Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 13, Issue. 8, August 2024, pg.16 – 22

Automated Windows-Based Timelining Tool for Memory and Disk Image Analysis: Leveraging Timsort Algorithm with WinPmem, FTK Imager, Volatility 3 and The Sleuth Kit

Aaron Lensmer M. Abdon[1]; Judilee Christian M. Ang[2]; Keinaz N. Domingo[3]; Jerome E. Gutierrez[4]

[1]Department of Computer Technology, De La Salle University, Philippines
[2]Department of Computer Technology, De La Salle University, Philippines
[3]Department of Computer Technology, De La Salle University, Philippines
[4]Department of Computer Technology, De La Salle University, Philippines

[1] aaron_lensmer_abdon@dlsu.edu.ph; [2] judilee_ang@dlsu.edu.ph;
[3] keinaz.domingo@dlsu.edu.ph; [4] jerome.gutierrez@dlsu.edu.ph

DOI: https://doi.org/10.47760/ijcsmc.2024.v13i08.002

Abstract— As technology evolves, so does the threat of cyberattacks – making Digital Forensics crucial for damage control and prevention. This paper aims to address the inefficiencies faced by investigators in a forensic setting by automating the processes necessary for disk and memory image acquisition, forensic artifact parsing, and timeline generation. Leveraging publicly available tools such as: WinPmem, FTK Imager, Volatility 3 (VOL3), and The Sleuth Kit (TSK), the developed Python script is then able to provide for a clearer insight into the series of events that have transpired during a cyber incident through the generation of detailed and cohesively organized timelines, then using the Timsort algorithm for timeline analysis.

Keywords— Image Acquisition, Forensic Artifact Parsing, Timeline Generation, Timsort Algorithm

I. INTRODUCTION

As the usage of technology grows in our modern age, the possibility and threat of cyberattacks grow together with it. The importance of Digital Forensics therefore cannot be understated as it helps with damage mitigation whilst providing useful insights through security measures to prevent attacks on computer systems [1]. Digital Forensics involves the identification, preservation, analysis, documentation, and presentation of digital evidence – all of which are crucial steps for uncovering the series of events that took place during cyber incidents [2].

It should be noted, however, that various researchers and experts in the field have felt that the number and complexity of investigations would inevitably rise in the future, giving rise to distinct potential challenges [3]. These include technical, law enforcement, personnel-related, and operational challenges [4].

Automation involves the use of a software or program to perform tasks that are usually done manually, enhancing the efficiency and accuracy of forensic investigations [5]. While there have been multiple studies tackling process automation to further simplify tedious work, most only focus on its importance and the contextualization of the challenges it brings about, rather than actively building the necessary tools for it.

As calls for the creation of advanced forensics tools for evidence collection are continuously made [3], this study aims to fill in this gap and develop a straightforward solution that would provide for the simplification of tasks for digital incident responders by combining image acquisition for both disk and memory aspects of digital forensics together with the timelining of their respective outputs.

Digital forensic investigations require the use of multiple tools for the collection and analysis of forensic data. This often involves manual intervention, leading to inefficiencies and potentially human-based errors.

A. *Design of the Study*

Throughout the research process, significant focus was placed on integrating multiple digital forensics tools to create a systematic approach in determining the series of events that transpired on a forensic target. More specifically, the study aims to:

i. Design a user-friendly script that provides an automated workflow for acquiring disk and memory images, parsing data, and generating timelines for further analysis.
ii. Integrate disk and memory image acquisition for forensic analysis by leveraging existing, publicly available image acquisition tools such as WinPmem and FTK Imager.
iii. Integrate disk and memory image parsing to extract the relevant artifacts necessary for timeline generation by leveraging existing, publicly available parsing and timeline generation tools such as The Sleuth Kit (TSK) and Volatility 3 (VOL3).

II. LITERATURE REVIEW

A. *Open-source Tools*

a) *FTK Imager*

Obtaining the disk image is a crucial part of disk forensics. FTK Imager serves as one of such tools used by experts in the field of cybersecurity in gathering and analyzing forensic evidence [6]. Its publicly available executable files provided by Exterro, after having acquired the rights from AccessData, features in-depth hard disk searching mechanism to reconstruct an image of hard disk drives [7]. Integrity verification is then facilitated by computing the MD5 and SHA1 checksums on any of its possible file format outputs: DD/raw, AD1, and E01.

While recent media installations of FTK Imager, as provided by AccessData, allows for the acquisition of both disk and memory images, these features are Graphical User Interface (GUI) restricted and are not partial to Command Line Interface (CLI) automation.

Looking back into previous version iterations, the v3.1.1 release of FTK Imager once offered support for command line functions but is, however, no longer publicly available for downloads after AccessData's change of domains following its acquisition by Exterro on the 3rd of December 2020 [8].

