National College of Ireland

Comparative Analysis of Open-Source Forensics Tools to Efficiently Conduct Memory Forensics

MSc Research Project
MSc Cybersecurity

Ashley Sunny George
Student ID: 21218714

School of Computing
National College of Ireland

Supervisor: Mr. Eugene McLaughlin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. …Ashley Sunny George……

Student ID: ……21218714..……..……

Programme: ……MSc Cybersecurity…… **Year:** ……1……..

Module: ……MSc Research Project……..……

Supervisor: ……Mr. Eugene McLaughlin……..……..……

Submission Due Date: ……14-12-2023.……..……

Project Title: …… Comparative Analysis of Open-Source Forensics Tools to Efficiently Conduct Memory Forensics

Word Count: ……8761…… **Page Count**……19……..……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: 

Date: ……14-12-2023……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Comparative Analysis of Open-Source Forensics Tools to Efficiently Conduct Memory Forensics

Ashley Sunny George
21218714
MSc Cybersecurity
National College of Ireland

Abstract

In a world where the development of new technologies are celebrated by everyone, criminals try to exploit it and cybersecurity professionals try to defend it. During an attack, one of the main tasks of the Incident Responders is to isolate the system/s which are being attacked and conduct forensics on it to understand the weight of the situation. This is done using various tools, both commercial and open source. Knowing how to use these tools efficiently comes with a lot of practice. This paper focuses on the open-source tools available for aspiring individuals to practice and learn these much-needed memory forensics skills in an efficient way. Over the course of this research, a comparative analysis is done on two sets of tools. The first one being the memory acquisition tools which includes DumpIt, FTK Imager, Belkasoft RAM Capturer, Magnet RAM capturer and Redline. The second set is the analysis frameworks where comparison is made between Volatility and Redline. The experiments conducted show promising results and suggests that the most efficient way to conduct memory forensics is by using DumpIt and Volatility in combination.

Keywords: Memory Forensics, Volatile, Efficiency, Open-source, Acquisition

1. Introduction

In this ever-growing field of technology, we cannot ignore the fact that each and every device is vulnerable in some or the other way and can be exploited. These devices can also be used to conduct criminal activities or to plan any criminal activity. A very common type of cyber-attack is when an attacker injects malware into a victim's system. Through the ages, the development of malicious software (malware) has become more sophisticated. This requires us as cybersecurity professionals to be aware of the latest developments in the dark world of malware.

A malware, just like any other piece of software, runs based on commands given by the programmer which is then executed in the RAM (Random Access Memory). The items stored in the RAM are not permanent and is thus called a volatile memory. This means that if an application taking up certain amount of RAM space is closed, the memory in the RAM is freed up too. The same runs in the case of a malware. When a malware is executed, it utilizes the spaces in the RAM and those spaces are then freed up once the malware finishes its execution. The contents of the RAM can be "dumped" and then be forensically analysed to understand the state of the system at that given instance. This also allows a forensic investigator to understand what damage the malware is doing to the system and much more.

In such cases, it is the role of a computer forensics investigator or incident responders to extract as much evidence as possible from these devices. We can take an example of a computer in an organization, connected to the organization's network being attacked by some malware. Investigating this would require the team to capture the memory (RAM) of the device to understand the processes running before, during and after the execution of the malware. In this research, we would be using all open-source tools such as Volatility, Dumpit, FTK Imager, Redline, Magnet RAM Capturer and Belkasoft RAM capturer. A comparative analysis on the performance of these tools during these simulations would be made to determine which tool and chain of tools is best suited for which scenario.

In the real-world scenario, most such investigations are conducted by big organizations with a good level of funding either by the government or investors. This gives them the advantage of using expensive enterprise level tools to conduct these tasks. But what about individuals who are just trying to learn computer forensics? Or even private investigators or security enthusiast? The expensive enterprise level tools are for sure very efficient and more powerful compared to the open-source tools. But keeping expense as a major factor for individuals, we shall focus on using open-source tools and come up with an implementation that would find an efficient way to conduct these processes and investigations. In this field of ever-growing risks, the need for cybersecurity professionals is growing exponentially. Organizations are in need of such professionals but at the same time, cannot just hire anyone with just a basic understanding of cybersecurity. Employers look for candidates with experience or with relevant skills. This is where the use of open-source tools will play a major role. Aspiring security professionals with little to no experience can increase their chances at getting employed in the field by growing their skills by using such open-source tools. The motivation behind this research topic comes from my own personal experience being a cybersecurity student trying to acquire the skills of conducting memory forensics. During my practices, using some of these tools would sometimes take up too much of my computer's resources such as RAM, GPU and processors. At times, even storage space. This would sometimes stand as an obstacle in learning the skills. This would also sometimes demotivate me in progressing further as I wonder if I will have to invest in a new device. Thus, I try to study and overcome this problem with this research.

Now using these tools can be confusing and may take up a lot of resource during the process. In this study, we will be monitoring the resources used by these tools on the host machine. The comparative analysis would consist of the study and analysis of these resources.

Past papers have definitely addressed these tools and have written various analysis on them but most of them fail to come up with an efficient method of conducting memory forensics using these open-source tools. Papers related to various fields such as Android forensics, Linux forensics, Windows forensics, analysis of tools such as Volatility and Autopsy, remote memory acquisition tools, physical and logical disk dump have been studied. Upon reading and reviewing these papers, it came to my attention that though these papers talk about the various processes with regards to the open source and enterprise level tools, they still do not address the issue of the most efficient way of using these tools keeping the main target audience as beginners in this field, trying to acquire new memory forensics skills. This then brings us to our research question:

"What set of open-source tools should be used to efficiently conduct memory forensics by using the least resources?"

This research is aimed to contribute towards one of the main problems stated previously. How a new learner may be able to go about attaining the skills for conducting memory forensics by using only open-source tools while shining light on the system resources consumed by these open-source tools so as to keep in mind the limitations of the systems used by a new learner. Special attention is given to the system resources as a part of the comparative analysis because most of the learners are assumed to not have access to very high-end systems either due to financial aids or any other reason. Thus, coming up with a comparative analysis would help individuals in choosing the right tool for the specified tasks.

In this document, we will further be discussing about some of the key literatures/papers that we came across which dealt with similar situations. In the literature review, we would be comparing various conference papers and other literatures and discuss the future works and gaps in them, leading us to our current research question. Post the literature review, we would move on with the research methodology where we will be discussing on how we initially planned to go ahead with the experiments and how we finally conducted it. The section would then be followed by the results of the experiments, discussion and what we can conclude from this whole research. Each of these sections would also have an element of self- criticism where the limitations, failure to conduct better experiments and more would be discussed.

