SOUTHERN ILLINOIS UNIVERSITY
EDWARDSVILLE

Project proposal

Automated Memory Forensics Bootstrapping with Volatility

Manish Kumar Bobbili

under the guidance of

Dr. Thoshitha Gamage

November 4, 2016

Contents

List of Figures

List of Tables

1 Introduction

Computer forensics, a branch of forensics science, is used to investigate and collect evidence from a computer device. The goal of a forensic investigation is to do a structured investigation, find out exactly what happened on a digital system, and who was responsible for it. Due to the advances in technology, cybercrime has also been on the rise. For example, in 2010 a cyberwarfare weapon called Stuxnet [1] was released to allegedly sabotage the nuclear reactor PLCs (Programmable Logic Controllers) in Iran [1][2]. Along with the billions of dollars lost, this also caused political tension between United States and Iran.

The Zues malware [3] which has reportedly infected over 3.6 million computers in USA. Zeus used a webpage injection [3][4] to steal online bank accounts credentials. FBI in 2010, was able to arrest the guilty party with the help of computer forensics [5], is another example where memory forensics analysis is found helpful. In other words, computer forensics can be utilized to better understand previous attacks and how they worked to take precautions in future cases.

1.1 Computer Forensics

Meyers at el. in [4] defined computer forensics[1] , as "*the use of an expert to retrieve, preserve and, analyze data from volatile and non-volatile media storage*". Computer forensics can be defined as: "*the use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation, and presentation of digital evidence derived from digital sources for the purpose of facilitation or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations*" [6].In other words, a computer forensics expert will gather the evidence from volatile (primary memory) and non-volatile memory (secondary memory) and analyze it to find out what happened exactly in the memory sample.

A proper computer forensics analysis should be able to provide the past and present state of the evidence. There are many methods for investigation in computer forensics. The two most common methods are **live data investigation** and **dead data investigation**. Live data investigation is an investigation performed while the computer is still running. Dead data investigation is investigation performed by acquiring a recent memory image of the computer that is suspected of being infected [7].

An advantage of live data investigation is that artifacts gathered from a live system can provide evidence that is not available in memory [8]. A disadvantage with live data investigation is that it could lead to false evidence during investigation. For example, the reliability of the results found on an attacked server are questionable. This is because the investigation is being performed on a compromised system, which means the results could also be inaccurate [7][9]. Dead data

[1] The author referred computer forensics as digital forensics

investigation has some advantages over live data investigation counterpart. For example, some of the artifacts like closed network connections, unlinked processes, virtual address tree etc., are not available in static memory. Also, we can investigate the data in a trusted environment using trusted applications. This gives birth to another branch of computer forensics called memory forensics, which is the focus of my study.

1.2 Memory Forensics

Memory forensics is a branch of digital forensics which is used to gather artifacts on the volatile data (primary memory). Malware such as Code Red [10], Witty[11], and SQL Slammer [12] are the examples where the artifacts are hidden in the volatile data rather than the static memory [9].