Utilizing the Wayback Machine, a digital archive of the world wide web, it becomes possible to recover the now lost domain with most of its features intact, including the download page for FTK Imager v.3.1.1 [9].

TABLE I
FTK IMAGER V.3.1.1 RELEVANT COMMAND LINE FUNCTIONS

Options	**Description**
--list-drives	Shows detected physical drives.
--verify	Verifies the destination image. Verifies the source image if no destination is specified.
--e01	Creates an E01 format image.
--s01	Creates a SMART ew-compressed image.

As detailed in its provided documentation file, FTK Imager v3.1.1 follows the usage format: ftkimager [source] [destination file] [options]. While it offers a plethora of other helpful functions in disk forensics, the above listed options are among those that are concerned with the automation of the disk imaging process. Incidentally, unlike the most recent versions with added support for memory imaging, FTK Imager v3.1.1 only supports disk imaging in either E01 or S01 file extensions.

b) *WinPmem*

To extend the support of the timeline generation features of the developed script to also include memory imaging, a different standalone executable file with CLI support is necessary. The v4.0 release of WinPmem implements a Fast IO mode and can perform memory imaging via the command line [10], outputting a .raw file format. It is a straightforward CLI based application following the usage format: winpmem [destination_file].

Despite its decline in popularity with the growing preference over GUI-based tools, WinPmem has been noted for its alternative imaging methods that are capable of circumventing anti-forensic attempts [11]. While it is limited only to Windows OS systems, its support for command line functions makes it an invaluable tool in the filling the role of a memory acquisition tool for workflow automation.

c) Volatility 3 (VOL3)

In contrast to Linux systems, Volatility 3 is not native on Windows OS machines. Primarily used for memory image analysis, VOL3 offers support for performing scans that target process objects, thread objects, network connections and sockets, as well as loaded DLLs – all of which, with their respective timestamps, contribute to a coherent consolidation of what transpired on the forensic target's memory dump [12].

TABLE II
VOLATILITY-3 (WINDOWS OS) RELEVANT COMMAND LINE FUNCTIONS

Plugins	**Description**
windows.psscan.PsScan	Scans for and lists processes structures in memory.
windows.thrdscan.ThrdScan	Scans for and lists thread structures in memory.
windows.netscan.NetScan	Scans for and lists network connections and sockets.
windows.dlllist.DllList	Scans for and lists loaded Dynamic Link Libraries (DDLs) by running processes.

d) The Sleuth Kit (TSK)

TSK offers functions for disk image analysis, offering means for the analysis, recovery, and collection of digital evidence [13]. Its plugins allow for examining the provided file system, recursively going through each file and directory, and appending their corresponding file activity [14]. While it has built-in timelining capabilities, it exports the data in a text file format, making it unsuitable for immediate integration with structured databases or for advanced analysis that requires more complex querying and filtering capabilities.

TABLE III
THE SLEUTH KIT (TSK) RELEVANT COMMAND LINE FUNCTIONS

Tools	**Description**
mmls.exe	Obtains the offset value for the NTFS filesystem.
fls.exe	Extracts metadata and lists file system contents, including deleted files.
mactime.pl	Creates a timeline of file activity based on MAC (modified, accessed, and changed) times.

III. TIMSORT ALGORITHM

Sorting algorithms are generally used in various fields to rearrange a collection of data following a given rule set [15]. In the context of programming, multiple different factors play pivotal roles in choosing the most appropriate algorithm to use depe[nding on the situation at hand. Determining which algorithm is the best fit for a given dataset proves to be an arduous task as analysts would have to consider multiple factors such as worst-case time complexity, memory usage, and specific application requirements to maximize its effectiveness and efficiency [16].

In Python, the primary sorting algorithm for the sort() function is what is known as Timsort. Developed by Tim Peters in 2002 specifically for the programming language, the algorithm integrates the principal functions of Merge and Insertion sort, both of which are designed to handle sorting through heaps and stacks of uncategorized information [16]. Studies have shown its superior effectiveness against classical sorting algorithms: Heap Sort, Shell Sort, Merge Sort, and Quick Sort. Despite the increase on each sorting algorithm's individual execution time as the data size increases, Timsort was shown to consistently produce the fastest execution time across all data sizes [17]. Data sets reaching up to over a million elements highlights Quick Sort and Timsort's efficiency among all the other tested sorting algorithms.

The rationale behind the implementation of the Timsort algorithm in the developed script lies in the unpredictability of the possible number of elements each image file might contain when parsed. Given that Timsort has been shown to have yielded promising results for both small and large datasets, it becomes imperative to use when rearranging extracted forensic artifacts based on their corresponding timestamps in reverse chronological order to provide a clearer picture of a cybersecurity incident for the forensic analyst in the most efficient way possible.

IV. PROPOSED WORK

An outline of the systematic approach for acquiring and analyzing forensic evidence is shown below. The proposed methodology is divided into three main workflows based on the type of data being analyzed. For memory and disk acquisition, WinPmem, and FTK imager are utilized, respectively.

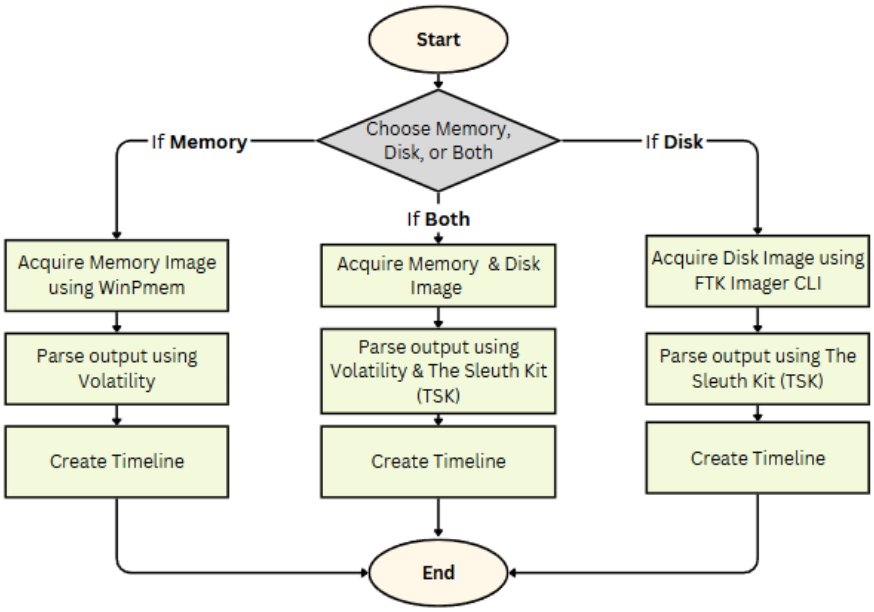


Fig. 1 Code execution flowchart for image acquisition and timeline generation

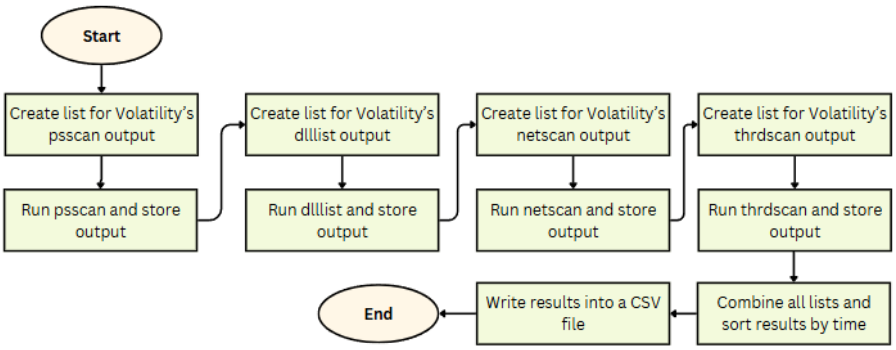


Fig. 2 Workflow process for the parsing of memory artifacts using VOL3 on the generated .raw file by WinPmem

The acquired memory image from WinPmem will be parsed using Volatility 3. VOL3 provides various plugins for extracting information, complete with their corresponding timestamps for running processes, threads, inbound and outbound network connections, and loaded DLLs — all of which are to be collected and used to create a comprehensive timeline of events.

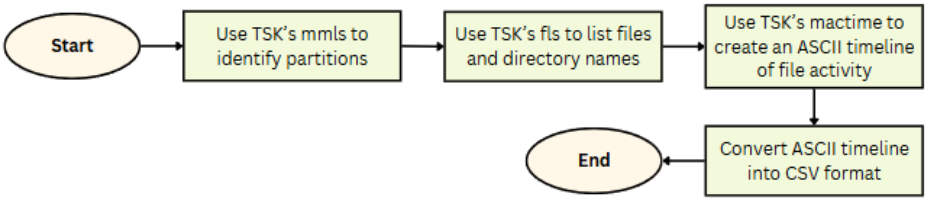


Fig. 3 Workflow process for the parsing of disk artifacts using TSK on the generated. E01 file by FTK Imager

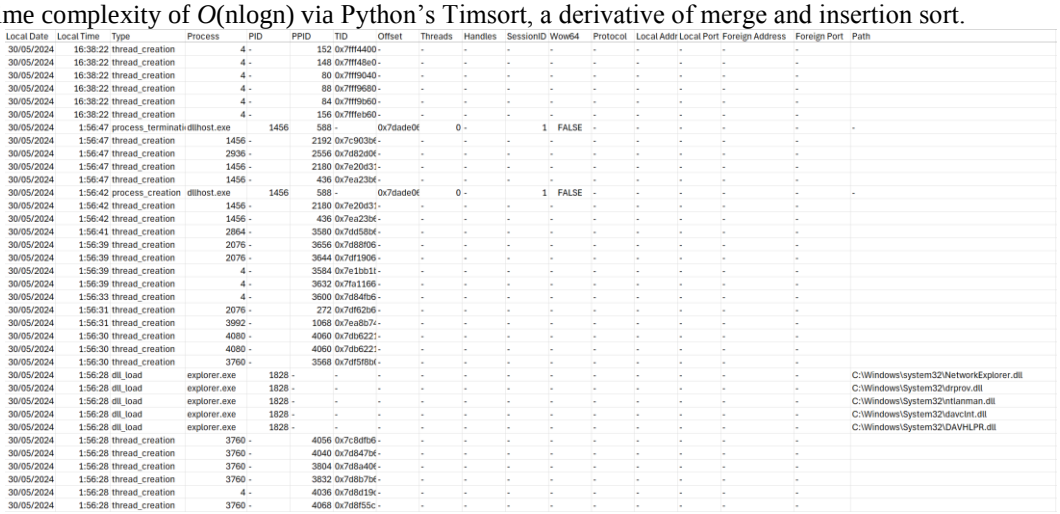
On the other hand, the acquired disk image from FTK Imager is instead to be analyzed using TSK's different tools. This process will extract file system activity logs, converting the exported list into a CSV format for further analysis.

V. RESULT AND DISCUSSION

The study involves developing a Python script to automate the acquisition of either the memory or the disk image, or both depending on user preference – from which forensic artifacts will be extracted and processed to generate a timeline in a .csv file format.

A. *Memory Forensics Timelining*

Utilizing command-line functions to automate scanning for processes, threads, network connections, and loaded DLLs, each individual event is then saved into a list and eventually sorted in reverse chronological order with a time complexity of *O*(nlogn) via Python's Timsort, a derivative of merge and insertion sort.



Local Date	Local Time	Type	Process	PID	PPID	TID	Offset	Threads	Handles	SessionID	Wow64	Protocol	Local Addr	Local Port	Foreign Address	Foreign Port	Path
30/05/2024	16:38:22	thread_creation	-	4	-	152	0x7fff4400	-	-	-	-	-	-	-	-	-	
30/05/2024	16:38:22	thread_creation	-	4	-	148	0x7fff48e0	-	-	-	-	-	-	-	-	-	
30/05/2024	16:38:22	thread_creation	-	4	-	80	0x7fff9040	-	-	-	-	-	-	-	-	-	
30/05/2024	16:38:22	thread_creation	-	4	-	88	0x7fff9680	-	-	-	-	-	-	-	-	-	
30/05/2024	16:38:22	thread_creation	-	4	-	84	0x7fff9b60	-	-	-	-	-	-	-	-	-	
30/05/2024	16:38:22	thread_creation	-	4	-	156	0x7fffeb60	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:47	process_terminati	dllhost.exe	1456	588	-	0x7dade0(	0	-	1	FALSE	-	-	-	-	-	-
30/05/2024	1:56:47	thread_creation	-	1456	-	2192	0x7c903b(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:47	thread_creation	-	2936	-	2556	0x7d82d0(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:47	thread_creation	-	1456	-	2180	0x7e20d3(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:47	thread_creation	-	1456	-	436	0x7ea23b(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:42	process_creation	dllhost.exe	1456	588	-	0x7dade0(	0	-	1	FALSE	-	-	-	-	-	-
30/05/2024	1:56:42	thread_creation	-	1456	-	2180	0x7e20d3(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:42	thread_creation	-	1456	-	436	0x7ea23b(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:41	thread_creation	-	2864	-	3580	0x7dd58b(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:39	thread_creation	-	2076	-	3656	0x7d88f06	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:39	thread_creation	-	2076	-	3644	0x7df1906	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:39	thread_creation	-	4	-	3584	0x7e1bb1(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:39	thread_creation	-	4	-	3632	0x7fa1166	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:33	thread_creation	-	4	-	3600	0x7d84fb6	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:31	thread_creation	-	2076	-	272	0x7df62b6	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:31	thread_creation	-	3992	-	1068	0x7ea8b7(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:30	thread_creation	-	4080	-	4060	0x7db622(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:30	thread_creation	-	4080	-	4060	0x7db622(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:30	thread_creation	-	3760	-	3568	0x7df5f8b(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:28	dll_load	explorer.exe	1828	-	-	-	-	-	-	-	-	-	-	-	-	C:\Windows\system32\NetworkExplorer.dll
30/05/2024	1:56:28	dll_load	explorer.exe	1828	-	-	-	-	-	-	-	-	-	-	-	-	C:\Windows\System32\drprov.dll
30/05/2024	1:56:28	dll_load	explorer.exe	1828	-	-	-	-	-	-	-	-	-	-	-	-	C:\Windows\System32\ntlanman.dll
30/05/2024	1:56:28	dll_load	explorer.exe	1828	-	-	-	-	-	-	-	-	-	-	-	-	C:\Windows\System32\davclnt.dll
30/05/2024	1:56:28	dll_load	explorer.exe	1828	-	-	-	-	-	-	-	-	-	-	-	-	C:\Windows\System32\DAVHLPR.dll
30/05/2024	1:56:28	thread_creation	-	3760	-	4056	0x7c8dfb6	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:28	thread_creation	-	3760	-	4040	0x7d847bf	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:28	thread_creation	-	3760	-	3804	0x7d8a40(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:28	thread_creation	-	3760	-	3832	0x7d8b7bf	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:28	thread_creation	-	4	-	4036	0x7d8d19(	-	-	-	-	-	-	-	-	-	
30/05/2024	1:56:28	thread_creation	-	3760	-	4068	0x7d8f55c	-	-	-	-	-	-	-	-	-	

