A Forensic Memory Image Acquisition Protocol Based on Windows Memory Analysis

José R. de la Cruz Echeandía
Computer Engineering
Jeffrey Duffany, Ph.D.
Electrical and Computer Engineering and Computer Science Department
Polytechnic University of Puerto Rico

Abstract *— Computer Forensics has become an extremely important evidence gathering and analysis field in the modern electronic driven world. Most of the evidence acquired, preserved, processed and analyzed originates from long term storage media. The importance of obtaining a forensic memory image has grown in importance in order to support the evidence analysis and obtain correct and irrefutable results. This project has developed a memory acquisition protocol that provides forensic examiners with the necessary tools to complete a comprehensive investigation. The protocol developed, which is targeted at the acquisition step of the evidence collection process, is based on memory analysis. Including memory as a data source empowers the analyst with context information that can be used to enhance the analysis of evidence extracted from long term storage media.*

Key Terms *— Computer Forensics, Computer Forensics Protocols, Digital Forensics, Memory Analysis.*

INTRODUCTION

Computer Forensics is defined by the National Information Assurance glossary as "*the practice of gathering, retaining, and analyzing computer-related data for investigative purposes in a manner that maintains the integrity of the data*." [1] Since long term data in computers is primarily stored in magnetic disks, most forensic investigations are based on such data. The acquisition of disk images is now mature, and the protocols used are widely accepted.

The Association of Chief Police Officers (ACPO) has developed a document that describes Good Practices for digital forensic investigations [2]. This document details the fundamental guidelines for the digital evidence collection process. The procedures detailed in the ACPO's report have been generally accepted by the international law enforcement community. Summarizing the ACPO's document, the Computer Forensics process can be described as consisting of six major steps:

1. Analyze the scene
2. Acquire the evidence
3. Preserve the evidence
4. Analyze the evidence
5. Report the findings

One of the principal motivations for this project stems from a paper on Windows memory analysis written by Jesse D. Kornblum [3]. It describes a technique to reconstruct the state of a computer from information carved out of a memory image. Although Kornblum's approach lies in the analysis phase of the forensic process, it provided much needed insight into the memory scenario. This project is based on the second phase of the process: evidence acquisition. In order to develop a correct procedure to secure a memory image it was important to understand how memory is used by the operating system.

BACKGROUND

Computers follow the Von Neumann model [4] in which all data and instructions reach the CPU (Central Processing Unit) via busses that connect the peripherals with the system's physical memory. This model is the current standard, and is depicted in Figure 1. As a result, computer forensic investigations have recently shifted to include memory acquisition as part of the data acquisition protocol. The main difficulty hindering the acceptance of data attained from memory has been the inability to attest the *integrity of the data* that has been acquired from such a volatile medium.

Because a computer's memory is subject to a high rate of change, acquiring a forensic image has proven to be a difficult task. This paper aims to offer insight into this rapidly developing research space by contributing a protocol based on memory analysis.

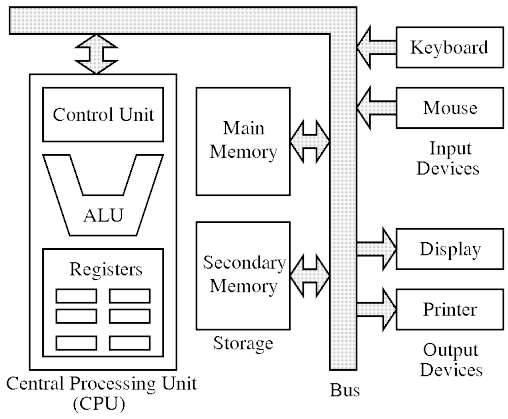

Figure 1
The Von Neumann Model

MEMORY ANALYSIS

A digital computer performs work when the CPU executes instructions. These instructions are provided by the software applications invoked by users to complete some task.

Software is developed using a programming language, such as C or Java. The "plain English" commands provided by the programmer are then converted into machine readable instructions by virtue of the compilation process. Once the executable code is invoked by the computer, it is loaded into the system's memory by the operating system. When a program is summoned to execute it becomes a process. This process is handled by the operating system, along with all other system requirements, who acts as the doorman to the CPU.

When a process is invoked, the operating system allocates sufficient memory space so as to ensure that the process can execute all of the instructions. The operating system then assigns a priority to the process, and inserts it in the appropriate queue. One of the main responsibilities of the operating system is to ensure that the memory space allocated to store the program's data is kept concurrent and separate from other processes. The memory problem starts at this instance.

Since modern programs can require a large memory space to hold all data and instructions, the operating system frequently "swaps" process blocks in and out of memory in order to service all processes which are waiting for a turn at the CPU. The "swapped" processes are store in a reserved area of the system's main storage device, usually a hard disk drive. This situation promotes a high turnover of memory space allocation. It is very possible for one process that takes a long time (more than 10 seconds for instance) to execute to be moved in and out of memory many times.

Since modern CPU's operate at clock rates in excess of 2 Gigahertz the rate of change in the memory is very high. A Hertz is defined as the amount of cycles that can be achieved in one second, and a cycle in this context refers to the time in seconds it takes for the clock to switch between high and low. In a typical Windows machine, with more than 40 processes in queue for execution, a memory location can be reused more than 10 times per second.

Therefore, the memory rate of change becomes a concern when performing memory procurement because when the acquisition process ends the data recovered might be inconsistent. Notwithstanding, the research performed shows that data collected from memory images is sound and can be used to provide extra context to the evidence gathered from long term storage devices.

Windows Memory

As stated by Solomon and Russinovich in Windows Internals [5], Windows divides the memory in two parts: user space and system space. The user space is where the processes invoked by some software application are stored while they await execution. The system space provides temporary storage for operating system processes and can be described as more stable. The system area stores the operating system's kernel, the process page tables, and the system cache among

others. The data and processes stored in the system area depend largely on the requirements of the users-level processes in queue for processing. Although Russinovich, et al, contend that some of the system level regions are fixed, this largely depends on the size of the memory itself. When the memory is large enough to hold all required system level processes the statement holds. But recent, and continuous, security threats to operating systems that target such fixed memory positions have given rise to random boot time memory space allocation in order to prevent adversaries to guess where these extremely important system services are to be located in memory.

In 32 bit systems, where addresses are formed using 8 hexadecimal characters, Windows can address $2^{32} - 1$ address locations, which roughly translates to 4 Gigabytes (exactly 4,294,967,295 bytes). Hence the operating system should be able to secure 2 Gigabytes of memory space for the system region, while the user-level address space is left with the remaining 2 Gigabytes for its processes. Dealing with 2 Gigabytes of space is no small feat; hence most operating systems have further logically divided the memory into a construct known as pages. A page is generally 4 Kilobytes in size (exactly 4,096 bytes). This division of memory allows the operating system to reference process areas by page number. Therefore if a process requires 5 Kilobytes of memory space, 2 pages must be allocated. This last scenario describes why pages in memory are mostly "blank". But the same scenario presents an opportunity to the forensic investigator by being able to inspect data that was left behind from other processes.

When a process is "swapped" to long term storage the operating system does not "zero-out" the newly available memory real estate. The operating system transfers the data required to perform the next process, "zeroing" out any excess area not used on a per page basis. Therefore, and with much frequency, some tables are unallocated but contain data pertinent to the "swapped" process. This scenario provides some of the context mentioned at the beginning of this report. Because the swap area is located in long term storage, evidence stored in it can be obtained using ordinary forensic techniques. Once the memory is collected, these unallocated pages provide the forensic analyst with supporting data which leads to a sounder forensic analysis report.

To summarize this section, Windows allocates memory area to user-level applications by assigning enough space to hold all instructions and data in units of memory pages and not single byte units. Since pages are the logical memory unit in use, the operating system is required to hold an index that contains the page number where a process has been allocated. This information is not usually collected or required when analyzing hard disk drives, but becomes an essential piece of information when dealing with memory.