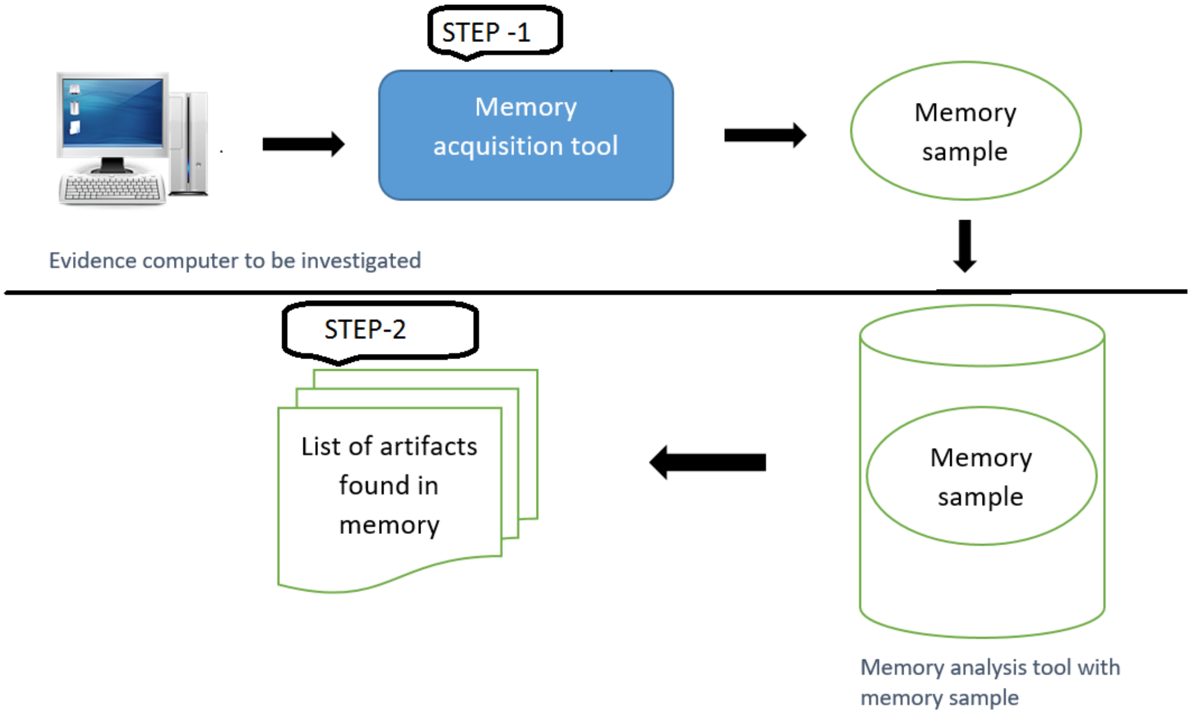

STEP -1
Memory acquisition tool
Memory sample
Evidence computer to be investigated
STEP-2
List of artifacts found in memory
Memory sample
Memory analysis tool with memory sample

Figure 1: Memory Forensics Investigate Process

1.2.1 Memory Forensics Investigation Workflow

Memory forensics investigation is a two-step process: acquiring the image and analyzing it with a tool. An illustration of the process is provided in Figure 1. Firstly, an image of the compromised system is acquired. There are several techniques for memory acquisition. Some techniques discussed by Ruff [13] are hardware-based acquisition, firewire bus, “dd” and “nc” tools, crash dump, snapshot, page file and hibernation file. In [14], the author discussed the importance of carefully acquiring sample to avoid any modifications to it. The second step is to analyze the memory sample

Table 1: Top Five Forensics Tools

Name of the tool	Implementation	Disadvantages
SANS SIFT [16]	VMware appliance which contains the combination of several other forensic tools such as (dd, volatility framework)to conduct an in-depth forensic investigation	It only works in Ubuntu operating system
Digital-Forensics Framework[17]	An open source tool which can be used to access the remote or local devices forensics, hidden files in windows and Linux OS.	Won't support live forensic analysis of the computer
Volatility[18]	An open source framework which investigates on volatile memory	The output of the tool wont differentiate between malicious and non-malicious tools.
The Sleuth Kit[19]	Open source tool which investigates specifically file system	Limited domain knowledge
Caine[20]	It provides a complete forensics environment which integrate the software tools as software modules for in-depth forensics investigation	Can perform only for live data forensics analysis examination.