Fig. 4 Sample timeline in .csv file format performed on a Windows 7 memory image (.raw)

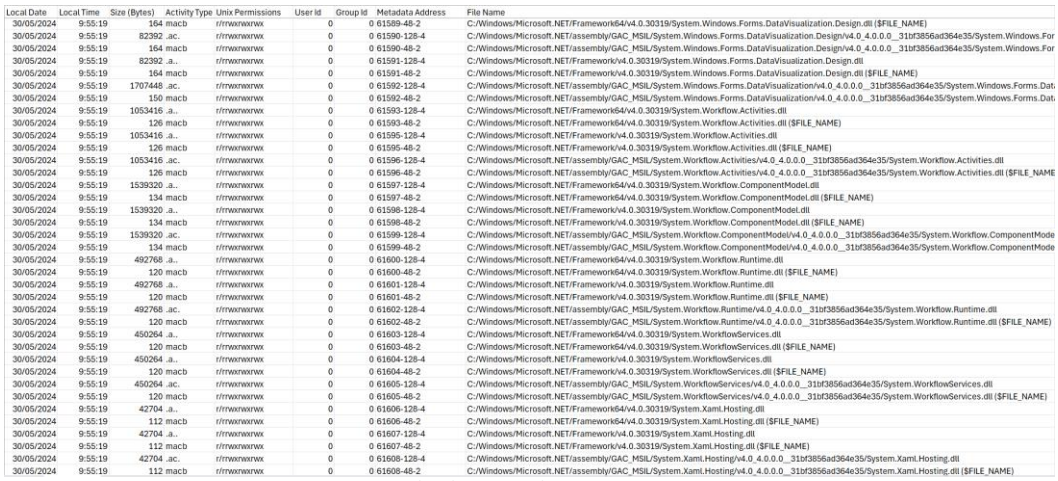
TABLE IV
ACTIVITY TYPE FOR VOL3 OUTPUT

Type	**Description**
process_creation	Denotes an event where a process was executed in the system.
process_termination	Denotes an event where a process was terminated in the system.
thread_creation	Denotes an event where a process called for the creation of a new thread in the system.
network_connection	Denotes an event where a connection is established or closed between the local machine and a remote machine over a network.
dll_load	Denotes an event where a DLL is loaded into the memory of a process.

The events listed in the generated timeline are sorted depending on their activity type. Relevant columns are filled with corresponding information regarding the event details.

B. *Disk Forensics Timelining*

Disk forensics instead utilizes fls.exe to recursively go through the file system and extract their corresponding metadata. The handling of the timeline generation falls on mactime.pl, with the above-mentioned sorting algorithm also applying to the timelining process for disk forensics.



