12-4-2013

Volatile Memory Acquisition Tools – A Comparison Across Taint And Correctness

William Campbell
Edith Cowan University

DOI: 10.4225/75/57b3bfa7fb867
11th Australian Digital Forensics Conference. Held on the 2nd-4th December, 2013 at Edith Cowan University, Perth, Western Australia

VOLATILE MEMORY ACQUISITION TOOLS – A COMPARISON ACROSS TAINT AND CORRECTNESS

William Campbell
School of Computer and Security Science, Edith Cowan University
wocampbe@our.ecu.edu.au

Abstract

The growth in volatile memory forensics has steadily increased in recent times. With this growth comes a need to test the tools associated with this practise. Although there appears to be a large amount of effort in testing static memory capture tools, there is perhaps less so for volatile memory capture. This paper describes the attempts at categorizing criteria for testing, and then introduces and extends upon a methodology proposed by Lempereur and colleagues in 2012. Four tools (Windows Memory Reader, WinPmem, FTK Imager and DumpIt) are tested against two criteria (impact and completeness). WMR and DumpIt were found to have the least impact, and also showed the greatest accuracy across the tests.

Keywords
Volatile memory forensics, impact, taint, completeness, comparison

INTRODUCTION

Digital memory forensics traverses a difficult path in the electronic wilderness. Important information must be discovered in its depths whilst maintaining the sanctity of the environment. Too much disturbance can lead to the inadmissibility and questioning of potential findings. Given this, much research has delved into finding solutions to the issues that this process faces.

Some guidelines suggest that evidence should be captured in order of volatility (Kent, et al., 2006). In this case volatility relates to the potential destruction of information over time. There also comes a seemingly more important premise to leave data and systems as undisturbed as possible (ACPO, 2012). What is not well documented is what happens when an attempt to capture information leads to a change in the system under investigation.

It is important in cases such as these to fully understand the impact and magnitude that an action may produce. From here, forensic investigators can make informed decisions about whether information is worth the potential risk to integrity. For example, does the use of a particular memory capture tool impact the system to a suitable level that capturing the systems memory is worth this impact?

Whilst there has been significant investigation and evaluation into static memory tools (see Guttman et al., 2011 for a summary example), there are fewer examples of dynamic memory capture tools. As a desire for this source of potential evidence grows, so too will a requirement to empirically study their collection. Unfortunately, the ability to categorically measure the effects of capturing volatile memory is a more complex than non-volatile memory.

By its very definition, volatile data (such as RAM) will change over time. These time-related changes will be polemic when attempting to measure changes due to other variables. For example, if changes in a system have been detected after a tool has been run, was this caused by the tool, or simply by time? What percentage of this change has the tool introduced, and what percentage has time? Asking these sorts of questions is perhaps easier than answering them.

There have been several different criteria suggested across the literature by which to measure and rank volatile memory captures (and thus the tools they were capture with). These include atomicity, availability, correctness, completeness, integrity, speed and interference or taint (Vomel & Freiling, 2012; Inoue,et al., 2011; Schatz, 2007). This investigation will assess tools across two of these criteria: correctness and taint.

Correctness refers to how representative a tools capture is of the original memory on the system, at the time of capture. A high level of correctness will indicate that a tool has been able to accurately capture the memory on the target system. 'Taint', on the other hand, refers to the impact of the tool on the system. This will be a measurement of how a tool has altered a system, an especially important criteria when considering the legal implications of evidence collection.

Clearly, the impact of time can be a complicated factor to account for when measuring volatile memory capture. Lempereur and colleagues' (2012) developed a novel solution to this problem. They engineered a design that employed two virtual machines, one to measure the effect of time (a control system) and one to measure the effects of an action and time (an experimental system). By comparing the difference between the two machines, the impacts of the action are theoretically obtained. This paper reports on the attempt to replicate the results found by Lempereur and his colleagues, as well as extend their implementation to cover the actual testing of tools.

This investigative methodology does contain several assumptions that must be considered. The primary assumption is that the state of the control virtual machine is an accurate representation of what the experimental system 'should' have been, assuming no manipulations had taken place. If this is true, any differences between the control and the experimental systems would be cause by the actions taken upon the experimental system.

An additional assumption in this research is that a byte-by-byte comparison is a valid measurement of memory change. As memory is expected to change over time on a given system (Lempereur et al., 2012), and if a byte-by-byte comparison is a valid measure, it should be capable of detecting these changes. Each of these two assumptions will require testing to validate the results.