The Subject Computer

The subject computer is a Dell Latitude D800 laptop computer. It "runs" on Microsoft Windows XP Professional, Service Pack 3. The CPU is an Intel Pentium M processor with a published clock rate of 1.5 Gigahertz. . It has a "C" partition that stores all program and system data with a 37.2 Gigabyte (40,007,729,152 bytes) capacity formatted using an NTFS (New Technology File System) file system, as shown in Figure 2

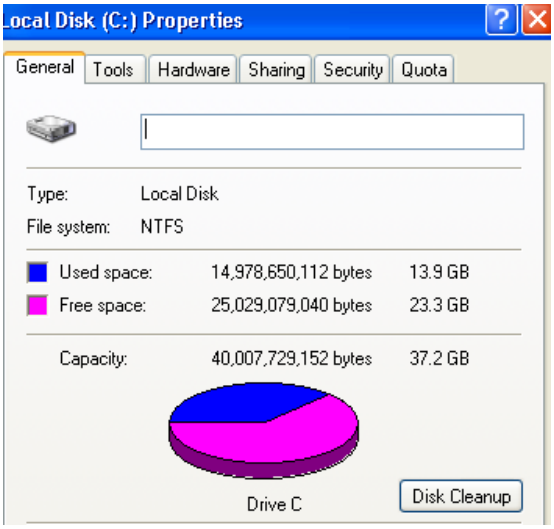

Figure 2
Target System Disk Capacity

Although considered to be an "old" system, this computer was specifically selected for its small memory size. Although this limitation can cause an increase in the memory rate of change, the images acquired are smaller in size and promote better analysis. The details of the CPU are shown in Figure 3, which was produced using the CPU-Z utility developed by CPUID.

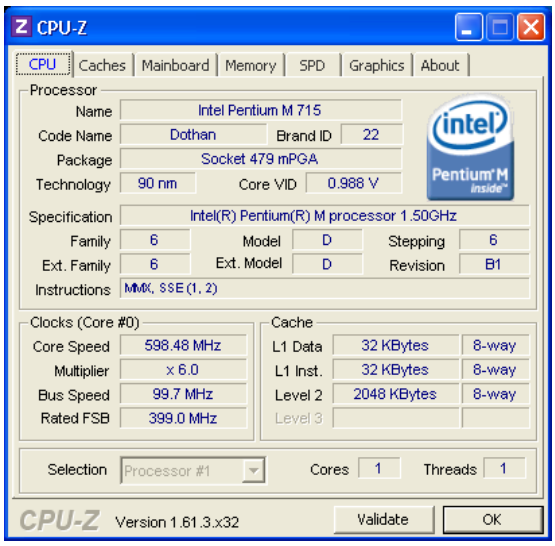

Figure 3
CPU Details

The system features a 512 MB physical RAM (Random Access Memory) memory capacity (actually 536,535,040 bytes), which is ideal for the research being conducted. Since most of the analysis was performed by reading the data captured from memory, the process was slow and tedious. Sifting through millions of data-bytes was enabled because of the system's small memory size.

ACQUISITION AND ANALYSIS TOOLS

In order to "disturb" the target system as little as possible, the acquisition and analysis tools selected were evaluated by two criteria: small footprint and the capability to be launched from a USB thumb drive. Some of the evaluated tools can be described as software systems, while other as mere utilities.

The preferred memory-image acquisition tool was DumpIT, developed by Matthieu Suiche and MoonSols [6]. This tool is extremely lightweight at only 203 KB, and is capable of generating complete physical memory "dumps" for 32 and 64 bit computer systems. Other acquisition tools considered were: Access Data's FTK Imager Lite [7], and Mandiant's Memoryze [8] and Redline [9]. Although the aforementioned tools are excellent forensic utensils, they tend to depend on the manufacturer's forensic "suite", hence their captures proved to be geared towards each system's underlying functionality. DumpIT, on the other hand, is simply a memory acquisition tool with no analysis system requirements or dependencies.

Once the physical memory was acquired, other tools were used to provide even more supporting data about the system at the time of inspection. The Sysinternals Suite, developed by Mark Russinovich and Bryce Cogswell [10], provided some very useful utilities. The two selected were Process Explorer [11] and TCP View [12]. Process Explorer was launched immediately after the memory had been captured in order to leverage its "save" utility. This feature allows the investigator to view the active processes when the utility was used. The results are saved into a text file. TCP View, as the name suggests, captures all TCP (Transfer Control Protocol) network connections at the time when the tool is invoked. The results are saved into a text file for further review. All of the utilities used during the acquisition process were launched from the command prompt of the subject system.