Local Date	Local Time	Size (Bytes)	Activity Type	Unix Permissions	User Id	Group Id	Metadata Address	File Name
30/05/2024	9:55:19	164	macb	r/rrwxrwxrwx	0	0	61589-48-2	C:/Windows/Microsoft.NET/Framework64/v4.0.30319/System.Windows.Forms.DataVisualization.Design.dll ($FILE_NAME)
30/05/2024	9:55:19	82392	.ac.	r/rrwxrwxrwx	0	0	61590-128-4	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Windows.Forms.DataVisualization.Design/v4.0_4.0.0.0__31bf3856ad364e35/System.Windows.For
30/05/2024	9:55:19	164	macb	r/rrwxrwxrwx	0	0	61590-48-2	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Windows.Forms.DataVisualization.Design/v4.0_4.0.0.0__31bf3856ad364e35/System.Windows.For
30/05/2024	9:55:19	82392	.a..	r/rrwxrwxrwx	0	0	61591-128-4	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.Windows.Forms.DataVisualization.Design.dll
30/05/2024	9:55:19	164	macb	r/rrwxrwxrwx	0	0	61591-48-2	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.Windows.Forms.DataVisualization.Design.dll ($FILE_NAME)
30/05/2024	9:55:19	1707448	.ac.	r/rrwxrwxrwx	0	0	61592-128-4	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Windows.Forms.DataVisualization/v4.0_4.0.0.0__31bf3856ad364e35/System.Windows.Forms.Data
30/05/2024	9:55:19	150	macb	r/rrwxrwxrwx	0	0	61592-48-2	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Windows.Forms.DataVisualization/v4.0_4.0.0.0__31bf3856ad364e35/System.Windows.Forms.Data
30/05/2024	9:55:19	1053416	.a..	r/rrwxrwxrwx	0	0	61593-128-4	C:/Windows/Microsoft.NET/Framework64/v4.0.30319/System.Workflow.Activities.dll
30/05/2024	9:55:19	126	macb	r/rrwxrwxrwx	0	0	61593-48-2	C:/Windows/Microsoft.NET/Framework64/v4.0.30319/System.Workflow.Activities.dll ($FILE_NAME)
30/05/2024	9:55:19	1053416	.a..	r/rrwxrwxrwx	0	0	61595-128-4	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.Workflow.Activities.dll
30/05/2024	9:55:19	126	macb	r/rrwxrwxrwx	0	0	61595-48-2	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.Workflow.Activities.dll ($FILE_NAME)
30/05/2024	9:55:19	1053416	.ac.	r/rrwxrwxrwx	0	0	61596-128-4	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Workflow.Activities/v4.0_4.0.0.0__31bf3856ad364e35/System.Workflow.Activities.dll
30/05/2024	9:55:19	126	macb	r/rrwxrwxrwx	0	0	61596-48-2	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Workflow.Activities/v4.0_4.0.0.0__31bf3856ad364e35/System.Workflow.Activities.dll ($FILE_NAME
30/05/2024	9:55:19	1539320	.a..	r/rrwxrwxrwx	0	0	61597-128-4	C:/Windows/Microsoft.NET/Framework64/v4.0.30319/System.Workflow.ComponentModel.dll
30/05/2024	9:55:19	134	macb	r/rrwxrwxrwx	0	0	61597-48-2	C:/Windows/Microsoft.NET/Framework64/v4.0.30319/System.Workflow.ComponentModel.dll ($FILE_NAME)
30/05/2024	9:55:19	1539320	.a..	r/rrwxrwxrwx	0	0	61598-128-4	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.Workflow.ComponentModel.dll
30/05/2024	9:55:19	134	macb	r/rrwxrwxrwx	0	0	61598-48-2	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.Workflow.ComponentModel.dll ($FILE_NAME)
30/05/2024	9:55:19	1539320	.ac.	r/rrwxrwxrwx	0	0	61599-128-4	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Workflow.ComponentModel/v4.0_4.0.0.0__31bf3856ad364e35/System.Workflow.ComponentMode
30/05/2024	9:55:19	134	macb	r/rrwxrwxrwx	0	0	61599-48-2	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Workflow.ComponentModel/v4.0_4.0.0.0__31bf3856ad364e35/System.Workflow.ComponentMode
30/05/2024	9:55:19	492768	.a..	r/rrwxrwxrwx	0	0	61600-128-4	C:/Windows/Microsoft.NET/Framework64/v4.0.30319/System.Workflow.Runtime.dll
30/05/2024	9:55:19	120	macb	r/rrwxrwxrwx	0	0	61600-48-2	C:/Windows/Microsoft.NET/Framework64/v4.0.30319/System.Workflow.Runtime.dll ($FILE_NAME)
30/05/2024	9:55:19	492768	.a..	r/rrwxrwxrwx	0	0	61601-128-4	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.Workflow.Runtime.dll
30/05/2024	9:55:19	120	macb	r/rrwxrwxrwx	0	0	61601-48-2	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.Workflow.Runtime.dll ($FILE_NAME)
30/05/2024	9:55:19	492768	.ac.	r/rrwxrwxrwx	0	0	61602-128-4	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Workflow.Runtime/v4.0_4.0.0.0__31bf3856ad364e35/System.Workflow.Runtime.dll
30/05/2024	9:55:19	120	macb	r/rrwxrwxrwx	0	0	61602-48-2	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Workflow.Runtime/v4.0_4.0.0.0__31bf3856ad364e35/System.Workflow.Runtime.dll ($FILE_NAME)
30/05/2024	9:55:19	450264	.a..	r/rrwxrwxrwx	0	0	61603-128-4	C:/Windows/Microsoft.NET/Framework64/v4.0.30319/System.WorkflowServices.dll
30/05/2024	9:55:19	120	macb	r/rrwxrwxrwx	0	0	61603-48-2	C:/Windows/Microsoft.NET/Framework64/v4.0.30319/System.WorkflowServices.dll ($FILE_NAME)
30/05/2024	9:55:19	450264	.a..	r/rrwxrwxrwx	0	0	61604-128-4	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.WorkflowServices.dll
30/05/2024	9:55:19	120	macb	r/rrwxrwxrwx	0	0	61604-48-2	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.WorkflowServices.dll ($FILE_NAME)
30/05/2024	9:55:19	450264	.ac.	r/rrwxrwxrwx	0	0	61605-128-4	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.WorkflowServices/v4.0_4.0.0.0__31bf3856ad364e35/System.WorkflowServices.dll
30/05/2024	9:55:19	120	macb	r/rrwxrwxrwx	0	0	61605-48-2	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.WorkflowServices/v4.0_4.0.0.0__31bf3856ad364e35/System.WorkflowServices.dll ($FILE_NAME)
30/05/2024	9:55:19	42704	.a..	r/rrwxrwxrwx	0	0	61606-128-4	C:/Windows/Microsoft.NET/Framework64/v4.0.30319/System.Xaml.Hosting.dll
30/05/2024	9:55:19	112	macb	r/rrwxrwxrwx	0	0	61606-48-2	C:/Windows/Microsoft.NET/Framework64/v4.0.30319/System.Xaml.Hosting.dll ($FILE_NAME)
30/05/2024	9:55:19	42704	.a..	r/rrwxrwxrwx	0	0	61607-128-4	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.Xaml.Hosting.dll
30/05/2024	9:55:19	112	macb	r/rrwxrwxrwx	0	0	61607-48-2	C:/Windows/Microsoft.NET/Framework/v4.0.30319/System.Xaml.Hosting.dll ($FILE_NAME)
30/05/2024	9:55:19	42704	.ac.	r/rrwxrwxrwx	0	0	61608-128-4	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Xaml.Hosting/v4.0_4.0.0.0__31bf3856ad364e35/System.Xaml.Hosting.dll
30/05/2024	9:55:19	112	macb	r/rrwxrwxrwx	0	0	61608-48-2	C:/Windows/Microsoft.NET/assembly/GAC_MSIL/System.Xaml.Hosting/v4.0_4.0.0.0__31bf3856ad364e35/System.Xaml.Hosting.dll ($FILE_NAME)