2. Literature Review

Memory forensics though is not a very vast area of research, there still is an exceptional amount research papers written with regards to memory forensics. We will be talking about some of the past papers that dealt or talked about memory forensics, tools used and more.

A. Previous analysis on the tools

The first paper that we came across was “Memory forensics tools: a comparative analysis”. This paper written by Daghmehchi Firoozjaei, M, *et al*. Ghorbani gives us a starting point or a base to our research question. This paper analyzed three tools, Volatility, Autopsy and Redline. They run simulations of malware attacks and run these tools to analyse the malware. Analysis is then done by monitoring the resources used by the systems. The results are then compared. The above-mentioned tools are only tools that help with the analysis of the memory dumps. To add to that, Redline is an enterprise level tool and costs money. In order to acquire memory dumps, we need another tool for it. In the above-mentioned paper, the authors only talk about a tool called Forensic Tool Kit (FTK) imager for acquiring the memory dump. But is the best and the most efficient tool? We do not know. This gives us the required information about needing to conduct a comparative analysis on specifically open-source tools to find the most efficient way of conducting memory forensics.
Daghmehchi Firoozjaei, M., *et al.*(2021).

Similarly, another paper was written that compared some of the acquisition tools that helped in acquiring the memory dumps. The authors of this paper, Himanshu, *et al* talked about using open-source tools such as Forensic Tool Kit (FTK) imager and ProDiscover to acquire memory dumps which can then later be analysed using the other set of tools mentioned in the other paper. The two scenarios discussed in this paper were dumping the volatile memory and acquiring the non-volatile memory. The time required for the same were monitored and discussed by the authors.
Himanshu, *et al.*(2021).

From the above two papers we come to understand the uses of the tools used. Though not all the tools were open source, we do see that the majority of them were open source. The papers do talk about the resources used by these tools using the process and compares them. But each of the papers only talk about one of the processes. The first paper talks about the analysis and the second paper talks about the acquisition of memory and disk. The analysis would be different when it comes to clubbing both these processes together and then running the various open-source tools and finding the most efficient way of conducting memory forensics, which is done in our research paper.

B. Memory Acquisition

Acquiring the memory dump is the first step in the process of memory forensics. There are various open-source tools to do the same. Some of which are: DumpIT, FTK Imager, Memoryze and Belkasoft Ram capturer. During our course of research, we came across a particular paper titled: "*Comparison of Acquisition Software for Digital Forensics Purposes*" written by Muhammad Nur Faiz and Wahyu Adi Prabowo. This paper solely focused on doing an in-depth comparative analysis on tools used for memory acquisition. The tools used by them were: FTK imager, Belkasoft Ram Capturer, Magnet RAM capturer, Dumpit and Memoryze. All these tools are free and open-source. The findings of their experiments showed that FTK imager dumped 10 times more artifacts than Dumpit and Memoryze while Magnet RAM capturer caught the most number of artifacts which were also four times more than the Belkasoft RAM capturer. When it came to the time consumed, Dumpit was the fastest among the five while Magnet RAM capturer was the slowest. While this paper does do a complete and in-dept comparative analysis on these 5 tools, which is in the same methodology that we would be going ahead with too, it is important to notice that this paper was solely focusing only on the first step of the memory forensics process: memory acquisition. The paper is very informative and is not a very old paper either. It is fairly new (2018) and thus does stand up to the standards of modern research. But as mentioned before, it does not cover the complete process and moreover, does not come up with any efficient method to conduct the memory forensics process.
Faiz, Muhammad & Prabowo, Wahyu. (2018).

While we stay along the lines of memory acquisition, we also come across another paper which talks about the impact of tools on the process of memory acquisition. This paper is titled: "*Impact of tools on the Acquisition of RAM Memory*" written by Marcos Fuentes Martinez. The paper was written in 2020 and was published in 2021 in the Journal of Cyber Forensics and Advanced Threat Investigation, Vol 1(1-3), 3-17. This paper talked along the same lines as the previous paper discussed under this section. The tools used for the comparative analysis were the same, except that this author also used another tool called Winpmem which is part of a framework called the Rekall Memory Forensics Framework. The areas of comparison for this paper were the time consumed, amount of RAM consumed during the process of acquiring a memory dump and the actual size of the final output file. The author surely did touch all the aspects of a comparative analysis and the results are quite interesting when we compare it with the previous paper mentioned. As mentioned earlier, they used more or less the same tools to conduct their analysis. But when comparing the results of the experiments conducted by Muhammad Nur Faiz *et al*. and Marcos Fuentes Martinez, it could not be further apart from each other. This difference in results could be caused by various factors such as RAM, processor, clock speeds, etc. We can conclude from this that the efficiency of these tools can majorly depend on individual systems. At the same time, we must understand that while all of this might stand true, both

these papers do fail to address the rest of the memory forensics processes. Like mentioned before, memory acquisition is only the first step in the whole process of memory acquisition. In our paper, we will be focussing on not just memory acquisition, but on the other processes too which uses various other tools.
Martinez, Marcos (2020).

Now, though the above papers are reliable and informative, they fail to address the audience we are focussing on. The comparison they conducted were only on the resources consumed by these tools. They failed to talk about the functionalities provided by each tools, their user experience and the overall usability of the tools. These comparisons will be done in our paper keeping in mind our target audience being new learners who are interested in developing their skills in memory forensics.

C. Papers on Volatility and Redline

Following our base paper written by Mahdi Daghmehchi Firoozjaeia *et al*., we dive a bit deeper into finding papers that talk about Volatility as a tool in itself. Their research compared 3 major tools, Volatility, Autopsy and Redline. However, we would not be going ahead with Autopsy as this tool is mainly for conducting forensic analysis on the disks rather than the volatile memory, RAM. Volatility and Redline are tools specifically built to do this task.

A paper written by David J. Dodd from ADMIN Network & Security, titled "*Forensic Analysis with Redline and Volatility*" in 2014 talks about these two main tools that is used for memory forensics. He talks about how to use these two tools to analyze a memory dump. The author, in another document before this, talked about how to acquire a memory dump of an infected system, which he then proceeds to use it with Redline and Volatility in this paper. David J. Dodd downloads the malware, executes it in his closed VM set up and acquires a memory dump. He is then using both, Redline and Volatility to analyze and find the malware. In his paper, he initially talks about using Redline to view an hierarchical processes list to search for proccesses that might have been called by other suspicious processes. By doing this, he does come across a suspicious process that he thinks might be the malware.
On the other hand, he now use volatility to find the malware. He first tries to find any suspicious DLLs by using the function 'dlllist', followed by checking the proccesses. Though procedure he used for volatility is different, he was still able to find the malware.
In a nutshell, this paper used Redline and Volatilty on a memory dump file to find the hidden malware. But what this paper fails to report is the comparison between these two tools. He used two tools to find the same malware in the same file and reported the steps he took. What we will be focusing in our paper is the comparison between these two tools targetting our audience of new learners who know very minimal about digital forensics.
David J. Dodd (2014).