When considering the use of a tool on a system, the act of capturing a systems memory can be broken down into three distinct phases: attaching the tool, navigating to the tool and executing the tool. Attaching consists of physically or logically attaching the tool to a device. For example, connecting a usb containing the tool onto a system. Navigating to the tool consists of interacting with the target system to locate the tool. For example, navigating to the tools location through the operating systems graphical user interface. Executing the tool consists of actually running the tool.
Technically speaking, a tool may be loaded or navigated to in several different ways. It may be that some of these methods alter memory more than others. For example, does navigating to a tool through a graphical interface produce greater or fewer changes to a systems memory than navigating to that same tool through a command prompt? As this question appears to be unsolved, it was felt important to separate the impact of loading and navigating to a tool, from the actual execution of one.

METHODOLOGY

Theoretical Approach

The research utilizes a methodology similar in nature to Lempereur and colleagues' (2012). Two identical virtual machine images will be used, with one of these systems employed as a control, and the other being experimented upon. Both systems will be 'powered on' for the same period of time. As such, the control virtual system, which has not been interacted with, should provide a baseline against which to measure changes in the experimental system. This will arguably remove (or at least greatly reduce) the impact of time, when studying the outcomes of interacting with the experimental system. By comparing the memory of the control and experimental virtual machines at certain points in time, it may be possible tp discover the impact of the tool. This will be used as a measurement for the taint.

Similarly, a comparison between a tools memory capture, and the experimental systems memory at the time the capture was started, will measure the tools accuracy (ie. completeness). It should be noted that the current measurement for taint is narrowed down to only the effects of a tool on volatile memory. For example, it will not account for changes to non-volatile memory, such as registry updates or changes. Although this is certainly an area of forensics tools that should be measured, this investigation is concentrating solely on the volatile aspects of taint.

PRACTICAL APPROACH

Two identical virtual machines images were created, one being labelled experimental, and the other control. These were based off a 32-bit Windows 7 image. These images were consequently initiated with QEMU (Bellard, 2013), and given 1024MB of RAM. They were also started using the snapshot flag on QEMU, meaning that any changes to an image were temporary. That is, on shutdown of the system, the image file would revert to its starting state.

Both virtual machines were controlled using a custom bash script. This script can be used to pause, resume and capture the memory for both VMs, using the QEMU hypervisor. It was also used to load a USB onto the experimental system, where required.

The experimentation was undertaken in two parts. The first set of experiments consisted of setting stationary points in time, at which both systems were paused, and their memories captured. No interaction other than this was performed on either virtual machine. These experiments would allow the testing of several assumptions that had been made, and to test the validity of the methodology as a whole. By comparing the two machines, the assumption that their memory states are identical can be tested. Additionally, measurements of their changes over time will indicate if a byte-by-byte comparison is a valid comparison technique.

Both systems were paused after 90 seconds, and their memories captured. The machines were then resumed and paused over three, 60-second blocks. Their respective memories were captured at the end of each of these blocks, giving a total of four captures for each machine. The experiment was repeated, this time using three, 180-second blocks. Each of these iterations (ie. 180-second blocks and 60-second blocks) were repeated several times.

The second set of experiments entailed testing several memory acquisition tools. See Table 1 for a summary of these tools. In these cases, the experimental system was loaded with a USB containing a specified tool. The tool was then navigated to, and executed on the system. During this process, both the experimental and control systems were paused at certain stages, and their respective memories captured. In all, four memory captures were created for each virtual machine (from figure 1, E1 to E4 and C1 to C4), and one was made by the tool itself (T5). See figure 1 for a diagrammatical explanation of this process. Each tool was tested a total of 10 times.

Tool	Shorthand	Use	Author
Windows Memory Reader v 1.0	WMR	Command Line EXE	ATC, 2012
Winpmem v 1.4	WINPMEM	Command Line EXE	scudette@gmail.com, 2013
FTK Imager CLI v 3.1.1.8	FTK	GUI EXE	Access Data, 2013
DumpIt v1.3.2.20110401	DUMPIT	GUI EXE	Mattieu Suiche, 2011

Table 1 - A summary of the tools tested in this investigation

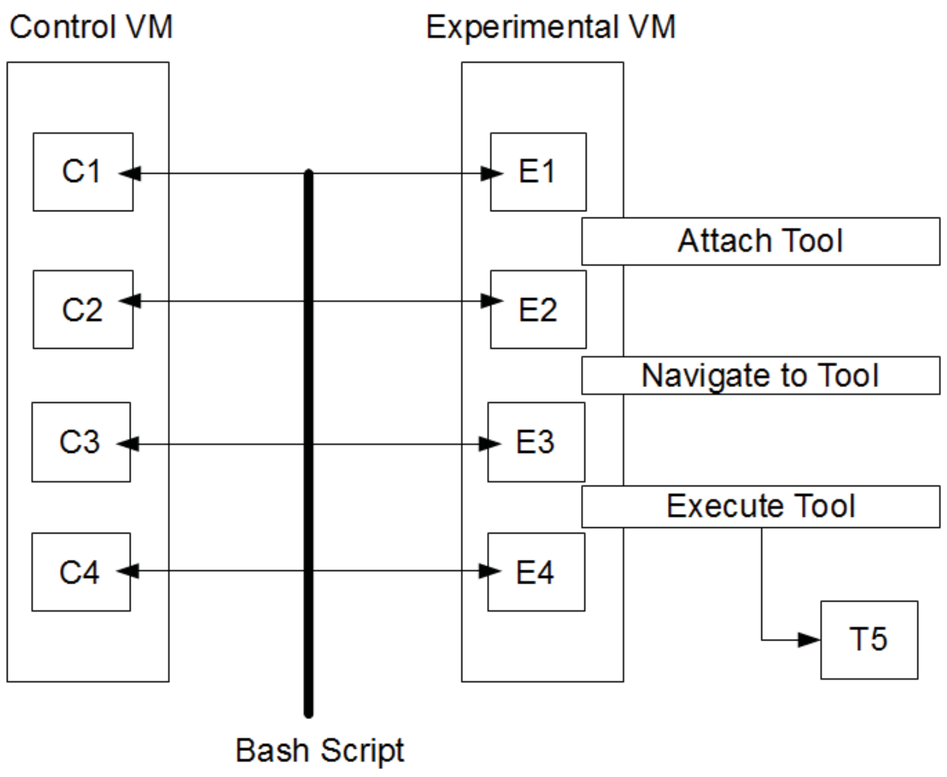

Figure 1 – A diagrammatical representation of the experimental methodology.
Note that the small square boxes (eg. C2) represent a memory capture.

From here, memory captures were compared to each other as needed. This was done by using a byte-by-byte comparison, to determine how many bytes were different between each capture. A higher number of byte differences was considered to represent a greater change between two captures.

RESULTS

Note that each capture file made by a tool was slightly smaller than that made by QEMU. Specifically, each tool capture was 8192KB shorter than that made by the emulator. Investigation showed that this difference was most likely at the end of the file, and did therefore not affect the comparisons.

Experiment 1- Difference Between Virtual Machines

Figure 2 shows the differences in memory for each virtual system over time. In this instance, a capture was taken at the 60, 120 and 180 second mark since the initial capture. The values are measured in percentage change based on the total amount of memory (ie. 1024MB). Each value represents the memory differences between the two machines across a certain time block.

Figure 3 represents a similar instance, this time with captures taken at the 180, 360 and 540 second mark. Again, the values are expressed as a percentage difference.

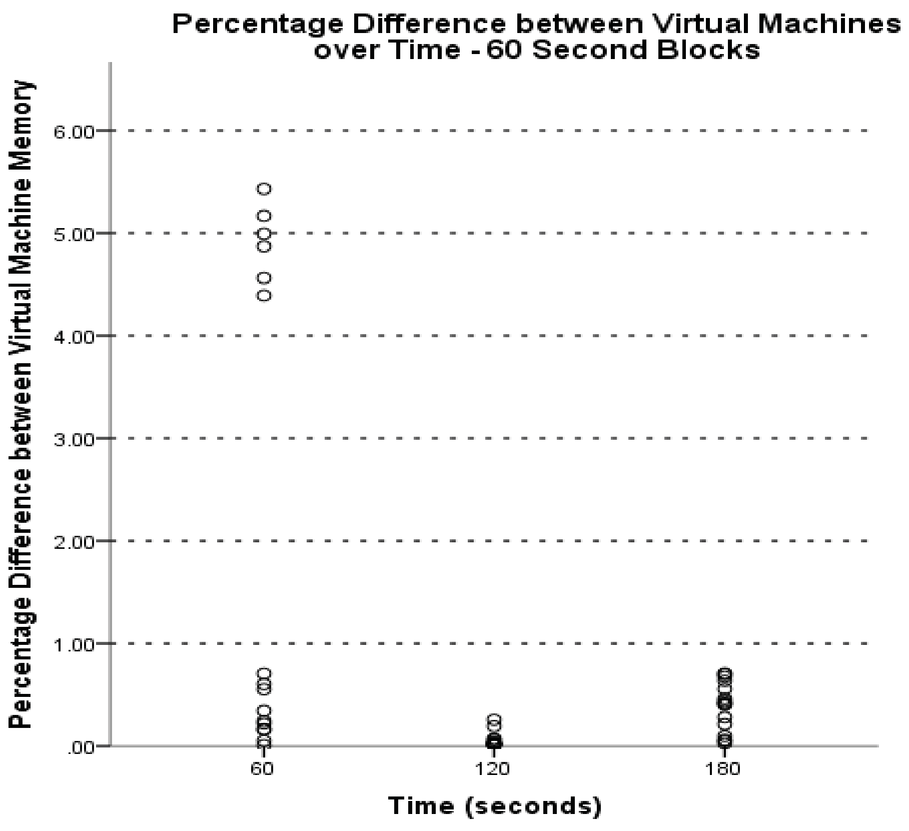

Figure 2 - The change in differences between the two virtual machines over 60-second blocks

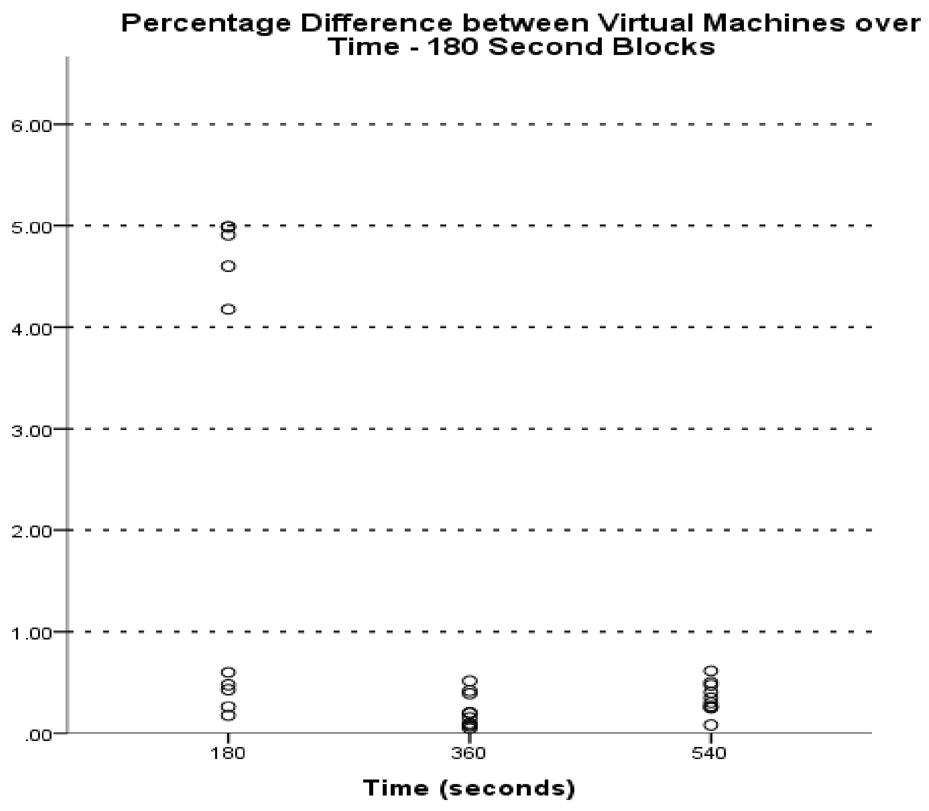

Figure 3 – The change in differences between the two virtual machines over 180-second blocks

As can be seen from figures 2 and 3, the majority of difference between the captures is less than one percent of total memory. This suggests that the memory changes over time are very similar between the two virtual machines.

There is some clustering of values around the five percent of total memory mark. A measurement of the mean values shows that this effect tends to be cancelled out across a large enough sample space (see Tables 2 and 3). It should be noted that the global mean for differences across the two machines was less than 0.3 percent of total memory across the 60-second blocks, and less than 0.16 of a percent across the 180-second blocks. These results would indicate that the two machines contain very similar memory states at the time of measurement.

Table2.The mean differences between Virtual Machine memory over 60 second blocks

Time	Mean	Number	Std. Deviation
60	-0.77	16	3.02
120	0.03	16	0.08
180	-0.04	16	0.46
Total	-0.26	48	1.76

Table 3.The mean differences between Virtual Machine memory over 180 second blocks

Time	Mean	Number	Std. Deviation
180	-0.73	10	3.47
360	0.10	10	0.26
540	0.17	10	0.36
Total	-0.15	30	1.99

Experiment 1 - Changes Over Time

The second assumption made was that a systems memory will change over time. A longer period of time would be correlated with greater changes in memory. Figures 4 and 5 show the mean value for each of the relevant capture differences (ie. The differences between captures 1 and 2, between captures 1 and 3, and between captures 1 and 4).

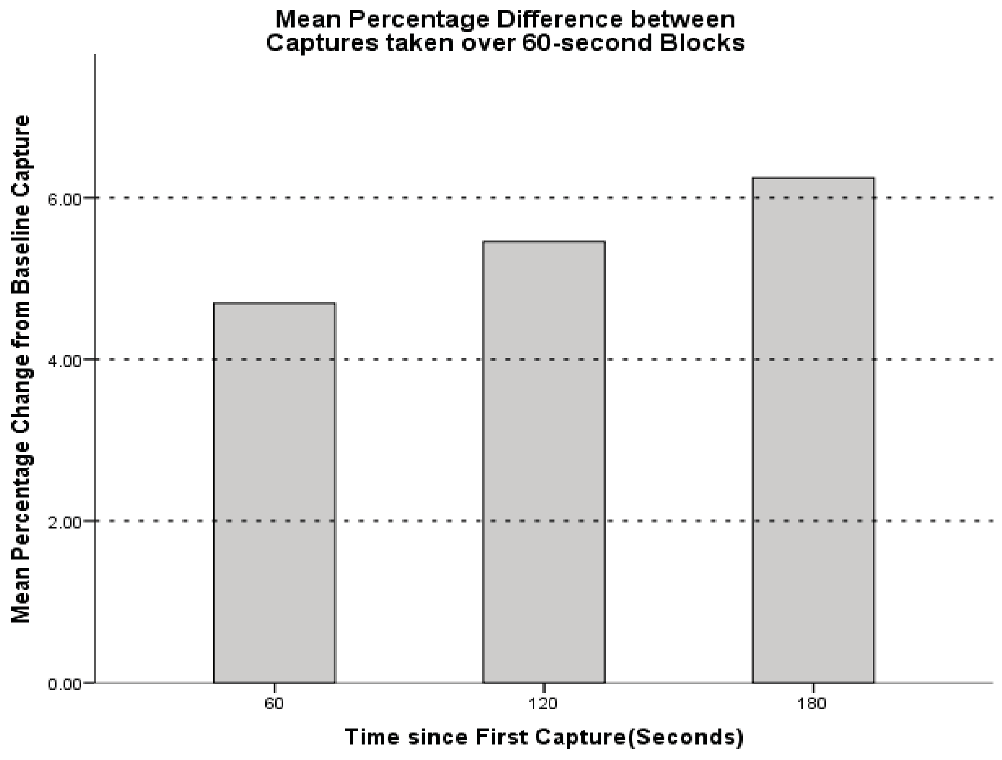

Figure 4 – Mean value of changes in VM memory across time (60 second blocks)