Once the acquisition process is complete the collected evidence can be analyzed. Two utilities were used for this process: Access Data FTK Imager Lite and X-Ways Software Technology's WinHex [13]. These two tools were used to view the contents of the recovered memory image in order to perform the required analysis.

It should be mentioned that several forensic suites were considered. The main reason for not using them was monetary cost. Those offering free versions did not include any memory acquisition utilities. Therefore, the use of complete forensic software suites was discarded.

MEMORY IMAGE ANALYSIS

The image collection process was performed several times in order to be able to view the contents acquired. After collecting more than 20 images, a process was established. At a random time of day, after the target system had been in use, the USB drive containing the required tools was inserted into the computer's USB port. A command prompt window was the opened, and the investigator navigated to the thumb drive directory. Once there the DumpIT application was executed and the resulting image saved on the same USB drive. The two Sysinternals tools were then called upon, saving the results to a text file. No images (pictures) of the process of this process were produced in order to preserve as much system evidence as possible.

The image was then viewed using both WinHex and FTK Imager. The investigator sifted through the memory contents, viewing the data in hexadecimal as well as ASCII format. Figure 4 illustrates one of the memory images being viewed with WinHex. It should be mentioned that analysis of the images was never performed on the target system. A similar view of the same file using FTK Imager is shown in Figure 5.

Inspection of Figures 4 and 5 reveals that both software applications are very similar; none is better than the other. They both have option to view the data as hexadecimal or text, and they both identify the memory address in hexadecimal format. Both tools also display the data 8 byte (64 bit) columns, two columns at a time (128 bit view). The text representation shown in the images corresponds to converting each byte into its equivalent character using the extended ASCII format. Therefore, a maximum of 16 characters can be represented per line displayed.

In the images acquired, the first 8 MB of memory space were "empty", meaning that the only data represented were all zeros. Some of the acquired memory dumps did not contain any data for the first 20,480 bytes of memory space. This finding confirms the statements in [5]. Furthermore, operating system information such as registry keys and character encoding were found to be located at the beginning of the data or the end. The samples taken could not reveal a specific location used for the same data. At the same time, when system data was located at the beginning of the memory, user data was found at the other extreme. Therefore it can be stated that the operating system selects memory positions randomly at boot time, and separates the system from the user space consistently. This process usually results in two "halves"; the system half and the user half.

