instances.

(U) Ce mémorandum technique examine divers outils logiciels qui peuvent être utilisés pour l'acquisition inforensique de la mémoire de divers systèmes Linux, BSD et Solaris basés sur x86. Aucun travail comparable n'a pu être trouvé dans la littérature publique après une enquête exhaustive sur le sujet. Cette étude est pertinents car ces systèmes UNIX sont omniprésents dans notre monde moderne et se retrouvent dans une variété d'environnements informatiques allant de la maison aux centres de données d'entreprise. En abordant l'importance de l'acquisition de la mémoire des systèmes UNIX x86, l'enquêteur inforensique sera doté des connaissances et des techniques nécessaires pour puiser plus facilement dans cette importante source de preuves potentiellement utiles. Deux outils se distinguent, Second Look et Fmem, qui ont chacun réussi à capter la mémoire du système sous-jacent dans toutes les expérimentations. Bien que certains des autres outils examinés aient certains autres atouts spécifiques, ils n'ont pas fonctionné comme prévu dans tous les cas.

13. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases characterizing a document and could be helpful in cataloguing it. Should be **Unclassified** text. If not , the classification of each term should be indicated as with the title. Equipment model designation, trade name, military project code name, and geographic location may be included. If possible, should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified.)

Digital Forensics
Computer Forensics
UNIX
BSD
Linux
FreeBSD
NetBSD
OpenBSD
Memory acquisition
Memdump
DD
Second Look
Fmem
Helix 3 Pro
X-Ways Capture

Defence R&D Canada

Canada's Leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca