Article can be accessed online at http: //www.publishingindia.com

A Novel Memory Forensics Technique for Windows 10

Azad Singh*, Pankaj Sharma, Sakshi Sharma*****

Abstract

Volatile memory forensics, henceforth referred to as memory forensics, is a subset of digital forensics, which deals with the preservation of the contents of memory of a computing device and the subsequent examination of that memory. The memory of a system typically contains useful runtime information. Such memories are volatile, causing the contents of memory to rapidly decay once no longer supplied with power. Using memory forensic techniques, it is possible to extract an image of the system's memory while it is still running, creating a copy that can be examined at a later point in time, even after the system has been turned off and the data contained within the original RAM has dissipated. This paper describe the implementation of the technique that collect volatile artifacts extracted from the RAM dump and Hibernation file of Windows 10 operating system and shows the extracted data of various process of the system.

Keyword : Windows Forensics, Memory Forensics, Volatile Data, Volatile Digital Evidence

Introduction

The use of memory forensics allows the creation of a snapshot of a system at a particular point in time, known as a memory image. Memory typically contains that information which is never written to disk. Memory forensic allows the extraction of various types of forensically significant information that would have been disappeared when the system was turned off. Such information can include running processes, network connections, passwords, encryption keys and copies of files that are encrypted on disk. Memory forensics aims at gathering evidence from systems using different operations and techniques related to primary memory content. Aljaedi et al. [19] described the volatile memory techniques, due to rapid increase in memory size, the forensic investigators strongly recommend the live response approach for acquisition of volatile evidence. Through this technique, the investigator can collect not only the information about live processes but also about the finished and cache processes. However, there are some legal issues about memory forensics which include inconsistency of results. If an independent investigator produces certain analysis result then after sometime another investigator may come up with bit different results. So, the issue of inconsistency of analysis is inherently linked to the live memory forensics [1]. On the other side of volatile analysis, there is static analysis paradigm, where Windows registry and archival data from hard disk drive plays an important role and forensic scientists obtain valuable information from that and use it for extracting evidences [2]. Windows is the most dominating operating system in the world, therefore, forensic investigators usually encountered Windows system. This paper describe the implementation of the technique that collect volatile artifacts extracted from the RAM dump and Hibernation file of Windows 10 installation disk [3]. The integrity of results has been verified by comparing the findings of live RAM captured and from the Hibernation file.

* M.tech student, Department of Computer science and Applications, Kurukshetra University, Kurukshetra, Haryana, India. Email: azadmehla@kuk.ac.in

** M.tech student, Department of Computer science and Applications, Kurukshetra University, Kurukshetra, Haryana, India. Email: pankajshastri@kuk.ac.in

*** M.tech student, Department of Computer science and Applications, Kurukshetra University, Kurukshetra, Haryana, India. Email: sakshi28sept@gmail.com

Related Work

Bolagh et al., [18] proposed a technique to recover the decoding keys from the dump of the live image of a volatile memory. His proposed approach worked on Windows and Linux with True Crypt, which was a free open source tool that performed on-the-fly disk encryption. The authors also suggested a method to decrease the size of dump image. Mainly when True Crypt was used for encryption, the size could be bounded to 1-2 MB only. However, the suggested technique bore a limitation that the image should be present nearby for forensic analysis.

Wang et al., [4] proposed a model for live analysis by breaking it into three different stages such as evidence gathering, examining it, analyzing it and finally generating a report. That model was based on the physical memory or memory image. Live analysis, which was based on the physical memory, took place on the running system. But in their model an external media toolkit was linked to the system which potentially changed the behavior of the target system as well as it would be difficult to verify the credibility of evidence.

Kristine et al. [5] covered the theory behind volatile memory analysis, which included why memory forensic was important, what kinds of data could be recovered from memory, various potential pitfalls of memory analysis, techniques for recovering and analyzing volatile data. The aim of the author was to make an argument for adding analysis of physical memory to the toolkit of the forensic analyst and to describe some of the techniques and tools that could be used by analyst to analyze the physical memory.

Rahman et al. [1] author considered different techniques of live analysis and critically reviewed them by identifying their benefits and limitations. The key areas focused in that study were to virtualization, page file extraction, hibernation file extraction and identifying the encryption keys. However, their paper just gave a theoretical background and did not describe any specific technique for extraction of artifacts from hibernation and page file of the system, as those file were in encoded and compressed format, so no one can directly extract evidences from those and moreover Windows didn't give permission to directly extract them.

Gianni et al. [6] conducted live forensic analysis on two virtual machine having two different operating system Windows XP and Windows 7. The authors focused only some general applications such as Google Talk, Skype and the Internet Explorer browser. They used FTK imager software for the acquisition of dump memory and saved the dump memory image into connected USB device. The purpose of their work was to figure out the differences between window XP and window 7 during live forensic analysis.

Hang et al. [7] proposed a solution for collecting the digital evidence; they developed a live USB/DVD. During analysis if the target system was alive they format a script program and saved it in simple USB device and through that script they collected the required information from the memory dump and stored that output files into USB device (live analysis). If the target system was turned off they reboot the system by live USB/DVD and produced an image file of the memory. Live USB/DVD had some tool to produce the image of the disk such as AIR, image file producer etc. The proposed system had its own self developed script stored in USB and they assumed the target system had Linux operating system installed, the proposed system also shown other information about target system such as current processes, network ports, freshly executed commands, information about kernel, date time and host name etc. The proposed system also gave a graphical user interface such as Xdialog to show the result of the target system analysis.

Dija et al., [8] had investigated the extraction of volatile artifacts from the hibernation file. Their work concentrated on application programs that could be searched for the hibernation file; however, it did not consider Windows apps and their associated page and swap files. Mrdovic and Huseinovic [9] recovered encryption keys from a physical memory dump with a hibernation file as the source. Gupta and Mehte [10] conducted similar work on the Sandboxed application in a non-encrypted system.

Iqbal et al., [11] investigated the Surface tablet with a standard Windows 8 RT installation. However, their work was limited to Windows 8 RT and covered only the Surface tablet, which, compared with Windows 8.x, was restricted in terms of its hardware and operating system functionality.

Carvey [12] described the procedures that should be followed during an incident response involving a Windows 8.0 device. He also identified Windows 8

artifacts that could be extracted. Additionally, he stressed the importance of the hibernation file and the fact that malware and other data could remain resident in the file even after anti-virus software and other temporary space cleaners had been employed.

Quick and Choo [13] [14] discussed methods for extracting artifacts related to the use of Sky Drive, Dropbox and Google Drive. The remnants gathered include user-related information and files. Quick and Choo [15] also analyzed RAM captures and local sources for the Dropbox application, but they did not consider the fact that the hibernation file might contain sensitive data related to the application.

Background

This section has provided an overview of the key memory management mechanisms and data structures that are of interest in memory forensics investigations. Such an understanding of the common data structures utilize by Windows operating systems to manage the execution of processes is an important part of memory analysis. At the hardware level, memory storage is organized into pages. The operating system maintains a page directory that stores Page Directory Entry (PDE) pointers. PDE pointers occupies 4 bytes each and the entry may contain up to 1024 unique links to page tables. Page tables, in turn, store up to 1024 pointers to Page Table Entry (PTE) data structures that correspond to a page in memory. The Memory Management Unit (MMU) first recovers the base address of the page directory, which is stored in the CR3 register of the system's processor [16]. The first 10 bits of a virtual address can then be used as an index into the page directory to retrieve the desired PDE. The PDE in conjunction with the Page Table Index (PTI), which is the next 10 bits of the virtual address, are used to identify the appropriate page table and PTE. Finally, the appropriate page in RAM is determined by following the link in the PTE. Processes are represented in memory by an EPROCESS data structure. The EPROCESS block contains many process attributes, as well as pointers to a number of related data structures. For example, a process has one or more threads represented by an ETHREAD block. The EPROCESS block and the data structures it contains reside within the system address space. Another key structure, the Process Environment Block (PEB), exists in process address space. The PEB contains a process's global contextual information such as startup parameters or synchronization objects. Each ETHREAD block has a pointer to a KTHREAD block. The KTHREAD block has a pointer to the Thread Environment Block (TEB) which maintains user-mode contextual information about threads [16]. The Windows operating system records all of the metadata necessary to manage the processes currently being executed in physical memory. Even if a process exits its metadata may be retained in memory for weeks. The information stored in the metadata provides a snapshot of the processes and threads that are either currently or recently executed on a system [17].

Problem Formulation

The analysis of any volatile memory captured by an incident responder is currently lesser precise than the analysis of hard disk. A hard disk has a specified strict pre-defined structure and analyst know where to look for certain structure and data types on a specific kind of file system. Memory on the other hand can be allocated or de-allocated to different areas depending on what memory is already being used. It is impossible to predict what you find in volatile memory or where it is stored. It is really apparent that acquiring and analyzing this type of data is more challenging and perilous than the analysis of secondary memory. This paper proposes a technique to extract the volatile evidences from captured RAM and Hibernation file of Windows 10. Windows 10 incorporates many new features like notification center, new start menu, frequent folders, cortana, Windows 10 native apps , new browser names Microsoft edge that have forensic relevant artifacts. There are no prior techniques reported in the literature to collect volatile evidences from Windows 10.

5. Proposed Approach

To address the problems mentioned in the last section, a novel technique is proposed to collect the static and volatile evidences in this section. Static data is present on the disk in the form of databases and registry files while some artifacts are present only in the volatile memory such as login passwords, encryption keys, visited websites, chats, notifications, send or received emails, running processes, IP addresses etc. For this reason a complete forensic picture will be available if an investigator consider both static and volatile artifacts and get a timeline of the activities performed by the system. Tools such as Privacy Eraser and other Cache cleaner such as Modern UI Apps Cleaner

enable users to clean cache, cookies, Internet history files, and other temporary folders. This cleaning results in the loss of potential static evidence from the disk so volatile memory is the only option to collect and analyze the evidences. The proposed technique in the form of flow chart is shown in the Figure 1. The approach follows the both steps of memory forensic i.e. memory acquisition and memory analysis. In memory acquisition phase the RAM dump and hibernation file are collected using Moonsols toolkit and in memory analysis phase Volatility tool and Winhex tool are used to analyze the physical memory of Windows 10. The Windows operating system records all of the metadata necessary to manage the processes currently being executed in physical memory. Even if a process exits, its metadata may be retained in memory for weeks. The information stored in the metadata provides a snapshot of the processes and threads that are either currently or have recently executed on a system. As such an understanding of the common data structures utilize by Windows operating systems to manage the execution of processes is an important part of memory analysis. The proposed technique works well for all the scenarios including live system, dead system scenarios and user provided image which include the dd image, EWF image or image or hard disk from any other tool. The proposed technique have assumed that the system to be analyzed has active hibernation mode, therefore one can extract the hibernation file in all the scenarios. Extraction of Hibernation file is more important in case of dead system as in such cases Hibernation file is the only source to have an archival image of memory. The precondition for the better results in such case is that hibernation file is not too much older because then analyst cannot extract the current image of the memory. Along with hibernation file one can also extract page file and system file which can helpful in validation of collected evidences, however in this research only the hibernation file is collected.

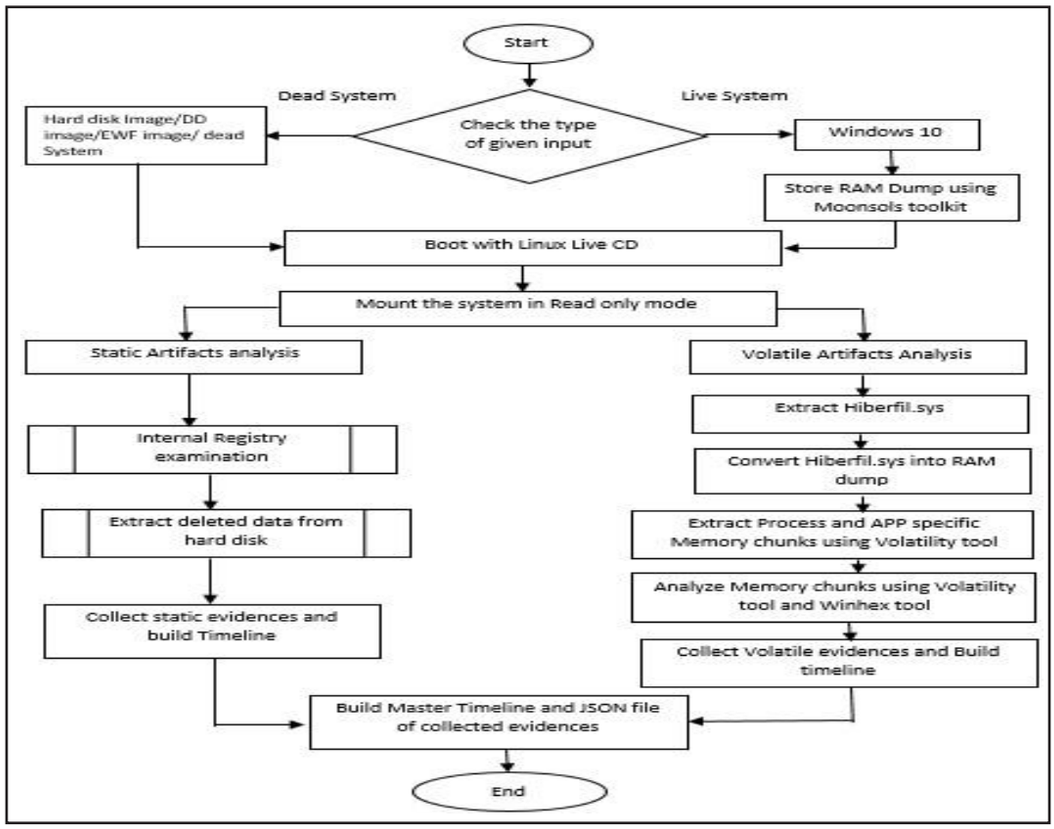


Figure 1: Proposed Technique for Extracting Evidences

Hibernation file is converted into raw dump using Moonsols toolkit, however Volatility tool can also be used for this but Moonsols is better than Volatility tool because it gives error free and faster results. The proposed technique collects the hibernation file from live system. The proposed technique is shown in Figure 1 in the form of flowchart. The steps of the flowchart are explained below:

a. If system is live then store RAM dump of system using Moonsols Toolkit and boot it with Linux Live CD.
b. Else user provides input in the form of hard disk image or DD image or EWF image, or dead system, boot it with Linux Live CD.
c. Mount the system in read only mode and invoke the static and volatile analysis process.
d. The internal registry and extract deleted data from hard drive by using static analysis.
e. Collect static evidences and build a timeline.
f. Extract hiberfil.sys by using volatile analysis.
g. Convert the hibefil.sys into RAM dump using Moonsols Toolkit.
h. Extract process and app specific memory chunks using Volatility tool.
i. Analyze memory chunks using Volatility tool and Winhex tool.
j. Collect volatile evidences and build a timeline
k. Build the master timeline from the timeline of static analysis and volatile analysis process along with collected evidences in JSON file format.

The physical memory of computers running Windows operating systems contains a large amount of metadata necessary to manage the execution of processes running within the operating system. The proposed tool, is able to identify headers or other suspicious data, usernames, email address and passwords within the memory image of Windows 10.

Experimental Results and Discussions

A tool for physical memory dump achieve the goal of directly reading physical memory by loading loadable kernel modules or gaining reading and writing accesses of \Device\PhysicalMemory. Dumpit of Moonsols toolkit is one of them which is used in this technique to acquire the RAM dump. It leaves as small footprint as possible, without hindering investigators from deeper forensics. The result of Dumpit is the raw dump (.raw extension) of memory which is of the size of RAM. Hibernation file is extracted using Linux Live CD whose size is also exactly same as of RAM but only in the case if file extracted after a valid hibernation process by Windows. Hibernation file is compressed using Xpress algorithm and encoded with Huffman and LZ algorithms. Therefore Hibernation file cannot directly analyze with any tool. The encoded content of the Hibernation file converted into RAM dump using Hiber2Bin which is another tool from Moonsols toolkit. Although Volatility tool can be used to convert an encoded hibernation file to a RAM dump. A hibernation file also contains archival data in slack space that is the result of previous writes. The amount of archival data present in a hibernation file depends on the size of the file and the frequency of writes. A general rule of thumb is that the larger the hibernation file, the greater the chances of finding artifacts related to earlier sessions. Volatility tool and Winhex tool are used to analyze the acquired .raw and hiberfil.sys of the system. Using the proposed tool, KDBG and Device Tree Blob (DTB) address as shown in Figure 2 are extracted which are used further to extract more information like list of running process and exact version of the operating system.

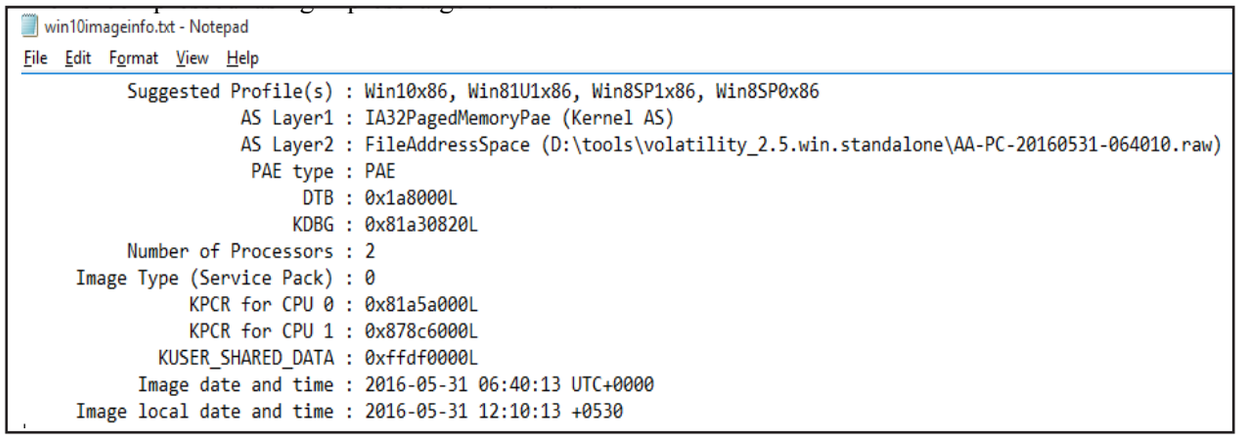
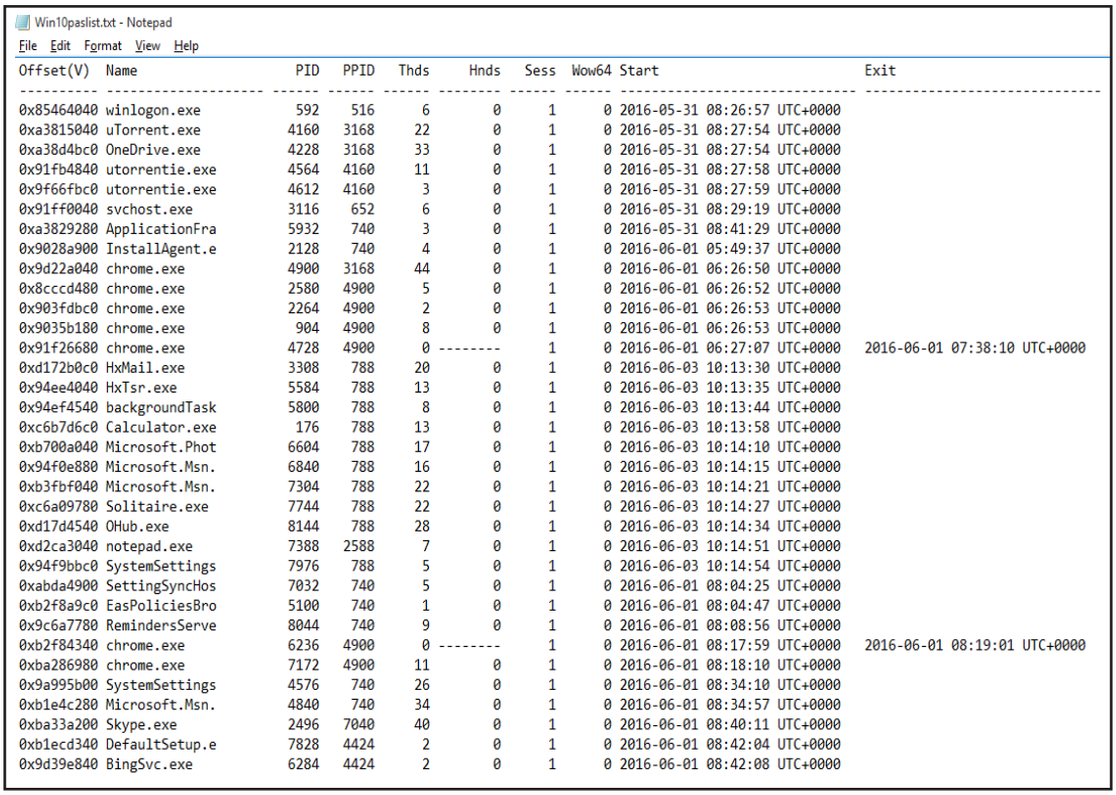


Figure 2: Extracted DTB and KDBG Information from RAM Dump

DTB is a database that represents the hardware components on the given mother board of the system. The KDBG is a structure maintained by the Windows kernel for debugging purpose. It contains a list of the running processes and loaded kernel modules. Version information allows to determine a memory dump come from which version of Windows, what Service Pack installed and the memory model (32-bit or 64-bit). After getting KDBG address one can have the running process at that time along with their timestamp as shown in Figure 3. From the running processes list, it can be identified that which process are running by the system along with their offset values which further can be used to extract data from the raw dump.



Win10paslist.txt - Notepad

File Edit Format View Help

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
0x85464040	winlogon.exe	592	516	6	0	1	0	2016-05-31 08:26:57 UTC+0000	
0xa3815040	uTorrent.exe	4160	3168	22	0	1	0	2016-05-31 08:27:54 UTC+0000	
0xa38d4bc0	OneDrive.exe	4228	3168	33	0	1	0	2016-05-31 08:27:54 UTC+0000	
0x91fb4840	utorrentie.exe	4564	4160	11	0	1	0	2016-05-31 08:27:58 UTC+0000	
0x9f66fbc0	utorrentie.exe	4612	4160	3	0	1	0	2016-05-31 08:27:59 UTC+0000	
0x91ff0040	svchost.exe	3116	652	6	0	1	0	2016-05-31 08:29:19 UTC+0000	
0xa3829280	ApplicationFra	5932	740	3	0	1	0	2016-05-31 08:41:29 UTC+0000	
0x9028a900	InstallAgent.e	2128	740	4	0	1	0	2016-06-01 05:49:37 UTC+0000	
0x9d22a040	chrome.exe	4900	3168	44	0	1	0	2016-06-01 06:26:50 UTC+0000	
0x8cccd480	chrome.exe	2580	4900	5	0	1	0	2016-06-01 06:26:52 UTC+0000	
0x903fdbc0	chrome.exe	2264	4900	2	0	1	0	2016-06-01 06:26:53 UTC+0000	
0x9035b180	chrome.exe	904	4900	8	0	1	0	2016-06-01 06:26:53 UTC+0000	
0x91f26680	chrome.exe	4728	4900	0	--------	1	0	2016-06-01 06:27:07 UTC+0000	2016-06-01 07:38:10 UTC+0000
0xd172b0c0	HxMail.exe	3308	788	20	0	1	0	2016-06-03 10:13:30 UTC+0000	
0x94ee4040	HxTsr.exe	5584	788	13	0	1	0	2016-06-03 10:13:35 UTC+0000	
0x94ef4540	backgroundTask	5800	788	8	0	1	0	2016-06-03 10:13:44 UTC+0000	
0xc6b7d6c0	Calculator.exe	176	788	13	0	1	0	2016-06-03 10:13:58 UTC+0000	
0xb700a040	Microsoft.Phot	6604	788	17	0	1	0	2016-06-03 10:14:10 UTC+0000	
0x94f0e880	Microsoft.Msn.	6840	788	16	0	1	0	2016-06-03 10:14:15 UTC+0000	
0xb3fbf040	Microsoft.Msn.	7304	788	22	0	1	0	2016-06-03 10:14:21 UTC+0000	
0xc6a09780	Solitaire.exe	7744	788	22	0	1	0	2016-06-03 10:14:27 UTC+0000	
0xd17d4540	OHub.exe	8144	788	28	0	1	0	2016-06-03 10:14:34 UTC+0000	
0xd2ca3040	notepad.exe	7388	2588	7	0	1	0	2016-06-03 10:14:51 UTC+0000	
0x94f9bbc0	SystemSettings	7976	788	5	0	1	0	2016-06-03 10:14:54 UTC+0000	
0xabda4900	SettingSyncHos	7032	740	5	0	1	0	2016-06-01 08:04:25 UTC+0000	
0xb2f8a9c0	EasPoliciesBro	5100	740	1	0	1	0	2016-06-01 08:04:47 UTC+0000	
0x9c6a7780	RemindersServe	8044	740	9	0	1	0	2016-06-01 08:08:56 UTC+0000	
0xb2f84340	chrome.exe	6236	4900	0	--------	1	0	2016-06-01 08:17:59 UTC+0000	2016-06-01 08:19:01 UTC+0000
0xba286980	chrome.exe	7172	4900	11	0	1	0	2016-06-01 08:18:10 UTC+0000	
0x9a995b00	SystemSettings	4576	740	26	0	1	0	2016-06-01 08:34:10 UTC+0000	
0xb1e4c280	Microsoft.Msn.	4840	740	34	0	1	0	2016-06-01 08:34:57 UTC+0000	
0xba33a200	Skype.exe	2496	7040	40	0	1	0	2016-06-01 08:40:11 UTC+0000	
0xb1ecd340	DefaultSetup.e	7828	4424	2	0	1	0	2016-06-01 08:42:04 UTC+0000	
0x9d39e840	BingSvc.exe	6284	4424	2	0	1	0	2016-06-01 08:42:08 UTC+0000	