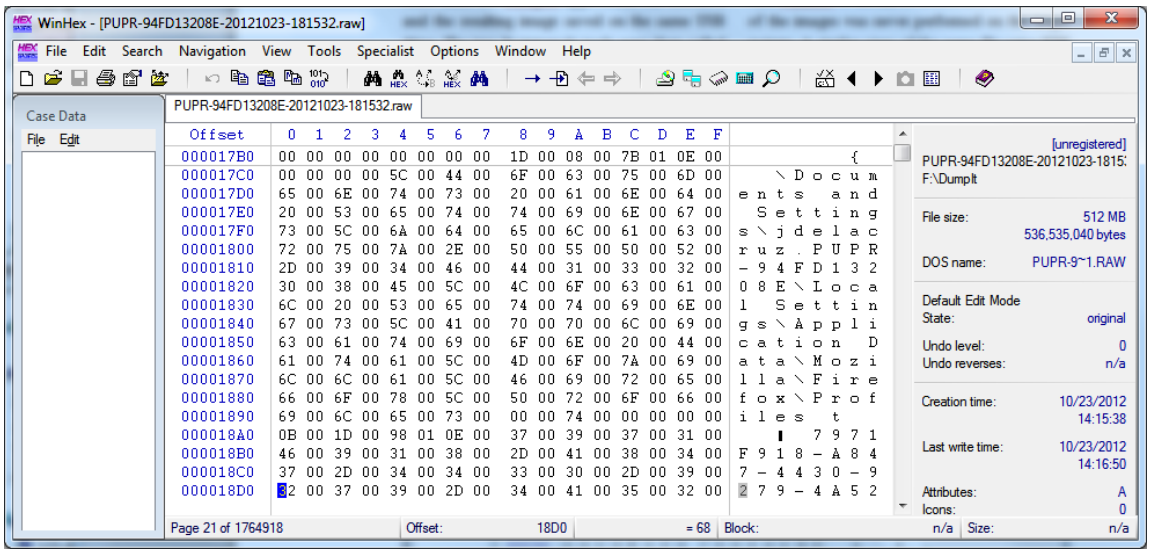

Figure 4
WinHex View

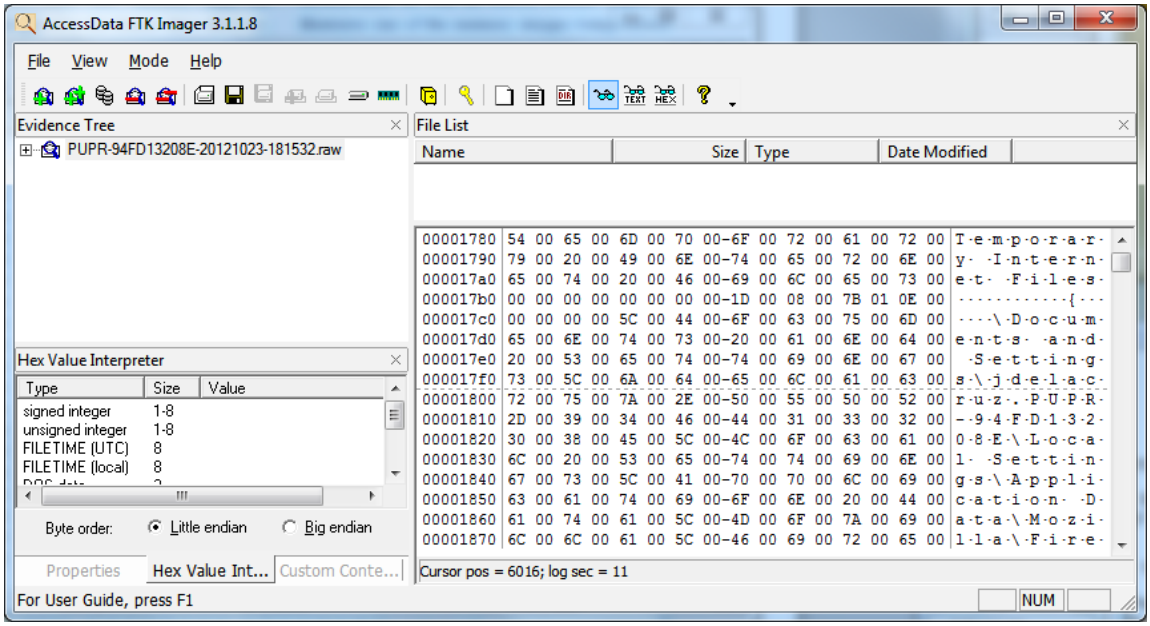

Figure 5
FTK Imager View

After the acquisition process had been completed, the investigator compared the information saved in the TCP View report with that shown by using the same tool but at the target system. The same procedure was used with the Process Explorer tool. The findings confirm that the data collected represents a correct and forensically sound copy of the actual system parameters.

Other tests performed included launching the Notepad application and writing the pangram "*The quick brown fox jumps over the lazy dog*" to the screen. With the application open, the memory image was acquired. The system was the shut-down and re-started. The process was then repeated, acquiring a second image with similar system parameters. Analysis of both images collected at shows that although in both instances the text written was located in memory, the memory addresses were completely different. This test was performed 10 more times during a period of one week with the same result. Not once was the text found at the same memory location in different images. Albeit timing concerns where some system processes might have been different at the exact time of memory acquisition, it can be stated that the operating system selects the memory address for each process according to an algorithm determined at boot time.

MEMORY ACQUISITION PROTOCOL

The Internet Engineering Task Force, IETF, develops many of the standards used in modern network communications. In RFC 3227 [14] the Order of Volatility for forensic evidence collection is defined. It states that the data to be acquired should be in the following order:

- Registers, cache
- Routing table, ARP cache, process table, kernel statistics, memory
- Temporary file systems
- Remote logging and monitoring data
- Physical configuration
- Archival media

The registers and cache are, at this time, "out of reach" to the investigator. These two components are part of the CPU and can only be accessed by the operating system. The data contained in these devices is also present in memory, but matching memory contents to the registers and cache is a task outside the scope of this project. The second set of items in the list

includes memory and network connections information. The research performed was directed towards these evidence sources. The rest of the items mentioned in the list are commonly acquired using standard forensic methods. This research project was set out to develop a forensically sound memory acquisition protocol.