Fig. 5 Sample timeline in .csv file format performed on a Windows 7 disk image (. E01)

For reference, the activity type for each file in column 4 in Fig 2. pertains to the following:

TABLE V
ACTIVITY TYPE FOR MACTIME.PL OUTPUT

File System	m	a	c	b
Ext4	Modified	Accessed	Changed	Created
Ext2/3	Modified	Accessed	Changed	N/A
FAT	Written	Accessed	N/A	Created
NTFS	File Modified	Accessed	MFT Modified	Created
USF	Modified	Accessed	Changed	N/A

Since disk forensics is primarily concerned with NTFS filesystems, the corresponding designations for the MACB timestamp types are as shown in the 3rd row.

VI. CONCLUSIONS AND RECOMMENDATIONS

The paper was determined to show the feasibility and corresponding benefits of automating key processes in digital forensics, specifically in disk and memory image acquisition, and timeline generation. By leveraging tools such as WinPmem and VOL3 for memory forensics and FTK Imager and TSK for disk forensics, results have shown that automation can enhance the efficiency and reliability of forensic investigations.

Since a memory image is only capable of capturing artifacts for currently running tasks, combining it with the artifacts obtained from a disk image will give the incident responder a clearer picture of what happened in the event of an attack. In the disk image timeline, files that were created or modified can be seen. This in tandem with the generated memory timeline provides a better view of the series of activities that transpired on the forensic target.

Further areas to consider for improvement include extending the support to other operating systems such as Mac and Linux. Researchers may also opt to perform comparative analysis of the automated system against manually obtained data from image parsing tools.