Figure 3: Extracted List of Running Process Along with Their PIDs from RAM

The PID of the running process is used to extract the process specific memory chunks from the RAM dump. Besides process IDs, address of Kernel Processor Control Register (KPCR) can be extracted, which is different for each core of the multi core processor. KPCR address can be used to find the Interrupt Descriptor Table (IDT) and Global Descriptor Table (GDT) addresses, current or idle, next thread, CPU number, vendor, speed, CR3 value etc. Beside this, commands entered through console shell (cmd.exe) are extracted which is the most valuable information about the events fired from the system by an attacker. Network connections as shown in Table 1 made by the different process of the system along with their IP addresses and transmission protocol used are also extracted which is very useful to trace out the websites.

Table 1: Extracted Network Connections of the Process from RAM

Offset(P)	*Protocol*	*Local Address*	*Foreign Address*	*State*	*Pid*	
0x8d6a3e98	UDPv4	0.0.0.0:512	*:*	1472	uTorrent.exe	2016-05-31 06:28:01 UTC+0000
0x8d77ef40	UDPv4	169.254.234.166:512	*:*	1472	uTorrent.exe	2016-05-31 06:28:01 UTC+0000

Offset(P)	*Protocol*	*Local Address*	*Foreign Address*	*State*	*Pid*		
0x8d667790	TCPv4	10.100.19.168:49610	-:3128	FIN_WAIT1	-1		
0x8d7813f8	TCPv4	10.100.19.168:49716	-:3128	CLOSED	-1		
0x8da78b10	TCPv4	10.100.19.168:49638	-:3128	CLOSED	-1		
0x8da908f0	TCPv4	10.100.19.168:49649	-:3128	CLOSED	-1		
0x97270df0	TCPv4	10.100.19.168:49701	-:3128	CLOSED	-1		
0x973807a8 UTC+0000	UDPv6	: 1:5888	*:*	2900	svchost.exe	2016-05-31 06:36:43	
0x973e45c8	TCPv4	10.100.19.168:49595	-:3128	CLOSED	-1		
0x9ee08df0	TCPv4	10.100.19.168:49732	-:3128	CLOSED	-1		
0x9eee8b08	TCPv4	10.100.19.168:49662	-:3128	CLOSED	-1		
0xa902c2d0 UTC+0000	UDPv4	127.0.0.1:512	*:*	2900	svchost.exe	2016-05-31 06:36:43	

From the extracted PID of the process one can find out which pages of the process are memory resident and also the virtual address of the page along with its corresponding physical offset and the size of page and also one can dump the memory of a particular process and can extract the data. This thing is applied upon Google Chrome and Firefox browser and also the Windows Mail app and extract some data from them in plain text format. Which shows some contents of the send and received mails, various Email IDs and visited sites by the user. Figure 5 shows the browsed website by the user which include amazons.com and freerecharge.com, which include the order ID along with date and timestamp of the visited sites.