D. Other useful papers

One particular paper talks about using the Volatility tool to Acquire and Analyse Firmware-based Malware Rootkits. According to Jacob Taylor, Dr Benjamin Turnbull, and Dr Gideon Creech, a Firmware is a low level software that directly controls and communicates with the hardware of the device. For example, BIOS is a firmware. Firmware does not rely on any operating system as it is not an application level software.

Injecting a malware into the Firmware of a device can guarantee really high privileges to the malware as it directly would be able to communicate with the hardware, including the RAM. These sorts of malwares would also be invisible to a forensic investigator if not properly investigated. This paper titled " Volatile Memory Forensics Acquisition Efficacy: A Comparative Study Towards Analysing Firmware-Based Rootkits" talks about using Microsoft operating systems' ability to crash dump the memory without needing any external tools. The paper purely focussed on acquiring the memory dump to focus on the Firmware-based malware rootkits. This process of acquiring a memory dump without any external tool is somewhat a complicated process especially keeping in consideration that our target audience are new learners. We will be comparing different opensource tools that we can use to acquire a memory dump.
Taylor, J., *et al.*(2018).

On the other hand, going as far away from firmware, one paper talks about the analysis of memory dump to investigate the data left behind some of the portable private internet browsers such as Brave, TOR (The Onion Router), Vivladi and Maxthon. These browsers are known for their safe and private browsing. But how private exactly are the private (incognito) modes on these browsers? This question was addressed by the authors Meenu Hariharan, Dr. Akash Thakar, and Dr. Parvesh Sharma in the paper titled "Forensic Analysis of Private Mode Browsing Artifacts in Portable Web Browsers Using Memory Forensics". It is a known fact that browsers do tend to capture history and store it in the host system's memory. Though switching to "incognito" or "private" mode does not save the history on the browser, we do not know if it does save it on the system. The experiment utilized a bunch of tools to capture and analyse the volatile memory dump in scenarios such as when the browser was open and when it was closed.
Though these two papers do not seem to have much relevance with our topic, they still do discuss about the process of acquiring a volatile memory dump using open-source tools and also an in-built, not-so-traditional crash dump feature on windows. These papers laid a foundation for the further study that is to by conducted in our research which would to find an efficient way to conduct memory forensics using open-source tools.
Hariharan, M., *et al*. (2022).

E. Literature Conclusion

From the above-mentioned literatures, we can conclude that though we did have papers doing a comparative analysis on the various tools, we can spilt those papers into two categories. The first set includes papers that only compared tools that would help with the memory acquisition, while the second set only compared tools that help with the analysis of the memory that were acquired by these tools. From these papers we also fail to see the authors talking about any efficient method for a target audience to follow while conducting memory forensics. This paper would shine light on these areas keeping our target audience in mind.

3. Methodology

Given below is the detailed explanation of the methodology followed during the research. The aim of the research was to conduct a comparative analysis of the available open-source tools which are used to conduct memory forensics and come up with an efficient way to use these tools while at the same time keeping in mind our target audience.

A. Research Method

Memory forensics is a branch of Digital Forensics where it focusses on the Volatile memory (RAM) part of the system. Every such computer system has two types of memories, volatile and non volatile. In short, volatile memory is temporary while non-volatile memory is permanent. All computers need both these memories to perform smoothly.

A volatile memory such as RAM is a primary component of the device and is what is mainly responsible for loading up programs and making them into processes for the processor to execute them. This means that for any computer program to be executed by the processor, the RAM is required. And by this, we can also say that even if it a malware that needs to be executed, it has to be loaded into the RAM first. When this is done, the RAM can store information of the malware program such as its process ID, process name, etc.

This is where the importance of Volatile Memory Forensics occurs. In a typical real world scenario, during a cyber attack, it is not only necessary for the company/organization to eradicate the malware, but is also important for the company/organization to study the malicious program so in order to learn the damage the malware has caused and could have potentially caused if not for their timely intervention. Learning the malware can also help investigators find potential evidence leading to the criminals behind the malware and its attack. Another main outcome from conducting such forensic analysis would be to help other organizations/companies/communities be updated with the latest form of malwares and how to prevent them.

Thoughout this paper a set of experiments will be carried out. The results of those experiments will then be critically analyzed to then later come to a conclusion regarding the efficiency of each tool and the impact it could have on the learning experience of our target audience. The experiments are devided into two catogories based on the purpose of the tools; acquisition and analysis. In both the categories, the tools will be tested on various grounds such as time taken, overall/average CPU used, CPU resources (Keys and DLLs) used, breakdown of RAM used and also the overall usability, impact of the UI and the ease of istallation keeping in mind our target audience being beginners in this field of memory forensics.

Acquisition is the first step in general forensics where the forensics personelle is expected to gather as much evience as possible to help with the further analysis and investigation. In the case of memory forensics, our main peice of evidence that we would need to acquire is the memory and its contents. This can be done by a so called “memory dump”. A memory dump is a copy of the contents of the RAM in the given intance of time. It can be closely related to a screenshot as well. Various open-source tools are available that is solely created for acquiring an image of the RAM. The tools that we will be experimenting with in this research are:

Tool Name	Version	Parent company
DumpIt	3.1.0.0	Magnet Forensics
Magnet RAM Capturer	1.2.0.0	Magnet Forensics
Belkasoft RAM Capturer	1.0	Belkasoft
FTK Imager	4.7.1.2	Access Data
Redline	2.0.1	FireEye

Table 1: List of Memory Acquisition tools

Each tool acquires an image of the RAM in its own ways. In our experiment, we ran each of the tools individually, captured the time it took, measured the final file size and then later compared the artefacts collected by each of the tools using Volatility and Redline. Note that each of the tools generate different output formats too.

Apart from measuring the time and size, we also measured the resources used by the tools such as CPU and RAM usage. Monitoring the resource can be a bit tricky by just using the Task Manager available in Windows OS as it might require us to physically make notes of the values shown in real time. Or at least that is what I did during the first trial of the experiment till I found an alternate. “Resource Monitor” is an application that works as an extended feature of Task Manager where we can start and stop monitoring the resources such as CPU, Memory (RAM), Disk and Network. The resources that we would be paying most attention to is CPU and Memory. The network resource, though monitored, would not be required as we are just capturing a memory dump and this can be done without a network connection either. In a real-world scenario, the network activities will be monitored separately using a tool like Wireshark or any other such network monitoring tools. Keeping the aim of our experiment in mind, we do not require to monitor this resource consumption. Disk usage is also monitored by this tool but again, we would not be using this as a factor for comparison because the data we would receive from the tool would be the speed of reading and writing the data to and from the disk. Now this can be ignored solely because of the fact that it is not affecting the process of the memory capture very evidently. The speed of reading and writing depends on what time of Disk is installed in the system; i.e., SSD would do the task faster than a standard HDD. The system used of this experiment is equipped with both SSD and HDD. Thus, if we were to go ahead with comparing disk usages between the tools, we would not get accurate results.