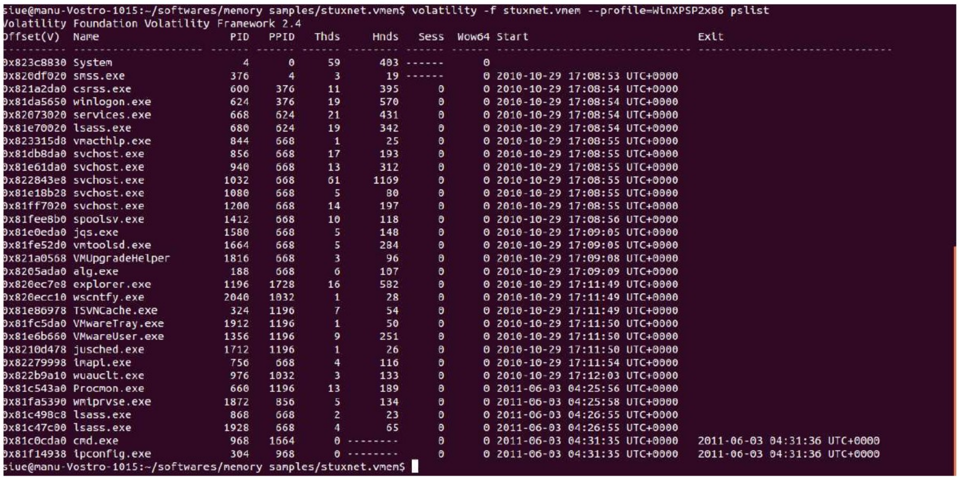
in a trusted operating system with memory forensic tools. In [15], Rafique introduced 38 forensics tools and including its advantages and disadvantages. In 2014 a survey conducted by Infosec Institute (*https://www.infosecinstitute.com/*) identified top five tools based on their features such as disk analysis, registry analysis, network analysis, malware, and rootkit analysis.

Most memory forensics tools produce the same type of result but in different format. Some of the common outputs from all the memory forensics tools are memory offsets of a particular program, timestamps, network connections, closed and open sockets. To gather any artifacts, a researcher needs to have expert knowledge of digital forensics and a thorough understanding of the operating system Specific internals. As a result, it takes so much time to do the actual investigation.

1.2.2 Volatility Framework

The Volatility Memory Framework is an entirely open assemblage of tools, executed in Python under the GNU GPL v2 for the abstraction of digital articles from unstable memory (RAM) models [21]. The extraction procedures are performed absolutely independent of the structure being investigated which allows for higher granular visibility in the runtime situation of the structure. For example, the volatility framework allows pausing an ongoing analysis at particular offset to perform an in depth analysis of a partial results.

The Volatility framework will help people to familiarize people to the techniques of how to extract artifacts from volatile memory samples. Figure 2 shows a snapshot of volatility analysis on Stuxnet, memory sample acquired from an affected computer that will show the running program list. In particular, the image has offset of every running programs with PID and PPID (parent PID). It also contains the number of handles handled by the running programs and with timestamps.



```
siue@manu-Vostro-1015:~/softwares/memory samples/stuxnet.vmem$ volatility -f stuxnet.vmem --profile=WinXPSP2x86 pslist
Volatility Foundation Volatility Framework 2.4
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x823c8830 System                    4      0     59      403 ------      0
0x820df020 smss.exe                376      4      3       19 ------      0 2010-10-29 17:08:53 UTC+0000
0x821a2da0 csrss.exe               600    376     11      395      0      0 2010-10-29 17:08:54 UTC+0000
0x81da5650 winlogon.exe            624    376     19      570      0      0 2010-10-29 17:08:54 UTC+0000
0x82073020 services.exe            668    624     21      431      0      0 2010-10-29 17:08:54 UTC+0000
0x81e70020 lsass.exe               680    624     19      342      0      0 2010-10-29 17:08:54 UTC+0000
0x823315d8 vmacthlp.exe            844    668      1       25      0      0 2010-10-29 17:08:55 UTC+0000
0x81db8da0 svchost.exe             856    668     17      193      0      0 2010-10-29 17:08:55 UTC+0000
0x81e61da0 svchost.exe             940    668     13      312      0      0 2010-10-29 17:08:55 UTC+0000
0x822843e8 svchost.exe            1032    668     61     1169      0      0 2010-10-29 17:08:55 UTC+0000
0x81e18b28 svchost.exe            1080    668      5       80      0      0 2010-10-29 17:08:55 UTC+0000
0x81ff7020 svchost.exe            1200    668     14      197      0      0 2010-10-29 17:08:55 UTC+0000
0x81fee8b0 spoolsv.exe            1412    668     10      118      0      0 2010-10-29 17:08:56 UTC+0000
0x81e0eda0 jqs.exe                1580    668      5      148      0      0 2010-10-29 17:09:05 UTC+0000
0x81fe52d0 vmtoolsd.exe           1664    668      5      284      0      0 2010-10-29 17:09:05 UTC+0000
0x821a0568 VMUpgradeHelper        1816    668      3       96      0      0 2010-10-29 17:09:08 UTC+0000
0x8205ada0 alg.exe                 188    668      6      107      0      0 2010-10-29 17:09:09 UTC+0000
0x820ec7e8 explorer.exe           1196   1728     16      582      0      0 2010-10-29 17:11:49 UTC+0000
0x820ecc10 wscntfy.exe            2040   1032      1       28      0      0 2010-10-29 17:11:49 UTC+0000
0x81e86978 TSVNCache.exe           324   1196      7       54      0      0 2010-10-29 17:11:49 UTC+0000
0x81fc5da0 VMwareTray.exe         1912   1196      1       50      0      0 2010-10-29 17:11:50 UTC+0000
0x81e6b660 VMwareUser.exe         1356   1196      9      251      0      0 2010-10-29 17:11:50 UTC+0000
0x8210d478 jusched.exe            1712   1196      1       26      0      0 2010-10-29 17:11:50 UTC+0000
0x82279998 imapi.exe               756    668      4      116      0      0 2010-10-29 17:11:54 UTC+0000
0x822b9a10 wuauclt.exe             976   1032      3      133      0      0 2010-10-29 17:12:03 UTC+0000
0x81c543a0 Procmon.exe             660   1196     13      189      0      0 2011-06-03 04:25:56 UTC+0000
0x81fa5390 wmiprvse.exe           1872    856      5      134      0      0 2011-06-03 04:25:58 UTC+0000
0x81c498c8 lsass.exe               868    668      2       23      0      0 2011-06-03 04:26:55 UTC+0000
0x81c47c00 lsass.exe              1928    668      4       65      0      0 2011-06-03 04:26:55 UTC+0000
0x81c0cda0 cmd.exe                 968   1664      0 --------      0      0 2011-06-03 04:31:35 UTC+0000   2011-06-03 04:31:36 UTC+0000
0x81f14938 ipconfig.exe            304    968      0 --------      0      0 2011-06-03 04:31:35 UTC+0000   2011-06-03 04:31:36 UTC+0000
siue@manu-Vostro-1015:~/softwares/memory samples/stuxnet.vmem$
```

Figure 2: Stuxnet Experiment Results

2 Problem Statement

The output of most common analyzing memory forensics analysis tools are in a pre-defined format. This output does not necessarily differentiate between malicious and non-malicious programs. For an investigator, this could be time consuming as the output should be understood before further investigations are performed. In other words, the results aren't **filtered to the specific investigation**. Due to this, the investigator will be spending more time filtering the data rather than spending that time on progressing the investigation.

The flow of experiments using volatility framework manually is shown in Figure 3. The experiment is performed in a hierarchical manner where the output of one experiment can be fed as input to next experiment. The forensics investigator has to do recursively all the experiments analyzing the output from every experiment.