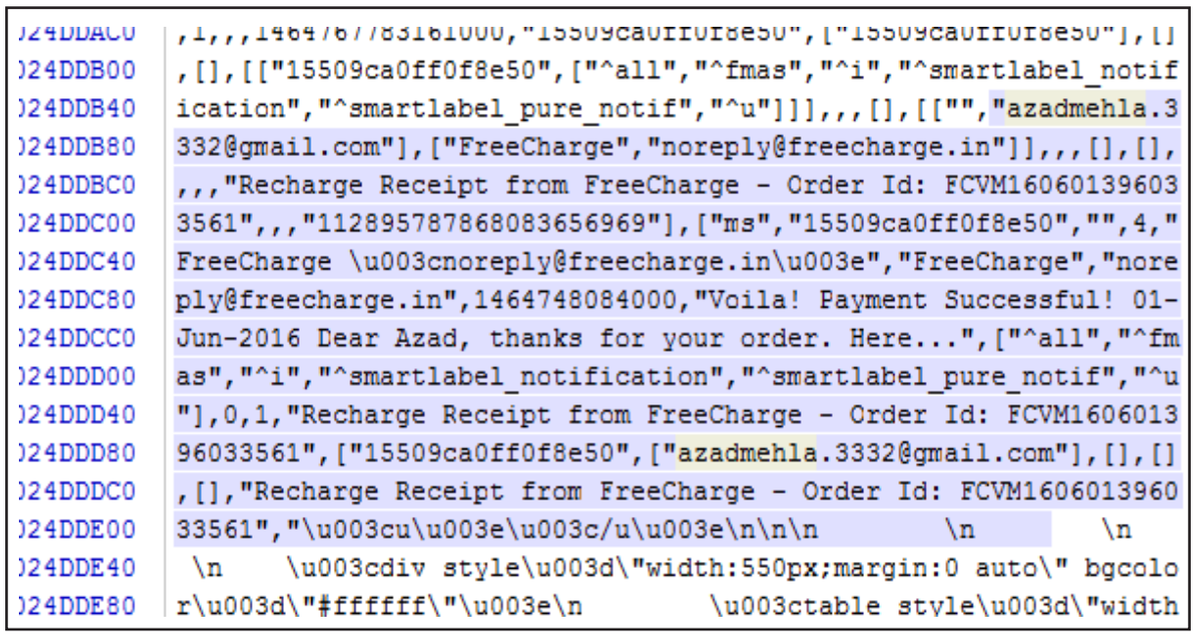
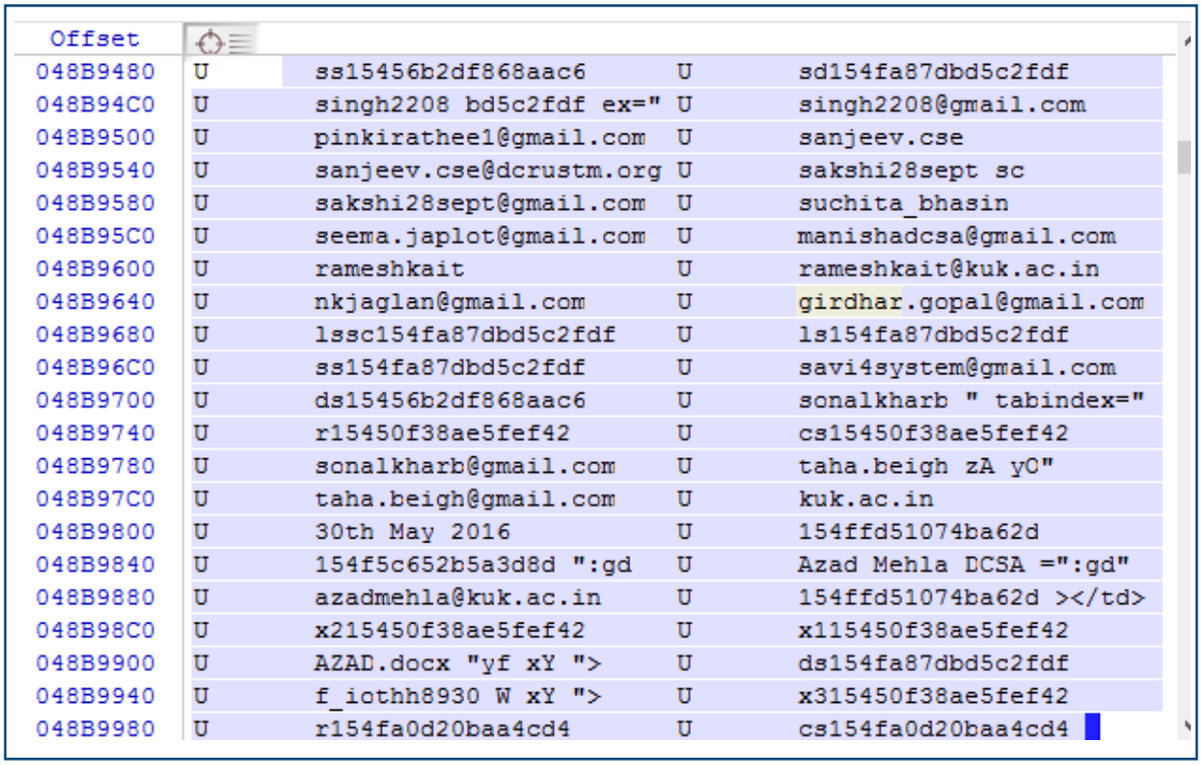


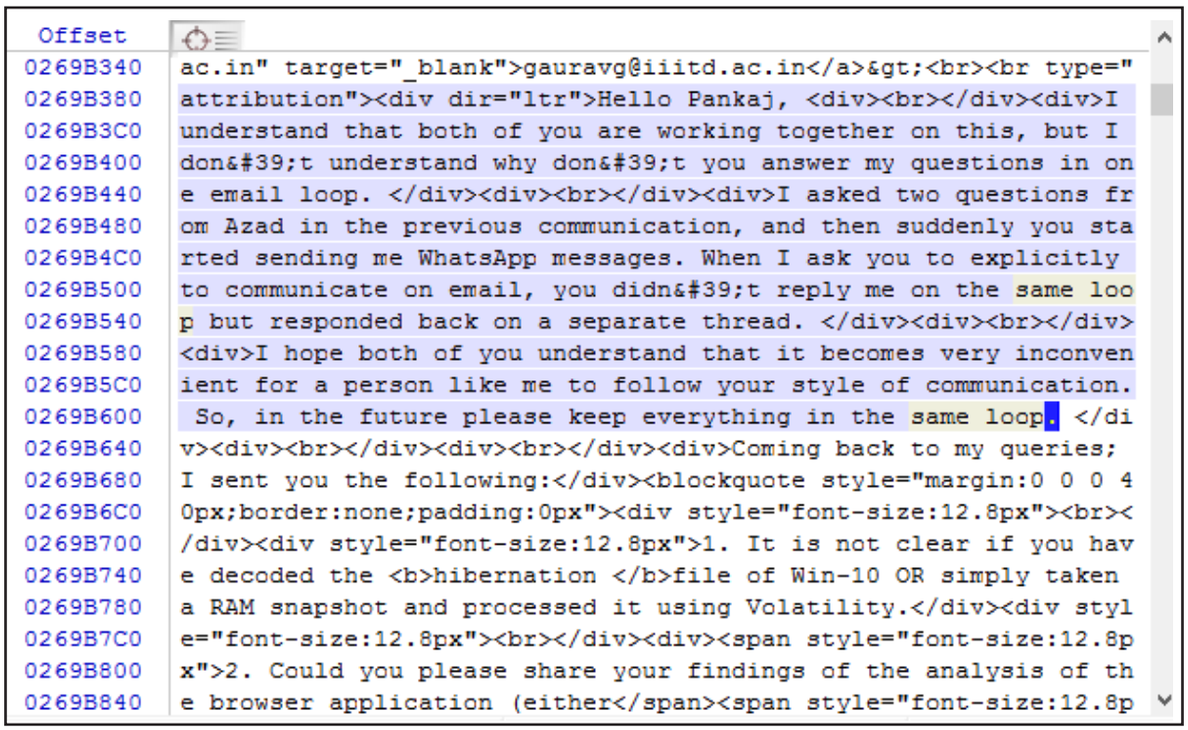
Figure 5: Extracted Amazons and Freecharge Data from Google Chrome

The Figure 6 shows the extracted gmail data, which includes the email IDs of the people with whom user connected. This list can give a huge opportunity to an investigator to trace these accounts.