To open this application, we can either open it through Task Manager -> Performance -> Click on the 3 dots on the top right -> Resource Monitor. This can also simply be done by searching for “Resource Monitor” in the search bar available in the Start menu. Once opened, we can start/stop monitoring the resources by clicking on Monitor -> Start/Stop monitoring. In our case, we will first open the application that we need to monitor, select the application in the resource monitor window to monitor the selected application specifically. We will then start the monitoring and immediately capture the memory. By doing this, we will be able to monitor exactly how much resource the application is consuming. Once the capture is done we will stop monitoring and then later takes notes of the data that is provided. We will perform the task for all the tools that we will be comparing and then finally compare the resources used. The tables and graphs of these will be in the results section of the paper. To demonstate an example of using this built-in tool for monitoring, we will use DumpIt and the sceenshots of the resources captured is given below:

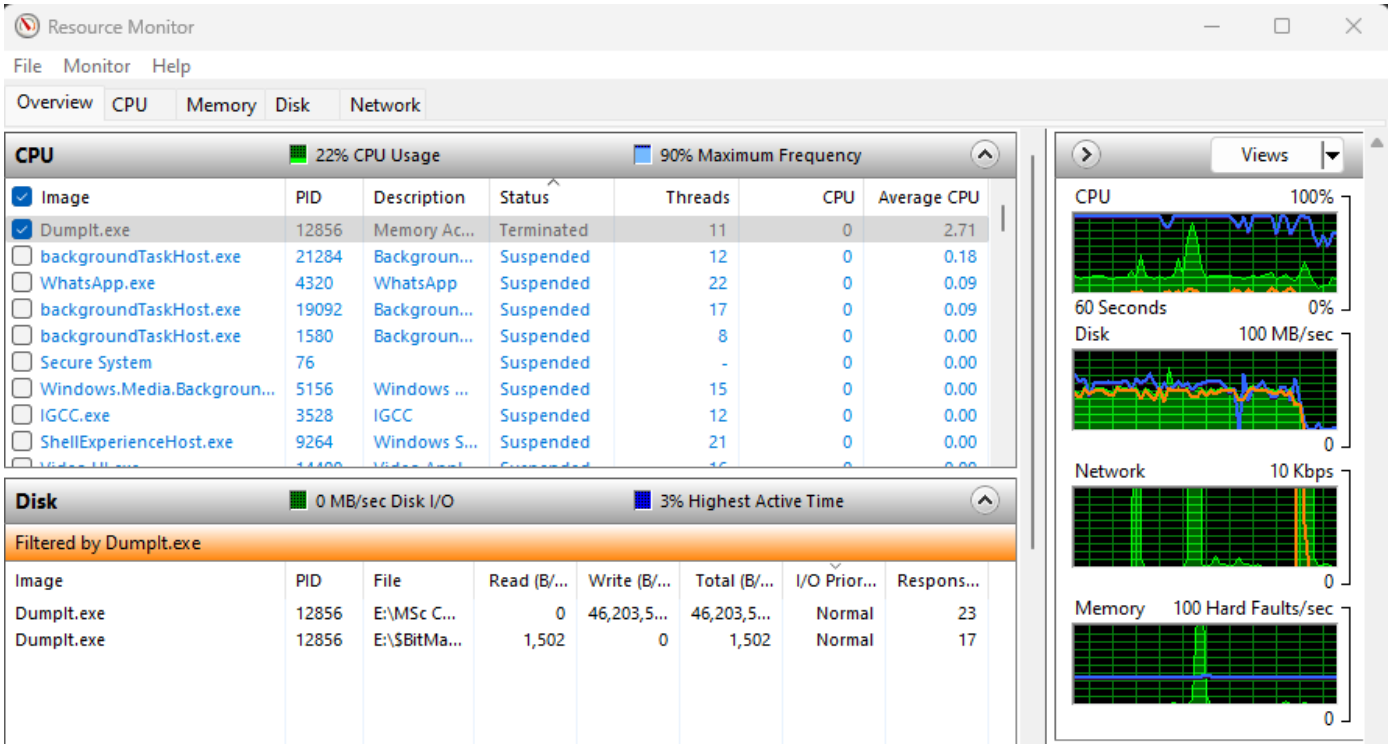

Fig 1: Example of Resource Monitor used for DumpIt

As we can observe in the above image, this tool is pretty handy when we need to look at the resources used by an application in detail. Though the screenshot only shows the overview of the resources used, we can dive deeper into it by heading into the other tabs: CPU, Memory, Disk and Network respectively. These tabs show much detailed information realated to the resources used by the application chosen. Here is another example where we can see in much detail the keys and DLLs used by the application.

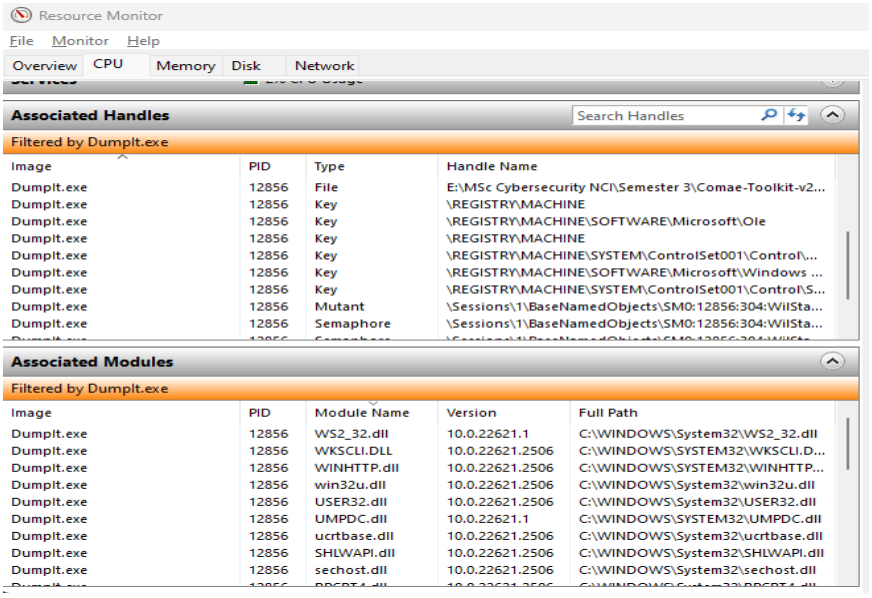

Fig 2: Example of detailed CPU resources used by DumpIt

This criteria of measurement can be deemed important during the comparision. Higher key useage would mean that the tool is actively interacting with multiple system configurations and higher DLL (Dynamic Link Libraries) usage would mean that the application is taking up more resources from the CPU. This type of

comparision would serve us great importance when it comes to determining the efficiency of these tools. More on this comparision is discussed in the results and discussion section.

As a part of this research, we must take into consideration the conditions and specifications of the host system used in this experiment. The experiments were solely run on the host system directly and not in a virtual environment in order to reduce the stress it would have on the resources such as CPU and RAM as it could affect our overall results when measuring the resources used up by the tools. Given below are the details of the system:

CPU	Intel(R) Core(TM) i5-6300U
RAM	16.0 GB
Clock Speed	2.50 GHz
Operating System	Windows 11 Pro
Architecture	x64
Device Manufacturer	HP
Model	Elitebook 840 G3
SSD	256 GB
HDD	1 TB

Table 2: Host system specifications

Considering that our target audience are new learners, we can assume that the specifications of their systems might be fairly similar to the ones mentioned above. This can give an idea of how well these open source tools might work on their systems.

The memory dumps captured in our scenario were just random dumps at random instances. Unlike other experiments, we did not infect the system with a malware and then capture a dump of the memory. This was followed since we did not use virtual machines as a means to reduce the load on our resources.

The next section of our research is to be dealt with two main tools: Volatility and Redline. As a part of the comparison for the acquisition tools, Volatility and Redline were used to compare the number of artefacts captured by the tools and to also check how well the images were captured; i.e, to see if the images were clean or corrupted. But apart from this, we also did use a dump file that was publically and freely available for practices. The memory file was taken from an infected system and was published on the internet for individuals to practice their memory forensics skills. The task related to this file was to find the hidden malware using the memory dump. Though we had no use to search for the malware, using a common trusted dump file was a great way of comparing the two main frameworks: Volatiliy and Redline. The details of the tools are as given:

Tool Name	Version	Parent company
Volatility	2.6	Various Authors
Redline	2.0.1	FireEye

Table 3: List of Analysis Frameworks

Though there are various other tools and frameworks that are used to analyze digital artefacts, there are only a handful of them that deal with analysing volatile memory artefacts. Two most popular among them are Volatility and Redline.

During the experiment, comparison was made based on various grounds such as ease of install, user experience, size, process speeds and resources used such as RAM, Disk and CPU. Like the experiments ran in the acquisition section, a total of 3 test cases will be run for each measurement and the average of which will be taken as the final values for comparison. Apart from these, we would able to comparing some of the funtions found in both these frameworks. Interestingly there were some big differences when it came to comparing the functions and options they offered.

For all the resource measurements, we will be using the built in Windows Task Manager to monitor and note the values. Values will also be taken before and during the the tool is run so we can know how much of the resources were exactly used up by the tools. All of this data is finally compared and analysed and an efficient way of conducting memory forensics is concluded along with the reasonings.

B. Research Resources

The resources pertaining to this particular research were not very complicated. To start off, we used only open-source tools. This takes away the burden of spending any money to conduct this research. Since I have only one device to conduct this project, simulations/experiments were be run on the host machine itself multiple tims with different tools. This cancels out the need for multiple devices.
No publically or privately available data set was used during the course of this research. The data set was created by me during the course of the experiments by recording and monitoring the required values. However, a safe and reliable memory dump sample which was publically available for practices and learning purpose had to taken from a GitHub repository created by "*volatilityfoundation*"(2019). This was done in order for us to make comparisons between Volatility and Redline using a common test sample file. None of the memory dumps acquired by me was taken for the second part of the experiment due to all the dumps being slightly corrupted and thus was not reliable for the experiment.

4. Results & Discussion

Given below is the comparison table for the memory acquisition tools with its basic properties.

Tool Name	Download Size	Installation Size	User Interface
DumpIt	1880 KB	4280 KB	Command Line
Magnet RAM Capturer	343 KB	343 KB	GUI
Belkasoft RAM Capturer	3018 KB	3018 KB	GUI
FTK Imager	52213 KB	128000 KB	GUI
Redline	76219 KB	78219 KB	GUI

Table 4: Size and UI comparison of acquisition tools

From the above comparison we can see mainly that DumpIt is the only tool that is Command Line while the rest are all GUI based. This is relevant as this research is addressed to a target audience of beginners who are just learning new skills in the field. A CLI for any beginner would be a bit complicated to understand. However, we must also note that this does not mean that the application itself need be installed and loaded from command line. The installation is not CLI but rather is just like any other software install on windows. After the installation, we just have to navigate to the application and double click it. It will then open up the command line to initiate the RAM capture. The difference in this could be crucial if the new learner is not familiar with the command line terminal.

We can also notice in Table 4 that FTK Imager is 152.22 times larger and Redline is 222.5 times larger in size than the Magnet Ram Capturer. The reason why FTK Imager and Redline are drastically larger in size is due to the fact that they are not just memory acquisition tools. FTK Imager is well known for analysing disk images and other digital forensic artefacts such as network forensic artefacts. Redline on the other hand is an endpoint security tool by FireEye which is used to detect threats and also used for memory analysis and will be studied in the further experiments of the research.

We shall now move on with comparing the resources used up by these acquisition tools. As mentioned earlier in the methodology, we will be measuring time taken, RAM used and CPU used. Given below are the graphs indicating the same.

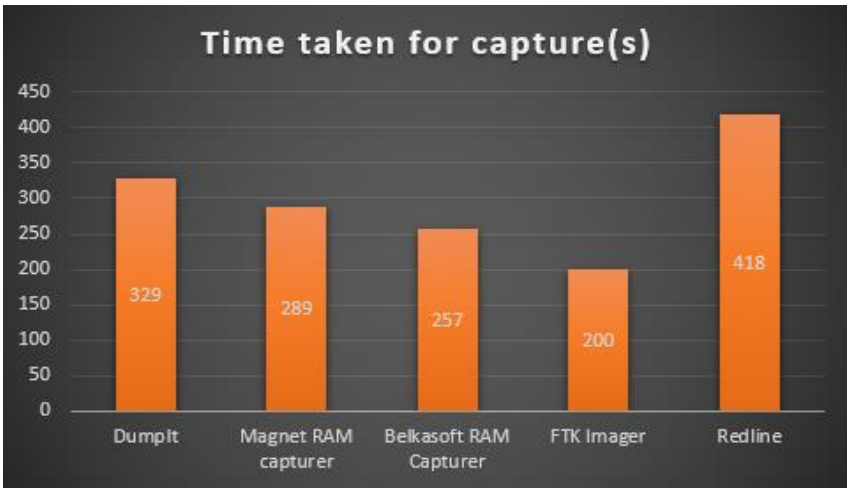

Fig 3: Comparison of average memory capture time taken

The above graph shows us the mean time taken by each of the tools to capture the memory. Each tool was made to capture the memory thrice as an attempt to evenly spread the data. Average of those three values per tool is shown in the graph above. Given below is the exact recorded time per case.

Tool Name	Trial 1 in seconds	Trial 2 in seconds	Trial 3 in seconds	Average time
DumpIt	321	354	312	329
Magnet RAM Capturer	266	306	295	289
Belkasoft RAM Capturer	259	274	238	257
FTK Imager	189	174	237	200
Redline	511	365	378	418

Table 5: Exact time in seconds recorded during each trial run

Moving on with the experiment, we capture memory again, but this time we focus on recording the resources consumed during the capture. As mentioned above, the system that we are using for the experiment has a 6th gen i5, clocked at 2.50 GHz and a 16GB RAM. Running the experiment as discussed in the methodology section we get the following results:

	DumpIt	Magent RAM Capturer	Belkasoft RAM Capturer	FTK Imager	Redline
Keys	6	103	8	44	120
DLLs	25	117	32	121	169
Average CPU Usage	2.71%	3.83%	2.09%	4.21%	3.97%

Table 6: Average CPU, Keys and DLL used by the tools

From the table above, we see that DumpIt uses the least CPU resources and Redline uses the most. But when it comes to the Average CPU usage, we have Belkasoft RAM Capturer with the least amount of CPU usage as seen in table 6. We must also very importantly note that the average CPU measurement done with "Resource Monitor" would not be precise as the value would go up when the tool is running (capturing memory) and it would go down close to zero when the tool is in idle. If the monitoring process (start/stop) is not timed precisely, there could be errors in the average percentage, as being in idle would still contribute to the average calculation. This is why we see the average CPU usage of Belkasoft RAM Capturer is lesser than the usage of DumpIt with a difference of 0.62. Keeping this human error in mind, we can go ahead with comparing mainly the Keys and DLLs to give us a better picture of the CPU usage. But we can also look at the average CPU usages for comparison if we can take into consideration the human factor and use the data as a rough estimate.

The next part of this comparison is with the load the tool puts on the RAM of the system. Given below is a detailed graph of the results of this experiment:

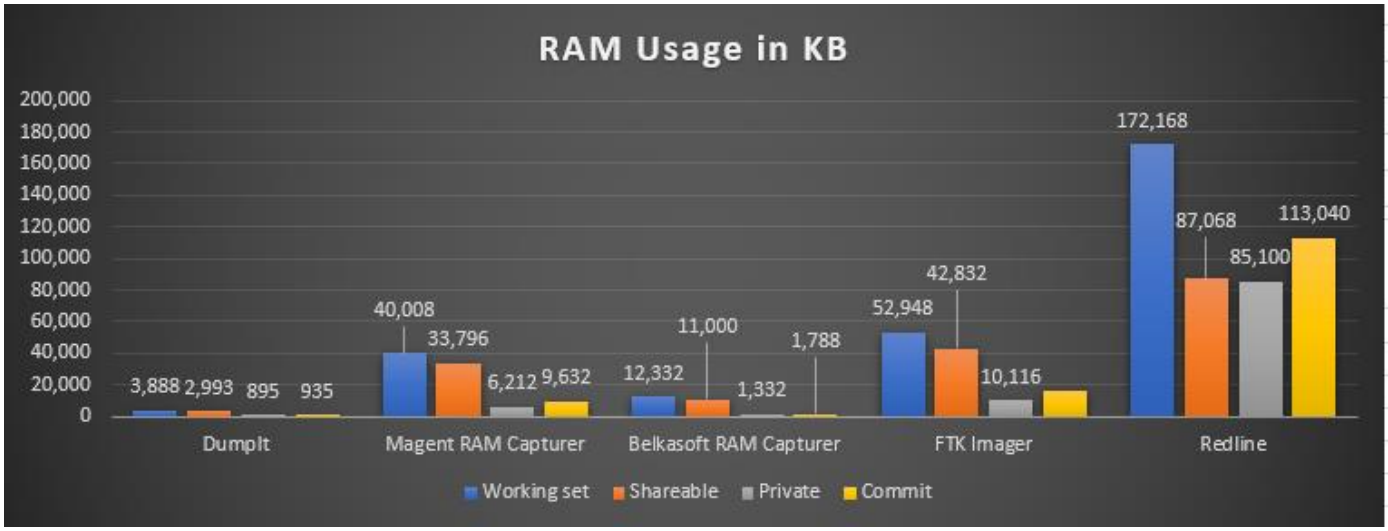

Fig 5: Graph of RAM usage of the tools in KB

From this graph we can see how light DumpIt is when it comes to RAM usage. A major reason why this tool can be this lightweight maybe because of its CLI. This reduces the processing needed unlike other tools where graphics have to be processed and it also shows other information like how much percent of the capture is complete and estimated time. Belkasoft RAM Capturer also shows in real time exactly how much MB of the RAM is captured. These were just assumptions on why each tool takes up more or lesser RAM. The category private refers to the space allocated for that tool by the system and that space cannot be shared with any other application. In the case of Redline, 113,040 kilobytes of memory was strictly allocated to Redline and FTK imager will not be able to access that memory space. Shareable is the space taken up by the application but this space is shared with other application too. Which means that the 42,832 kilobytes of shareable memory taken up by FTK imager could be shared with Belkasoft RAM Capturer too. Working set is nothing but the total live space taken up by the application (shareable + Private = Working Set). Commit is the virtual memory allocated for the processes by the system. In essence, the working set is shows the total RAM taken up by each of the tools.

This form of experiments was also conducted by Muhammad Nur Faiz *et al*., and Marcos Fuentes Martínez. We do see a common pattern of results among the papers, including this current paper. Results like, Belkasoft RAM capture being faster than DumpIt, FTK imager taking up more RAM compared to Belkasoft, Magnet and DumpIt, and last but not the least, DumpIt taking up the least amount of CPU resources compared to others. Though the results of my experiment do not line up identically with the experiments conducted in the previous papers, it does show similarities which can confirm that the experiments conducted in this paper were done right. However, there was more room for improvement as I could have taken up more tools such as Memoryze and LiME as a part of the comparative analysis.
Faiz, Muhammad & Prabowo, Wahyu. (2018).
Martinez, Marcos (2020).

Now that we can see clear lines for comparisons among the acquisition tools, we will not move ahead with comparing our two main frameworks that we would use to analyze the memory dumps collected; Volatility and Redline. From the above data collected for the acquisition tools, we can already see that Redline is

already using up a lot of resources. But this was when the tool was used to capture the memory. We shall not see how the tool performs in comparison to Volatility when it comes to analyzing the captured memory.