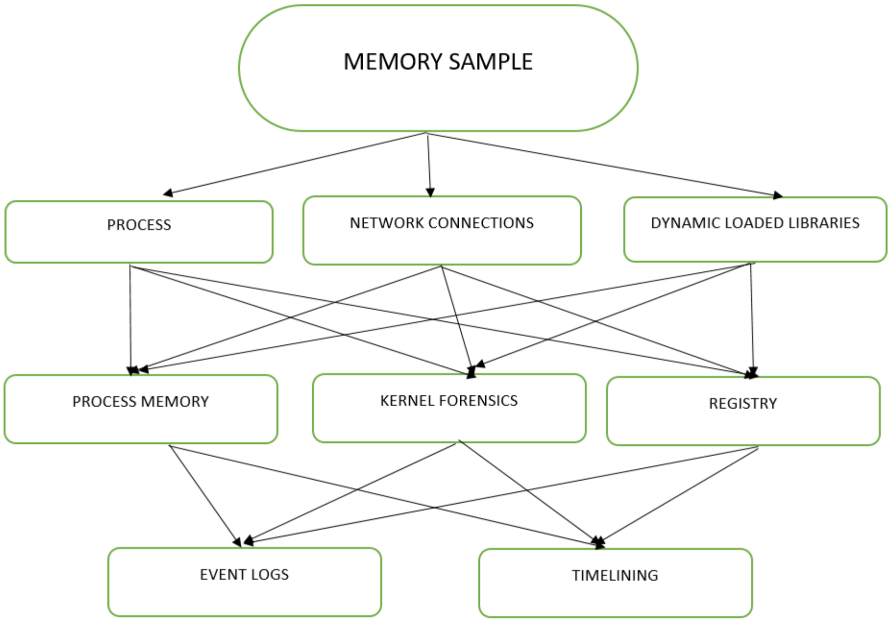


Figure 3: Flow of Experiments in Volatility with User Interventions.

3 Proposed Solution

In this work, I am proposing a new tool based on the Volatility framework. In my proposed work, I am using the output of the Volatility framework as input to my tool and to produce more accurate results. The output of the proposed tool allows investigator to be able to get basic information which he can interpret better and quicker.

3.1 Process

There are three layers in my proposed approach, which each layer has 2 to 3 experiments. The decision of which experiments to use at each layer will be determined by the output of the layer before. First part of the process would be feeding the memory sample to the tool which will then go through the Layer-1 investigation process. The Layer-1 investigation process will produce artifacts which will be used to decide the specific Layer-2 experiments. The outcome of the layer-2 experiments will decide the layer-3 experiments. At the end of Layer-3 the investigation is complete and the artifacts that are found is presented as output. The architecture of the tool consists of three layers, as shown in Figure 4.

- Layer 1: It consists of experiments which investigates the trivial information of the memory sample such as listing of running programs, network connections and dynamic loaded libraries.
- Layer 2: Based on the artifacts obtained from layer 1, an investigation on the kernel memory and registry of the sample is conducted.