ACKNOWLEDGEMENT

The researchers wish to acknowledge their De La Salle University Digital Forensics professor Mr. Jerome Gutierrez, and their former professor Mr. Keinaz Domingo, for their invaluable guidance and support throughout this research endeavour.

REFERENCES

[1]. A. Dimitriadis, N. Ivezic, B. Kulvatunyou and I. Mavridis, "D4I - Digital forensics framework for reviewing and investigating cyber-attacks," Array, vol. 5, 2020.

[2]. R. Fazzone, "Digital Forensics Fundamentals: Successful Preservation of Evidence," FTI Consulting Technology, [online]. Available: https://www.ftitechnology.com/resources/blog/digital-forensics-fundamentals-successful-preservation-of-evidence. [Accessed 12 July 2024].

[3]. A. R. Javed, W. Ahmed, and M. Alazab, "A Comprehensive Survey on Computer Forensics: State-of-the-Art, Tools, Techniques, Challenges, and Future Directions," IEEE, vol. 10, pp. 11065-11089, 2022.

[4]. M. Al Fahdi, N. L. Clarke, and S. M. Furnell, "Challenges to digital forensics: A survey of researchers & practitioners' attitudes and opinions," in Information Security for South Africa, Johannesburg, 2013.

[5]. C. Anglano, M. Canonico and M. Guazzone, "The Android Forensics Automator (AnForA): A tool for the Automated Forensic Analysis of Android Applications," Computers & Security, vol. 88, 2020.

[6]. Alshammari, A. (2023). Detection and investigation model for the hard disk drive attacks using FTK imager. International Journal of Advanced Computer Science and Applications, 14(7). https://doi.org/10.14569/IJACSA.2023.0140784

[7]. H. Tara and A. Mishra, "A Comparative Study of Digital Forensic Tools for Data Extraction From Electronic Devices," *Journal of Punjab Academy of Forensic Medicine & Toxicology*, vol. 21, no. 1, pp. 97–104, 2021, doi: 10.5958/0974-083X.2021.00016.9.

[8]. Exterro, "Exterro Acquires AccessData to Form the Leading Enterprise Legal GRC Software Platform Across Data Privacy, Forensics, and E-Discovery," Exterro, 15-Jul-2021. [Online]. Available: https://www.exterro.com/about/news/exterro-acquires-accessdata-to-form-the-leading-enterprise-legal-grc-software-platform-across-data-privacy-forensics-and-e-discovery. [Accessed 6 July 2024].

[9]. AccessData, "AccessData - Digital Forensics," AccessData, 18-Feb-2015. [Online]. Available: https://web.archive.org/web/20150218091741/http://accessdata.com/product-download/digital-forensics/windows-32bit-3.1.1. [Accessed 29 July 2024]

[10]. Velocidex, "WinPmem," GitHub, [online]. Available: https://github.com/Velocidex/WinPmem.

[11]. J.-N. Hilgert, M. Lambertz, and D. Plohmann, "Extending The Sleuth Kit and its underlying model for pooled storage file system forensic analysis," *Digital Investigation*, vol. 22, pp. S76–S85, 2017, doi: 10.1016/j.diin.2017.06.003.

[12]. Volatility Foundation, "Volatility3," GitHub, [online]. Available: https://github.com/volatilityfoundation/volatility3. [Accessed 6 July 2024]

[13]. J.-N. Hilgert, M. Lambertz, and D. Plohmann, "Extending The Sleuth Kit and its underlying model for pooled storage file system forensic analysis," *Digital Investigation*, vol. 22, pp. S76–S85, 2017, doi: 10.1016/j.diin.2017.06.003.

[14]. The Sleuth Kit, "TSK Tool Overview," The Sleuth Kit Wiki, [online]. Available: https://wiki.sleuthkit.org/index.php?title=TSK_Tool_Overview. [Accessed 6 July 2024]

[15]. A. Kusumaningrum, "Perbandingan Kecepatan Antara Selection Sort, Insertion Sort, dan Bubble Sort," *Teknomatika: Jurnal Informatika Dan Komputer*, vol. 3, no. 1, pp. 63–70, 2020, [Online]. Available: https://ejournal.unjaya.ac.id/index.php/teknomatika/article/view/363.

[16]. N. Auger, V. Jugé, C. Nicaud, and C. Pivoteau, "On the Worst-Case Complexity of TimSort," presented at the *26th Annual European Symposium on Algorithms (ESA 2018)*, France, 2018, doi: 10.48550/arXiv.1805.08612.

[17]. F. R. Wibowo and M. Faisal, "Comparative analysis of sorting algorithms: TimSort Python and classical sorting methods," *JISA (Jurnal Informatika Dan Sains)*, vol. 7, no. 1, pp. 11–18, 2024, doi: 10.31326/jisa.v7i1.1785.