Offset				
048B9480	U	ss15456b2df868aac6	U	sd154fa87dbd5c2fdf
048B94C0	U	singh2208 bd5c2fdf ex="	U	singh2208@gmail.com
048B9500	U	pinkirathee1@gmail.com	U	sanjeev.cse
048B9540	U	sanjeev.cse@dcrustm.org	U	sakshi28sept sc
048B9580	U	sakshi28sept@gmail.com	U	suchita_bhasin
048B95C0	U	seema.japlot@gmail.com	U	manishadcsa@gmail.com
048B9600	U	rameshkait	U	rameshkait@kuk.ac.in
048B9640	U	nkjaglan@gmail.com	U	girdhar.gopal@gmail.com
048B9680	U	lssc154fa87dbd5c2fdf	U	ls154fa87dbd5c2fdf
048B96C0	U	ss154fa87dbd5c2fdf	U	savi4system@gmail.com
048B9700	U	ds15456b2df868aac6	U	sonalkharb " tabindex="
048B9740	U	r15450f38ae5fef42	U	cs15450f38ae5fef42
048B9780	U	sonalkharb@gmail.com	U	taha.beigh zA yO"
048B97C0	U	taha.beigh@gmail.com	U	kuk.ac.in
048B9800	U	30th May 2016	U	154ffd51074ba62d
048B9840	U	154f5c652b5a3d8d ":gd	U	Azad Mehla DCSA =":gd"
048B9880	U	azadmehla@kuk.ac.in	U	154ffd51074ba62d ></td>
048B98C0	U	x215450f38ae5fef42	U	x115450f38ae5fef42
048B9900	U	AZAD.docx "yf xY ">	U	ds154fa87dbd5c2fdf
048B9940	U	f_iothh8930 W xY ">	U	x315450f38ae5fef42
048B9980	U	r154fa0d20baa4cd4	U	cs154fa0d20baa4cd4

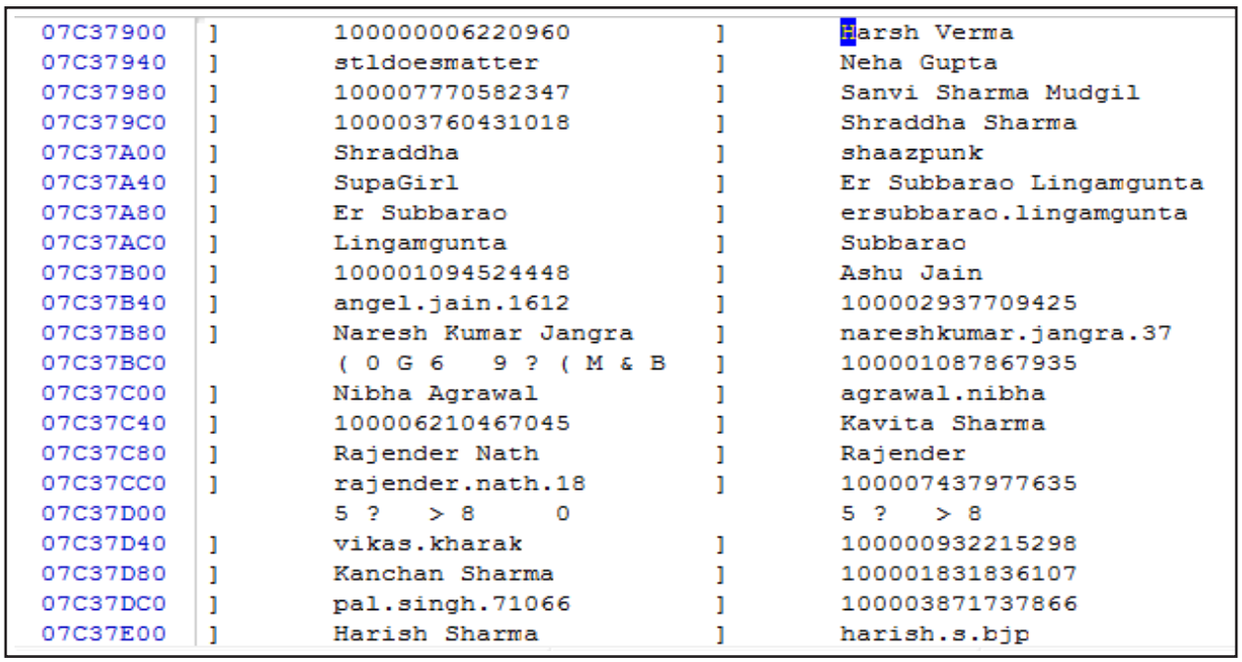
Figure 6: Extracted Email IDs of Persons with Whom Attacker Contacted



Offset	
0269B340	ac.in" target="_blank">gauravg@iiitd.ac.in> <br type="
0269B380	attribution"> Hello Pankaj, I
0269B3C0	understand that both of you are working together on this, but I
0269B400	don't understand why don't you answer my questions in on
0269B440	e email loop. I asked two questions fr
0269B480	om Azad in the previous communication, and then suddenly you sta
0269B4C0	rted sending me WhatsApp messages. When I ask you to explicitly
0269B500	to communicate on email, you didn't reply me on the same loo
0269B540	p but responded back on a separate thread.
0269B580	I hope both of you understand that it becomes very inconven
0269B5C0	ient for a person like me to follow your style of communication.
0269B600	So, in the future please keep everything in the same loop. </di
0269B640	v> Coming back to my queries;
0269B680	I sent you the following: <blockquote style="margin:0 0 0 4
0269B6C0	0px;border:none;padding:0px"> <
0269B700	/div> 1. It is not clear if you hav
0269B740	e decoded the hibernation file of Win-10 OR simply taken
0269B780	a RAM snapshot and processed it using Volatility. <div styl
0269B7C0	e="font-size:12.8px"> <span style="font-size:12.8p
0269B800	x">2. Could you please share your findings of the analysis of th
0269B840	e browser application (either<span style="font-size:12.8p

Figure 7: Extracted Message Contents of Received Email

Figure 8 shows the extracted user's Facebook data, which include the friend list of the user along with their name and Facebook IDs. This data can give a huge lead to the investigators.