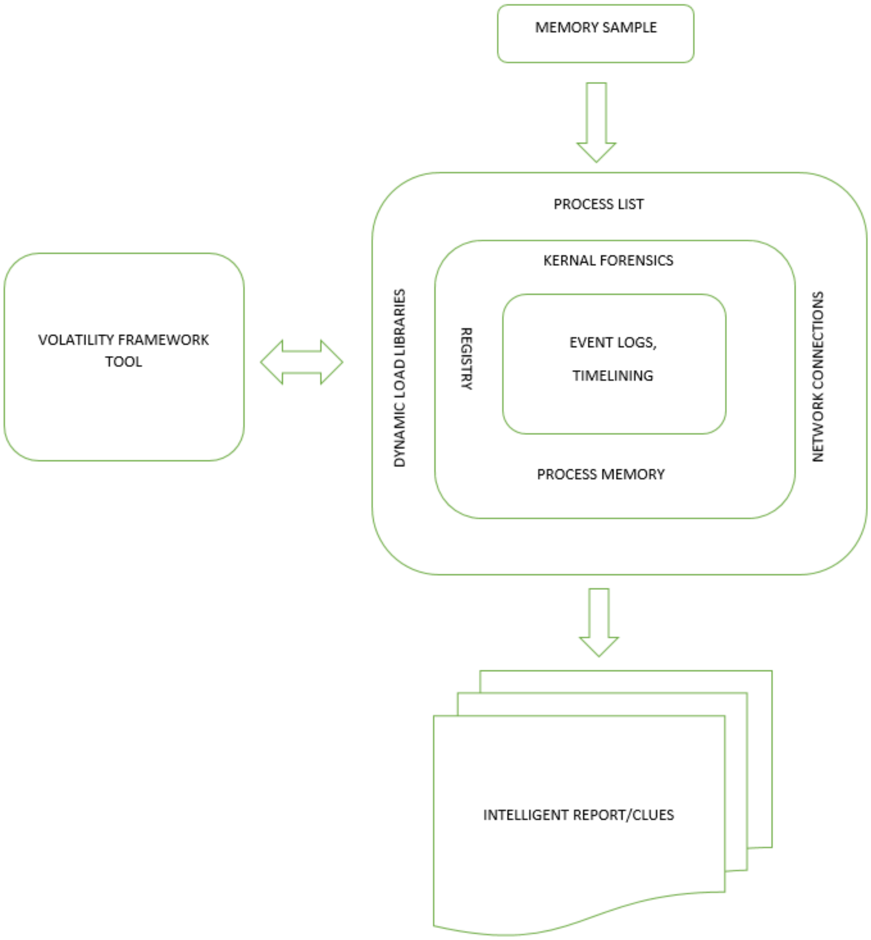
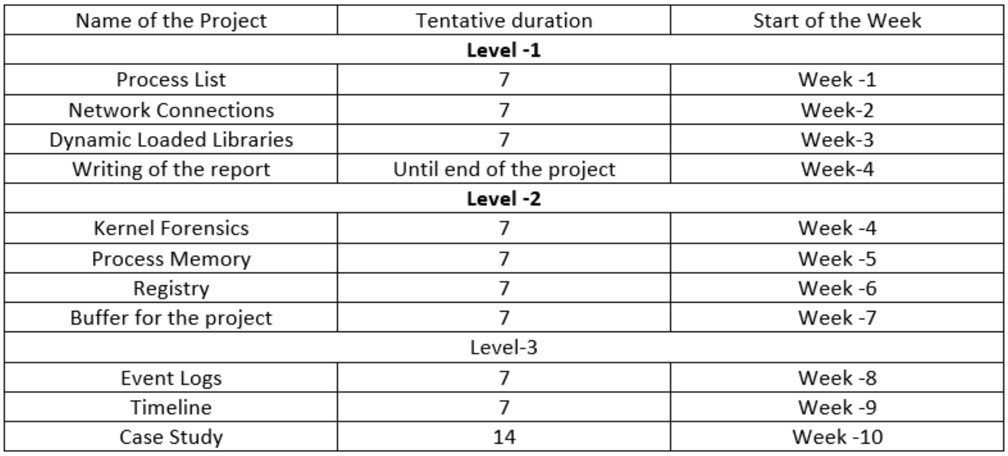


Figure 4: Architecture of the New Proposed Tool

- Layer 3: The final layer is the core layer. When we reach this layer we will be able to create the timeline of the memory sample by conducting the experiments such as event logs and time lining along with artifacts gathered from previous outer layers.

3.2 Expected Outcome

At the end of the Process, my tool should be able to generate a list of specific artifacts that an investigator can use in further analysis. More importantly, my tool will help to accelerate the forensic analysis by formatting the initial steps that must be performed in any investigation in an intelligent manner. This will reduce the manual intervention of users and automates the process to generate required output.



Name of the Project	Tentative duration	Start of the Week
Level -1		
Process List	7	Week -1
Network Connections	7	Week-2
Dynamic Loaded Libraries	7	Week-3
Writing of the report	Until end of the project	Week-4
Level -2		
Kernel Forensics	7	Week -4
Process Memory	7	Week -5
Registry	7	Week -6
Buffer for the project	7	Week -7
Level-3		
Event Logs	7	Week -8
Timeline	7	Week -9
Case Study	14	Week -10

Figure 5: Timeline of the Project with Duration

3.3 Validation

In my current research project, I am extending my study on memory forensics using volatility framework. The scope of this project is to fine tune the existing procedures used for finding artifacts in memory sample using volatility framework. I plan to validate the output of my tool by comparing it with what Volatility would have produced without any automation. In addition, I will attempt to replicate known results. For example, case studies from [22] and volatility labs [23] using my tool.

4 Timeline

The timeline of the project is shown in the above Figure 5 with the amount of duration for each experiment. The tasks shown in the image have dependency with previous tasks. The programming for automating the experiments is done as per the proposed architecture.