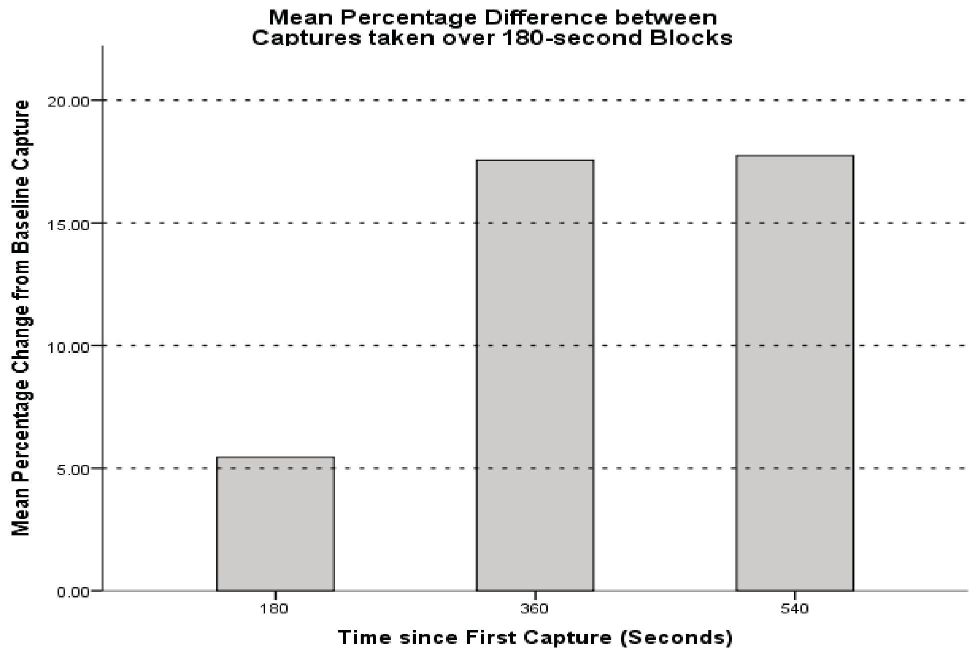

Figure 5 - Mean value of changes in VM memory across time (180 second blocks)

As can be seen from figure 4, the system appears to show changes over time. Figure 5 shows this change occurring on a larger scale (given the captures were taken over a large time period, this would be expected). It should be noted that the two systems appear to have changed very little in the final 180 second block.

Experiment 2

The second set of experiments consisted of the actual testing of memory acquisition tools. The first measurement was taken to study the impact of the tool on the system (ie. its taint). Figure 6 shows the impact of just the execution of a tool, where figure 7 shows the impact of the loading, navigation and execution of a tool.

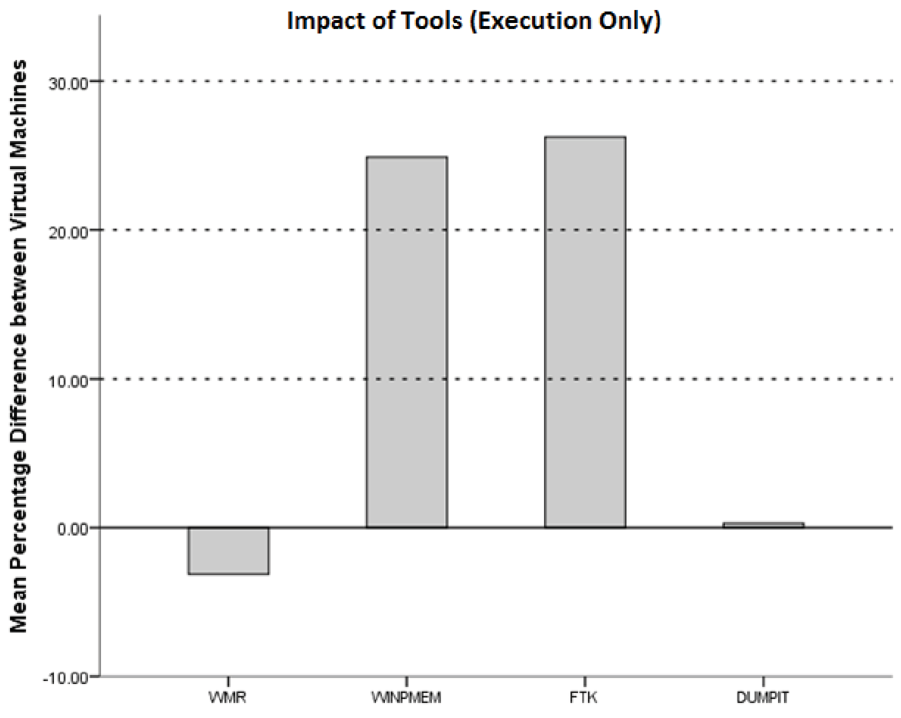

Figure 6 –The mean memory difference between the two virtual machines across the execution phase only