As discussed earlier, for this part of the experiment we used a common dump file from a trusted source on GitHub. This was chosen in order to give us a common scenario so we could compare the contents of the images that were successfully captured by either of the frameworks. None of the dump files captured by me in my acquisition part of the experiment is going to be taken into consideration due to the irregularity in my data and also because some parts of the captures were corrupted due to unknown reasons. This led to some of the functions of the framework forever to load.

Here is an initial comparison of the tools:

Name	Size	User Interface	Mode of install	Other notes
Volatility	32.7 MB (Python) 15 MB (Standalone)	CLI	CLI (python)	Standalone version available.
Redline	211 MB	GUI	Windows Installer	No command line

Table 7: Initial comparison of Volatility and Redline

From this table alone we can see how different these two tools are. Volatility is completely command line and does not have a GUI version available. Keeping our target audience in mind, assuming that they do not have a lot of experience around command line, they would find it hard to navigate around the framework and get the most out of it. Redline on the other hand is fairly straightforward with it's GUI which could help new learners in understanding the image and also helping them better analyze the images. The installation process is also much simpler with the Redline. A direct download from the website will give the user a windows installer file. They then just have to follow the usual installation process which is fairly simple and the framework would be installed. Volatility on the other hand has two versions of itself. The most commonly used one is the python version. Now this does mean that the user has to have python installed in their system and added to the system path. The standalone version is a much simpler version and it does not require any installation process at all unlike the python version which would use "pip" to install the downloaded file. The standalone version is available for download directly as an exe. This reduces the process of breaking the user's head for installation using pip. But once after downloading the standalone version, the user must know exactly how to run the framework. Given below is an example command of using the Volatility standalone framework:

```
PS E:\MSc Cybersecurity NCI\Semester 3> .\volatility -f 0zapftis.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
```

Fig 6: Example command for Volatility Standalone

Such commands might seem a bit complicated for a beginner to understand and grasp. But considering that Volatility is also used in the professional environment, it would be a good reason to catch on and learn to use the tool.

Redline on the other hand gives the user a fairly straight forward GUI to work with, allowing the user to be able to understand and quickly grasp the features of the framework and also the process of analyzing memory dumps.

Here are the comparisons of the frameworks in terms of RAM:

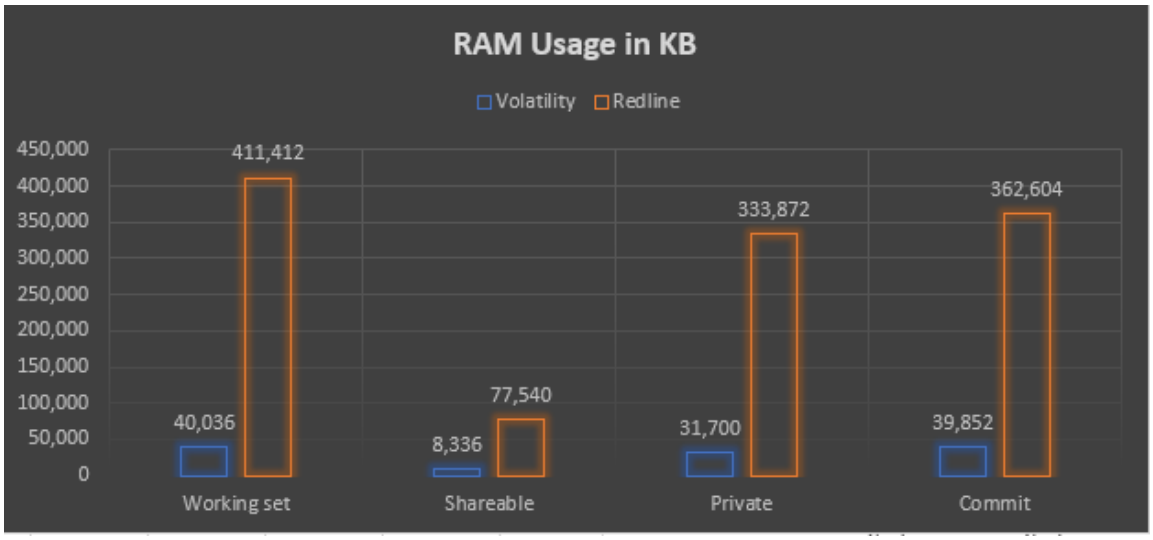

Fig 7: RAM usage for Volatility and Redline

Just like the comparison made for the acquisition tools, we can see how much of a difference the two tools have when it comes to RAM usage as shown in the graph above. Considering the working set (shareable + private), there a difference of 371,376 kilobytes of memory. Or in other terms, Volatility only took up 9.731% of the space Redline took up, almost less than a 10^{th} of the resource taken up by Redline. This huge difference could be due to Redline being a GUI framework and Volatility being a CLI based framework. During the experiment, Redline was running at all times once it was started. But Volatility on the other hand, kept terminating after each command was executed as shown in Fig 6. This meant that each time Volatility was run, it boots up, fetches the file specified, executes the plugin in relation to the file, gives us the output and then immediately terminates. This also means that each time Volatility is run, it has a different process ID (PID). By doing this, Volatility does not stay in the RAM for longer than it is executed, thus saving a lot of space for other processes.

Here are the comparisons of the frameworks in terms of CPU:

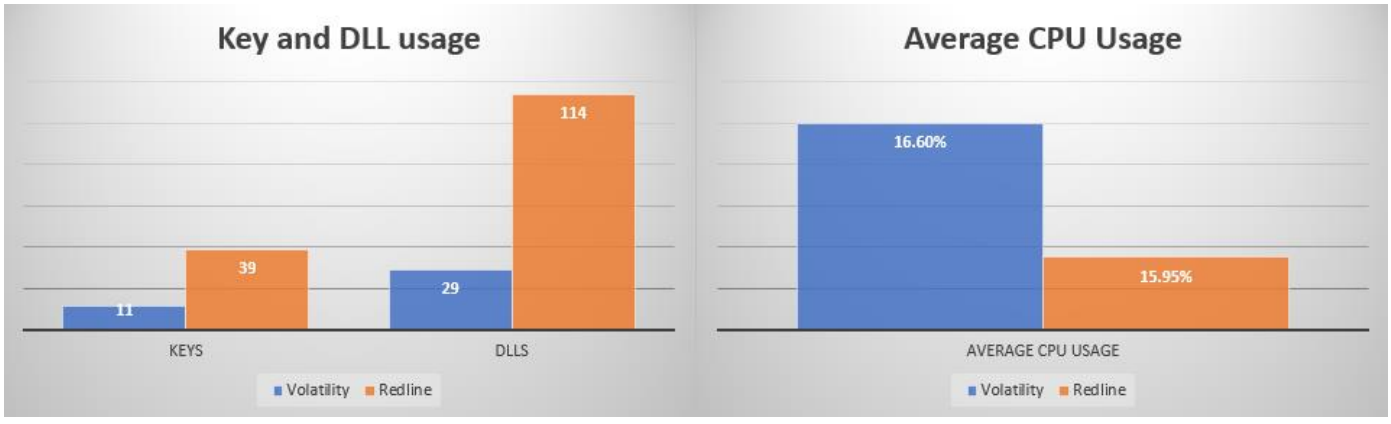

Fig 8: Comparison of CPU resources used by Volatility and Redline

Just like we looked at the difference between these two frameworks in case the RAM usage, we see a similar pattern when we look at the Key and DLL usage chart. Volatile uses 71.79% lesser keys and 74.56% lesser dlls in comparison to Redline. But when it comes to average CPU usage, volatility uses 0.65% more CPU than Redline. The reason behind this is because when a memory dump is opened with Redline, it already analyses the data, segregates it and is ready to just show it to you in just one click. But in the case of Volatility, because it keeps terminating its process every time it executes a command, it has boot up again, search through the memory dump file again according to the plugin and then output it. This takes up more CPU resource.

The research conducted by Daghmehchi Firoozjaei, M *et al*., set foot as the base paper for this research work. However, their paper did not cover any of the memory acquisition tools. It is important to note that as a part of memory forensics process, both the acquisition tools and the analysis frameworks serve great importance. The base paper compared three major tools, namely, Volatility, Autopsy and Redline. Though the inspiration for this paper was drawn from their research, autopsy does not really classify as a memory forensics tool. Autopsy is a very powerful tool for digital forensics and is great for analyzing disk images. But it hardly has any features for conducting memory Analysis. This leaves us with Volatility and Redline. Volatility as we know is one of the most poweful tools to conduct memory forensics. But Redline, however is an end point security tool by FireEye. The tool was not specifically built for memory analysis. But it still does support a bit of memory analysis but not as wide and powerful as Volatility.
However, there is another framework that is known for memory analysis called Rekall. Rekall originated from Volatality 2 and just like Volatility, is open source and powerful. This research was however limited to Volatility and Redline, unfortunately due to multiple failed attempts to install Rekall. A comparision between Rekall and Volatility would have been more appropriate considering both the tools are powerful, open source and CLI.
Daghmehchi Firoozjaei, M., *et al*(2022)

After all the critical analysis done above, we can say that among the Acquisition tools, DumpIt used up the least amount of resources even though it took up more time to capture memory than Belkasoft, Magnet and FTK Imager. But due to it being a CLI based application, it might throw some beginners off. Belkasoft RAM capturer does seem to be a close competitor to DumpIt and also has an upperhand by being GUI based. Since our aim for this research is to find the most efficient way for conducting memory forensics, it is only fair to go ahead with DumpIt as our chosen acquisition tool with the only sacrifice being GUI and a tiny bit of time.

As for our analysis framework, there does seem to be a clear winner when it comes to the least resources used. Though Volatility might be CLI based, it is very lightweight, extremely powerful and it used in real world scenarios by memory forensic analysts. As a beginner, there might be a tiny bit of hesitation or difficulty in understanding how cmd works but it is only a matter of time before any beginner can catch up with it. It is also important to note that on the other hand, Redline is an endpoint security framework and is not specifically built for memory forensics. This does not mean that Redline isnt a good enogh framework. Redline is one of the best frameworks when it comes to complete end point security monitoring and analysis. It is very widely used in Indicent Response and various other parts of digital forensics.

This leaves us with DumpIt and Volatility being the best combination to conduct memory forensics in the most efficient way.

5. Conclusion and Future Work

The aim of this research was to come up with an efficient way of conducting memory forensics. A comparative analysis was made between various tools which were split up into two categories. The first set of tools were the memory acquisition tools and the second set were the frameworks that could be used to conduct analysis on the memory dumps acquired by the previous set of tools. The experiments were also done keeping mind the main audience being beginner learners who are just entering the area of digital/memory forensics. Efficiency was taken into consideration in light that not everyone has high end system to develop their skills and thus the need to find lightweight, yet powerful tools was needed. Based on research conducted, it was concluded that the most efficient method of conducting memory forensics was to acquire a memory dump using DumpIt and analyze the image using Volatility. The sacrifices made in this method is that DumpIt is slower than other tools and both DumpIt and Volatility are CLI based. CLI being a sacrifice only because the target audience are expected to be beginners, having no much knowledge in the field.

The research though concluded here, can be evolved in various ways. The tools used in this research were only based on windows. However, tools based on Linux, MacOS and other available operating systems could open up greater possibilities for research. As mentioned in the discussions, another great tool that could be researched on for comparison is Rekall, which is also very powerful and command line based. Also with the ever growing development of technology, there will always be new tools and framework that comes up and research can be carried out on them too.

REFERENCES

Daghmehchi Firoozjaei, M., Habibi Lashkari, A. and Ghorbani, A. A. (2022) Journal of Cyber Security Technology, 6(3), pp. 149–173.

Himanshu, Bhatt, S., & Garg, G. (2021). Comparative analysis of acquisition methods in digital forensics. 2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT), 129-134.

Taylor, J., Turnbull, B.P., & Creech, G. (2018). Volatile Memory Forensics Acquisition Efficacy: A Comparative Study Towards Analysing Firmware-Based Rootkits. Proceedings of the 13th International Conference on Availability, Reliability and Security.

Hariharan, M., Thakar, A., & Sharma, P. (2022). Forensic Analysis of Private Mode Browsing Artifacts in Portable Web Browsers Using Memory Forensics. 2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS), 1-5.

Volatilityfoundation (2019) Memory Samples, GitHub. Available at: https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples (Accessed: 14 December 2023).

Volatilityfoundation (2020) volatilityfoundation/volatility: An advanced memory forensics framework, GitHub. Available at: https://github.com/volatilityfoundation/volatility (Accessed: 14 December 2023).

Martínez, Marcos. (2020). Impact of Tools on the Acquisition of RAM Memory. International Journal of Cyber Forensics and Advanced Threat Investigations. 1. 3-17. 10.46386/ijcfati.v1i1-3.12.

Faiz, Muhammad & Prabowo, Wahyu. (2018). Comparison of Acquisition Software for Digital Forensics Purposes. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control. 4. 10.22219/kinetik.v4i1.687.

David J. Dodd (2014) Forensic Analysis with Redline and Volatility. *Admin Network & Security.* Available at: https://admin-magazine.com/Archive/2014/21/Malware-Analysis [Accessed 14 December 2023].