07C37900	]	100000006220960	]	Harsh Verma
07C37940	]	stldoesmatter	]	Neha Gupta
07C37980	]	100007770582347	]	Sanvi Sharma Mudgil
07C379C0	]	100003760431018	]	Shraddha Sharma
07C37A00	]	Shraddha	]	shaazpunk
07C37A40	]	SupaGirl	]	Er Subbarao Lingamgunta
07C37A80	]	Er Subbarao	]	ersubbarao.lingamgunta
07C37AC0	]	Lingamgunta	]	Subbarao
07C37B00	]	100001094524448	]	Ashu Jain
07C37B40	]	angel.jain.1612	]	100002937709425
07C37B80	]	Naresh Kumar Jangra	]	nareshkumar.jangra.37
07C37BC0		(0 G 6 9 ? (M & B	]	100001087867935
07C37C00	]	Nibha Agrawal	]	agrawal.nibha
07C37C40	]	100006210467045	]	Kavita Sharma
07C37C80	]	Rajender Nath	]	Rajender
07C37CC0	]	rajender.nath.18	]	100007437977635
07C37D00		5 ? > 8 0		5 ? > 8
07C37D40	]	vikas.kharak	]	100000932215298
07C37D80	]	Kanchan Sharma	]	100001831836107
07C37DC0	]	pal.singh.71066	]	100003871737866
07C37E00	]	Harish Sharma	]	harish.s.bjp

Figure 8: Extracted Facebook's Friend List from Private Browsing of Mozilla Firefox

The technique that have proposed in the above section has collected a huge details of the attacker's system which also include some confidential data also like SBI net banking user ID and password. The processing time taken by the proposed technique is huge and also it depends upon the size of physical memory to be analyzed.

Conclusion and Future Work

The technique described in this paper gathers forensic artifacts related to Windows 10. It shows the experimental results of the proposed technique for extracting volatile evidences from physical memory of Windows 10. Volatile artifacts are extracted from the hibernation and RAM dump, which provides comprehensive reconstruction of events. Preliminary results show that the three are numerous valuable evidences extracted from the raw dump of the RAM, which can be used to identify the suspected process and finding malwares. Also the evidences gathered with this technique plays a significant role in solving the real time cases by identifying of the fraudster and suspected activates launched from his/her system. Further research will focus on developing similar technique for the Windows phones and tablet devices and also research will attempt to analyze the collected data more effectively.

References

1. Rahman, S., & Khan, M. N. A. (2015). Review of live forensic analysis techniques. *International Journal of Hybrid Information Technology,* 8(2), 379-388.
2. Youngsoo, K., Sangsu, L., & Hong, D. (2008). Suspects' data hiding at remaining registry values of uninstalled programs. In *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop*, Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering.
3. Singh, A., Sharma, P., & Nath, R. (2016). Role of hibernation file in memory forensics of Windows 10. *In Vivechana: National Conference on Advances in Computer Science and Engineering (ACSE-2016)*.
4. Wang, L., Zhang, R., & Zhang, S. (2009). A model of computer live forensics based on physical memory analysis. *In IEEE Information Science and Engineering (ICISE)*.
5. Kristine, A., & Carlos, C. (2009). Techniques and Tools for Recovering and Analyzing Data from Volatile Memory. *Sans Institute Infosec Reading Room*.
6. Gianni, F., & Solinas, F. (2013). Live Digital Forensics: Windows XP vs Windows 7. *In IEEE*

Informatics and Applications (ICIA), 2013 2nd International Conference.

7. Hang, C. H., & Yen, P. H. (2010). Fast deployment of computer forensics with USBs. *In IEEE Broadband, Wireless Computing, Communication and Applications (BWCCA)*.
8. Dija, S., Deepthi, T., Balan, C., & Thomas, K. (2012). Towards retrieving live forensic artifacts in offline forensics. *In Recent Trends in Computer Networks and Distributed Systems Security*, Berlin Heidelberg, Germany, Springer-Verlag, (pp. 225-233).
9. Mrdovic, S., & Huseinovic, A. (2011). Forensic analysis of encrypted volumes using hibernation files. *In Proceedings of the 19th Telecommunications Forum.*
10. Gupta, D., & Mehte, B. (2013). Forensic analysis of Sandboxie artifacts. *In Security in Computing and Communications*, Berlin Heidelberg, Germany, Springer-Verlag, (pp. 341-352).
11. Iqbal, A., Obaidli, H. A., Marrington, A., & Jone, A. (2014). Windows Surface RT tablet forensics. *Digital Investigation*, 11(S1), S87-S93.
12. Carvey, H. (2014). Windows Forensic Analysis Toolkit: Advanced Analysis Techniques for Windows 8, *Waltham*, Massachusetts: Syngress.
13. Quick, D., & Choo, K. (2013). Digital droplets: Microsoft SkyDrive forensic data remnants. *Future Generation Computer Systems*, 29(6), 1378-1394.
14. Quick, D., & Choo, K. (2014). Forensic analysis of data remnants. *Journal of Network and Computer Applications*, 40, 179-193.
15. Quick, D., & Choo, K. (2013). Dropbox analysis: Data remnants on user machines. *Digital Investigation*, 10(1), 3-18.
16. Russinovich, M., Solomon, D. A., & Ionescu, A. (2009). *Windows Internals*, (5thed), Microsoft Press.
17. Schuster, A. (2006). Searching for processes and threads in microsoft windows memory dumps. *In The Proceedings of the 6th Annual Digital Forensic Research Workshop (DFRWS '06)*.
18. Balogh, S., & Pondelik, M. (2011). Capturing encryption keys for digital analysis. *IEEE 6th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)*, 2.
19. Aljaedi, A., Lindskog, D., Zavarsky, P., Ruhl, R., & Almari, F. (2011). Comparative Analysis of Volatile Memory Forensics: Live Response vs. Memory Imaging. In IEEE Privacy, Security, Risk and Trust (PASSAT), *IEEE 3rd International Conference on Social Computing.*