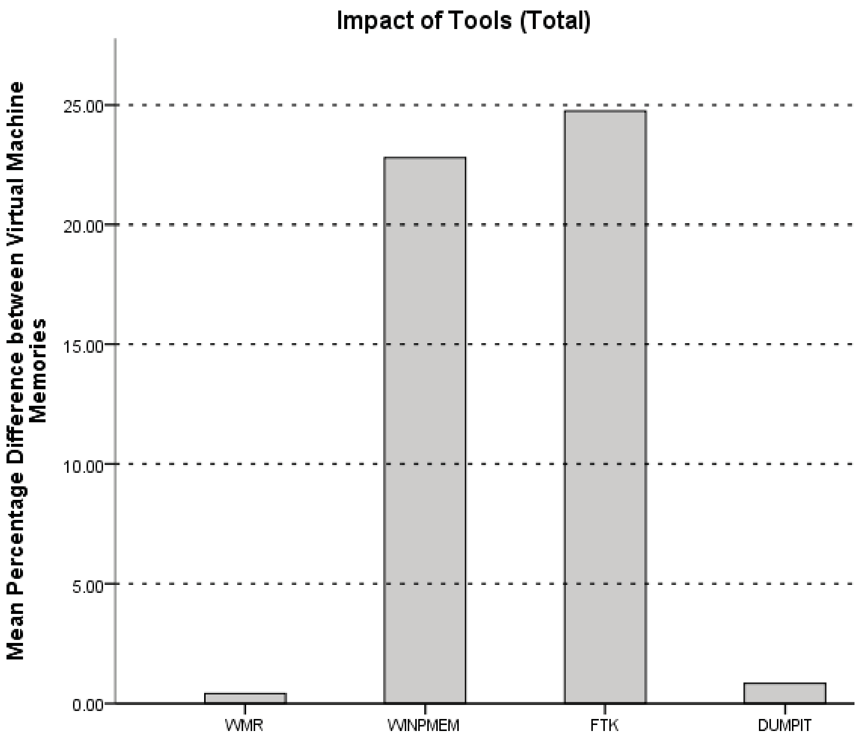

Figure 7 –The mean memory difference between the two virtual machines across the total lifespan (ie. Loading, navigating to, and executing a tool).

Note that a value of zero does not indicate that the tool had no effect on the system. A value close to zero indicates that the tool had an effect similar to that of a machine being left idle for the same time period. On the other hand, a higher value suggests a greater amount of change within a system than if it had simply been left to idle. In these figures, the negative value associated with the WMR tool indicates that the tool had *less* effect on the system than if the system had simply been left to idle. Possible reasons behind this are discussed in the final section. The accuracy of tools can be assessed by comparing the memory of the system at the start of the capture, with the memory capture achieved by the particular tool. From the methodology discussed earlier, this was the captures E3 and T5. The results of this comparison can be found in Figure 8.

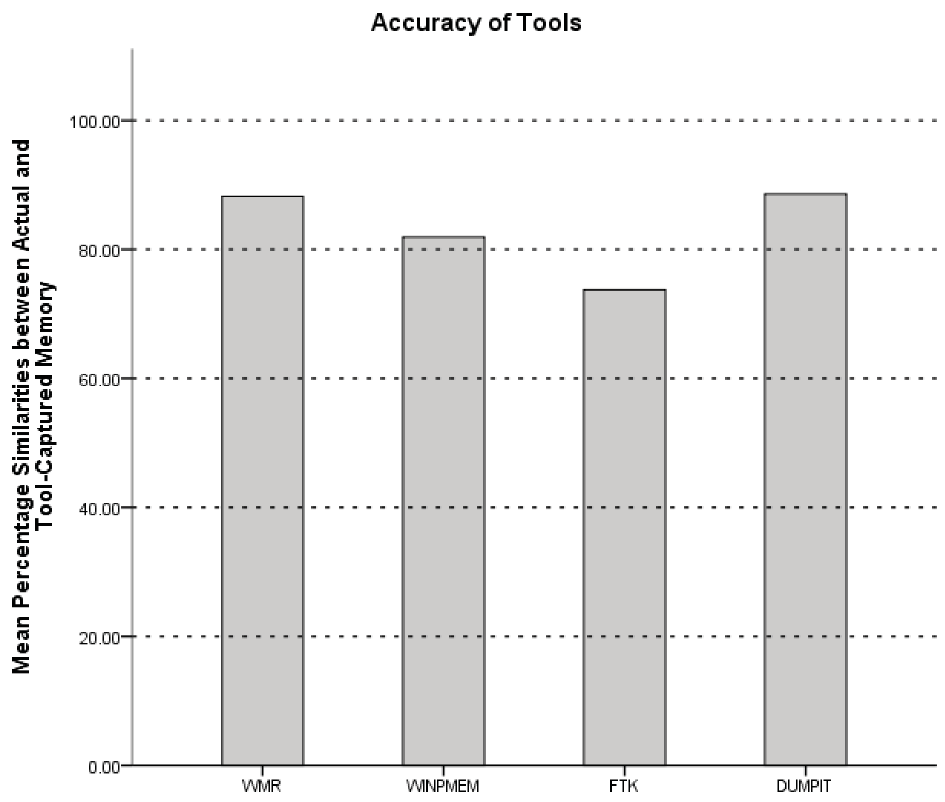

Figure 8 – Percentage similarities between the tools memory capture and the actual memory on the machine at the time of said capture.