The phrase “forensically sound” can be defined as evidence that can be determined to be correct and unaltered. The main problem with a forensically sound memory is the rate of change. As explained earlier, the rate at which data is removed and inserted into a computer’s memory exceeds the collection capability rate. It is very possible to start the collection process with a particular process block in memory and for that same process to be removed when the location it resides is acquired. Nevertheless, many of these more volatile processes are system utilities which should not be considered as evidence. User space process execution data collection should be the target for any investigator. This data reveals information about how the system was being used, rather than what the system was doing. The most important pieces of evidence that can be collected using memory acquisition are:

- Network connection data, including IP addresses and port numbers
- Active processes
- Encryption keys
- Page tables

To preserve their confidentiality, the keys used to encrypt and decrypt confidential data are themselves stored in an encrypted state. Once they are required for use, the system is responsible for obtaining the proper information in order to handle the confidential data. These “plain text” keys are then loaded into memory for their use. Memory acquisition provides the necessary tools to locate and identify these keys. The forensic examiner is then given more supporting information to perform the system analysis, being able to decipher previously encrypted data found in long term storage devices.

Page tables list the memory location of the processes that are in queue for execution, or those that are currently being executed. Since the operating system frequently swaps process data into the swap area of the long term storage device for the system, these pages tables offer insight into the last activities performed by the user at the computer.

In order to collect the aforementioned evidence items memory acquisition is required. But the collection process must follow a strict order and procedure for the collected data to be admitted as evidence. This order of procedure the forensic protocol developed as a result of this work.

The first order of business is to require a common set of physical tools. USB ports are present in most personal, laptop and business computers. These connections points enable users to rapidly transfer relevant data from disparate sources using USB devices (commonly called thumb drives). The ubiquity of USB ports and devices makes it the perfect choice for the physical medium that will contain the software utilities used in the collection process and be the destination of the acquired evidence.

The software tools required to perform the acquisition must be intensely evaluated. As stated earlier in this report, the utilities selected should be lightweight such as to impose the minimum disturbance possible to the target system. This investigator has clearly stated the preferred tools to be used, but the final choice remains an individual decision of each forensic investigator. If the forensic team uses X or Y forensic analysis systems, it is recommended that they select memory imaging tools that are closely coupled with their analysis software.

Once the physical and software tools have been selected the procedure followed must remain consistent. The main problem with memory acquisition is that the investigator only gets one chance. Therefore physical and software tool selection should be carried out in a controlled environment and proper testing techniques used. As memory is introduced into evidence in legal

procedures the courts will decide which tools are best suited for the process.

The forensic protocol developed provides a consistent procedure for obtaining forensically sound memory images. If all investigators follow the same protocol the process will be successful. The Forensic Memory Image Acquisition Protocol is now described.

Step 1: Photograph the computer system and the room, or area of the room, where it is found. Make sure that photographs of all actions performed on the computer are also photographed.

Step 2: Annotate any special circumstances that observed when the system was approach. These include, but are not limited to, open applications, peripheral devices attached to the system, and any noticeable processes being executed.

Step 3: Do not shut-down the system, or unplug any network cables.

Step 4: Insert the USB device into the system.

Step 5: Execute the memory acquisition software utility from the USB drive and store the memory image in the same device.

Step 6: Execute the Process Explorer (or similar) utility and immediately after the tool has captured the system's state save the results in the USB drive.

Step 7: Execute the TCP View (or similar) utility and immediately after the tool has captured the system's networking state save the results in the USB drive.

Step 8: Execute any other non-invasive evidence gathering utility, following the same procedure as in Step 7.

Step 9: Safely remove the USB drive from the system.

Step 10: Use the target system's hibernation utility instead of shutting it down. Hibernation preserves the state of the computer at the time the process is launched.