References

[1] R. Langner, "Stuxnet: Dissecting a Cyberwarfare Weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[2] S. Collins and S. McCombie, "Stuxnet: The Emergence of a New Cyber Weapon and its Implications," *Journal of policing, intelligence and counter terrorism*, vol. 7, no. 1, pp. 80–91, 2012.

[3] N. Falliere and E. Chien, "Zeus : King of the Bots," *Security Response*, 2008.

[4] M. Meyers and M. Rogers, "Computer Forensics: The Need for Standardization and Certification," *International Journal of Digital Evidence*, vol. 3, no. 2, pp. 1–11, 2004.

[5] B. News, ""More Than 100 Arrests, as FBI Uncovers Cyber Crime Ring"." `http://www.bbc.com/news/world-us-canada-11457611`, oct 2010. Accessed 01 Nov 2016.

[6] I. O. Ademu, C. O. Imafidon, and D. S. Preston, "A New Approach of Digital Forensic Model for Digital Forensic Investigation," *Int. J. Adv. Comput. Sci. Appl*, vol. 2, no. 12, pp. 175–178, 2011.

[7] B. D. Carrier, "Risks of Live Digital Forensic Analysis," *Communications of the ACM*, vol. 49, no. 2, pp. 56–61, 2006.

[8] F. Adelstein, "Live Forensics: Diagnosing Your System Without Killing It First," *Communications of the ACM*, vol. 49, no. 2, pp. 63–66, 2006.

[9] A. Aljaedi, D. Lindskog, P. Zavarsky, R. Ruhl, and F. Almari, "Comparative Analysis of Volatile Memory Forensics: Live Response vs. Memory Imaging," in *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pp. 1253–1258, IEEE, 2011.

[10] E. Chen, "Code Red Worm." `https://www.symantec.com/security_response/writeup.jsp?docid=2001-071911-5755-99`, 2001. [Online; Accessed 01 Nov 2016].

[11] C. Shannon and D. Moore, "The Spread of the Witty Worm," *IEEE Security & Privacy*, vol. 2, no. 4, pp. 46–50, 2004.

[12] D. Knowles, "W32.SQLExp.Worm." `https://www.symantec.com/security_response/writeup.jsp?docid=2003-012502-3306-99`, 01 2003.

[13] N. Ruff, "Windows Memory Forensics," *Journal in Computer Virology*, vol. 4, no. 2, pp. 83–100, 2008.

[14] A. Schuster, "The Impact of Microsoft Windows Pool Allocation Strategies on Memory Forensics," *Digital Investigation*, vol. 5, pp. S58–S64, 2008.

[15] M. Rafique and M. Khan, "Exploring Static and Live Digital Forensics: Methods, Practices and Tools," *International Journal of Scientific & Engineering Research*, vol. 4, no. 10, pp. 1048–1056, 2013.

[16] R. Lee, " Digital Forensics Training." `https://digital-forensics.sans.org/blog`, 10 2016. [Online; Accessed 01 Nov 2016].

[17] "Digital Forensics Framework." `https://www.demisto.com/digital-forensics-framework/`. [Online; Accessed 01 Nov 2016].

[18] "Volatility Open Source Framework." `https://www.volatilityfoundation.org`. [Online Accessed :01 Nov 2016].

[19] B. Carrier, " The Sleuth Kit and Autopsy: Open Source Digital Forensics Tools." `https://digital-forensics.sans.org/blog`, 10 2016. [Online; Accessed 01 Nov 2016].

[20] N. Bassetti, "Computer Aided INvestigative Environment." `http://www.caine-live.net/`. [Online Accessed on 01 Nov 2016].

[21] R. Carbone, "Malware Memory Analysis of the Jynx2 Linux Rootkit (Part 1)," *Valcartier Research Centre*, 2014.

[22] J. L. AAron Walters, Andrew Case and M. H. Ligh, *The Art of Memory Forensics Detecting Malware and Threats in Windows, Linux, and Mac Memory*. Wiley, 2014. Print.

[23] A. Waltors, "Volatiltiy Labs." `htttp"//http://volatility-labs.blogspot.com/`. Online Accessed on 01 Nov 2016.