In this case, a higher value represents a greater number of *similarities* between the tools memory capture and the memory of the experimental system at the time. This in turn indicates a higher level of accuracy. As can be seen, no tool was able to completely replicate the state of memory for the system.

DISCUSSION AND CONCLUSION

It appears that the methodology proposed by Lempereur et al. (2012) has been successfully replicated. Evidence seems to suggest that the control virtual machine provided an accurate representation of what the experimental system 'should' have been like, assuming no interaction had taken place. This is taken from the low difference rate between the two machines after an idle time period (less than 0.16% of total memory in the case of the 180-second blocks).

The clustering in the first block of each machine is concerning, although it does appear to even out over the average. It is hypothesised that this artefact is caused by an event during the system starting up. Although in experiment 1 the VM's were given 90 seconds before the initial baseline capture was taken, this may not have been enough. Should a starting event have occurred near this 90 second mark, it may have been captured on one VM but not the other. It is suggested that additional study be undertaken to assess exactly what causes this artefact, and to determine if it is preventable (such as by increasing the waiting time to 120 seconds for the initial base capture).

Of the four tools tested, DUMPIT and WMR appeared to have the lowest impact on the system. This is taken from the fact that each showed a very similar level of impact to if the system had simply been left idle. Likewise, these same two tools held the highest accuracy of the four tested. That said, there were some doubts cast upon the methodology, given the negative value associated with WMRs impact on the system. As suggested at the time, this would indicate that the tool had *less* effect on the system than if the system had simple been left to idle. This is somewhat counter-intuitive, given the fact that interaction with a system should alter its memory.

One possible suggestion for why this has occurred comes in the form of automated processes. It may in fact be that the Windows operating system, upon registering no activity for a certain period of time, will begin an automated process (for example, memory cleanup). As this event requires a state of no activity for it to occur, the experimental system will not be subject to this same event. If this automated process produces memory changes greater than that of the tool, it may create results as discovered in this experiment.

Overall, there is some positive evidence to suggest that the measurement of certain criteria regarding volatile memory analysis tools is possible. From here, it stands to reason that as tools are assessed and ranked, their desire to improve will increase. These increases in a tools attributes can only lead to the betterment of digital forensic investigation as a whole.

REFERENCES

Access Data (2013). "Product Downloads ". Retrieved 1 November, 2013, from http://www.accessdata.com/support/product-downloads.

Architecture Technology Corporation (2012). "Windows Memory Reader™ ". Retrieved 1st November, 2012, from http://cybermarshal.com/index.php/cyber-marshal-utilities/windows-memory-reader.

Association of Chief Police Officers (2012) ACPO Good Practise Guide for Digital Evidence.

Bellard, F. (2013). "QEMU: Open Source Processor Emulator.", http://wiki.qemu.org/Main_Page.

Guttman, B., et al. (2011). Ten years of computer forensic tool testing, from http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909329.

Inoue, H., et al. (2011). "Visualization in testing a volatile memory forensic tool." Digital Investigation 8: S42-S51.

Kent, K., et al. (2006) Guide to Integrating Forensic Techniques into Incident Response.

Lempereur, B., et al. (2012). "Pypette: A Platform for the Evaluation of Live Digital Forensics." International Journal of Digital Crime and Forensics 4(4): 31-46.

Schatz, B. (2007). "BodySnatcher: Towards reliable volatile memory acquisition by software." Digital Investigation 4: 126-134.

scudette@gmail.com (2013). "Winpmem release 1.4.1." from http://code.google.com/p/volatility/downloads/detail?name=winpmem-.4.1.zip&can=2&q=.

Vomel, S. and F. C. Freiling (2012). "Correctness, atomicity and integrity: Defining criteria for forensically-sound memory acquisition." Digital Investigation 9(2): 125-137.