Step 11: Remove or disconnect network cables.

Step 12: Remove or disconnect power cables.

Step 13: Remove or disconnect all peripherals.

Step 14: Proceed with the accepted best practice for removing the computer's long term storage device, usually a hard disk drive.

Protocol Discussion

Following this protocol preserves the data contained in the computer system, and that in the storage peripherals attached. The continuous observance of this protocol will, hopefully, result in its acceptance by courts of law and its widespread use. The first two steps are part of the common forensic evidence acquisition procedure, and as such were not part of the investigation.

Step 3 states that the system should not be shut-down, or powered-off. This step is critical for the memory image acquisition process because at the time memory is left without power its contents become unstable due to the capacitive nature of the circuits used. Without power the memory acquisition process cannot be performed.

Steps 4 to 8 describe the order of procedures to be performed. After the forensic collection medium has been inserted into the system, it is of utmost importance that the memory image be acquired first. All other utilities employed, if any, should be used to corroborate the data captured from memory.

Steps 9 through 14 describe the proper "shut-down" procedure. When a Windows system is placed in hibernation, the operating system saves the state of the computer at that moment. Active network connections are archived, as well as any processes that are currently in queue. This procedure is better than the normal "shut-down" because the state of the computer is saved to the hard disk drive. For Step 14, the investigator should follow the pertinent best practice.

CONCLUSION

The work described in this report was directed at the development of the Forensic Memory Image Acquisition Protocol. The analysis performed was not aimed at locating different evidence items in the memory images acquired, but instead meant to validate the protocol developed. This Forensic Memory Image Acquisition Protocol should be able to withstand the analysis, and obtain the approval, of the courts of law.

Further research should be directed towards the effectiveness of the protocol's implementation. The adaptation of the protocol by a large number of forensic investigators is the only venue available to "field test" its success. Many forensic software suites are incorporating, or are in the process of including, memory analysis utilities into their packages. The effective collection of memory images using the protocol described will most likely result in better forensic analysis and enhances the results of any investigation.

ACKNOWLEDGEMENT

This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF1110174.

REFERENCES

[1] Committee on National Security Systems (CNSS), "National Information Assurance (IA) Glossary", CNSS Instruction No. 4009, 26 April 2010, www.cnss.gov/Assets/pdf/cnssi_4009.pdf.pdf

[2] Association of Chief Police Officers, "Good Practice Guide for Computer-Based Electronic Evidence", 2011, http://www.acpo.police.uk/ProfessionalPractice/Crime.asp

[3] Kornblum, J.D., "Using every part of the buffalo in Windows memory analysis", *Digital Investigation*, Volume 4, Issue 1, March 2007, Pages 24–29

[4] von Neumann, J., "First Draft of a Report on the EDVAC", 1945, downloaded from the World Wide Web, http://qss.stanford.edu/~godfrey/vonNeumann/vnedvac.pdf

[5] Russinovich, M.E., *et. al.*, "Microsoft Windows Internals", 2005, Microsoft Press

[6] Suiche, M., "DumpIT", *MoonSols.com*, http://www.moonsols.com/ressources/

[7] AccessData, "FTK Imager Lite", *AccessData.com, http://www.accessdata.com/support/product-downloads*

[8] Mandiant Corporation, "Memoryze", *Mandiant.com*, http://www.mandiant.com/resources/download/memoryze

[9] Mandiant Corporation, "Redline", *Mandiant.com*, http://www.mandiant.com/resources/download/redline/

[10] Russinovich, M.E., *et. al.*, "Windows Sysinternals", http://technet.microsoft.com/en-us/sysinternals

[11] Russinovich, M.E., *et. al.*, "Process Explorer", http://technet.microsoft.com/en-us/sysinternals/bb896653

[12] Russinovich, M.E., *et. al.*, "TCP View", http://technet.microsoft.com/en-us/sysinternals/bb897437

[13] X-Ways Software Technology AG, "WinHex", *WinHex.com*, http://winhex.com/winhex/

[14] Brezinski, D., *et.al.*, "Guidelines for Evidence Collection and Archiving", *Internet Engineering Task Force Network Working Group*, February 2002, http://tools.ietf.org/html/rfc3227