Scientific Research Publishing

World Journal of Nano Science and Engineering, 2025, 15(1), 1-12
https://www.scirp.org/journal/wjnse
ISSN Online: 2161-4962
ISSN Print: 2161-4954

Security and Forensic Analysis of the Ram of a Computer Infected by a Malware

Yanogo Kiswendsida Jean Hermann[1], Djibo Moumouni[2], Kahoun Zita Phillipe[3], Nabollé Rachid Gaetan[4], Diberot Cidjeu[5], Gérald Yirga[6], Ouedraogo Tounwendyam Frederic[6]

[1]Institute of Computer Engineering and Telecommunication Polytechnic, School of Ouagadougou, Ouagadougou, Burkina Faso
[2]Virtual University, Ouagadougou, Burkina Faso
[3]Aube Nouvelle University, Ouagadougou, Burkina Faso
[4]Burkina Institute of Technology, Koudougou, Burkina Faso
[5]CERPAMAD, Ouagadougou, Burkina Faso
[6]Norbert Zongo University, Koudougou, Burkina Faso
Email: yanogohermann@yahoo.fr

How to cite this paper: Hermann, Y.K.J., Moumouni, D., Phillipe, K.Z., Gaetan, N.R., Cidjeu, D., Yirga, G. and Frederic, O.T. (2025) Security and Forensic Analysis of the Ram of a Computer Infected by a Malware. *World Journal of Nano Science and Engineering*, **15**, 1-12.
https://doi.org/10.4236/wjnse.2025.151001

Received: February 15, 2025
Accepted: March 28, 2025
Published: March 31, 2025

Abstract

Our world is increasingly immersed in technology and malware pose a formidable threat. Malware exploits security vulnerabilities (on a large scale), causing enormous financial losses and compromise of vital information for many businesses, individuals and government institutions. To counter these attacks, forensic malware analysis is crucial. This research focuses on analyzing the RAM of computers infected with virus. The goal of this research is to enable digital investigators to better understand malware behavior and implement effective solutions to analyze compromised computers. The research question is how IT professionals can better analyze infected computer bur using the right method. This work uses tools such as LiME and Volatility and secure environments such as CSI Linux and Tsurugi Linux. In this search we identify suspicious processes running, as well as suspicious active network connections. The results show characteristic alterations, such as changes in system processes or unusual memory access patterns, aligned with known virus techniques. Forensic analysis on network connections shows that the virus established outbound connections to command and control (C&C) servers to receive instructions and send encrypted data by identifying communications to associated suspicious IP addresses and ports to the C&C. This research is crucial because behavioral analysis of malware contributes to the development of more effective mitigation techniques, thereby reducing the risk of infection. Additionally, this research can be valuable for designing malware decryption tools, providing an opportunity for data recovery after a file encryption attack.

Keywords

Forensic, RAM, CSI Linux, Virus, Tsurugi, Volatility, LiME

1. Introduction

Forensic analysis of the RAM is the practice of collecting and organizing information found on an electronic device for investigative purposes. It is important to know both the technologies and the methods and frameworks investigators use in this field [1]. Technological progress has led to transformations within our society, thus shaping our individual and professional daily lives. The concurrent emergence of big data, artificial intelligence and the Internet of Things has substantially reshaped the business landscape, integrating these innovations as fundamental elements. These technologies, which have become omnipresent, are undergoing perpetual evolution, systemically redefining the methods of communication and work. Thus, this observed technological metamorphosis has significantly revolutionized human interactions and content transmission processes. However, with this increasing integration into our lives and professional environments, these technological advances have also given rise to a new form of crime operating in cyberspace (such as phishing, denial of service attacks, ransomware, etc.). The rise of certain cybercrime activities has greatly complicated the work of digital investigators, particularly with regard to malware. Thus, the objective of this research lies in the forensic analysis of the RAM of computers compromised by a cyber virus attack, exploring the methodologies, techniques and tools required to extract crucial information from this essential element of the system computer science. While traditional forensic investigation techniques focused primarily on hard drives and files, examining RAM provides an invaluable window into the state of the system at the time of infection, allowing a detailed understanding of activities. malicious activities in progress.

2. Methodology

Forensic analysis of the RAM of a virus infected computer requires a precise methodology to extract crucial information without altering the data. This methodology requires in-depth computer forensics expertise and the proper use of specialized tools to extract and analyze RAM without compromising the data. There're some problems that must be paid attention in the process of electronic evidence collection. The investigator should know the techniques and methods of cybercrime scene protection [2].

So during our work, the first step in our methodology consists of setting up a secure working environment called Sandbox. For the realization of our project, the choice of our environment fell on VirtualBox, a virtualization software which allows the creation of virtual machines on a host system. On this environment, we installed Ubuntu 18.04.6 victim machines and the CSI Linux machine which will

be used to carry out the analysis.

The second step was to infect our machines with Erabus malware. There are many ways in which a machine can become infected with virus malware. Regarding our work in this research, we had extracted the malicious malware in a victim's computer and then executed it in the Ubuntu virtual machine so that they were infected. One of the prerequisites was an internet connection during the analysis, so that the malware could communicate with the C&C server and thus describe its true behavior. Non-availability of internet prevents the malware from running as it cannot download encryption keys etc.

Then, it was essential to obtain samples of the RAM of the compromised machines by performing dumps using forensic tools such as LiME in its version 1.9.1.

Finally, we analyzed the data using the volatility framework to look for signatures, suspicious behavior and traces of ransomware in the memory. It is important to master the application tools that you will use for the investigation of RAM [3].

Figure 1 describes the different stages of our work methodology. So our work methodology was summarized according to the following points:

- Setting up the work environment;
- Attack of the virtual machine;
- Collection of data through memory captures;
- Analysis of dumps.

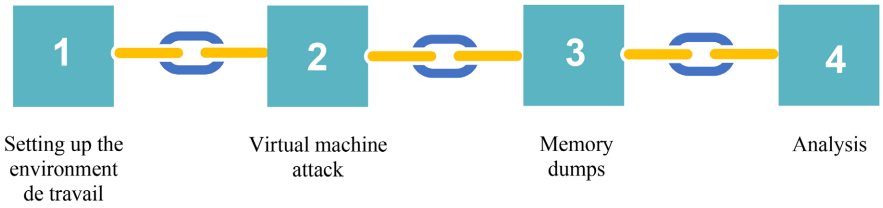


Figure 1. Methodology used.

3. Traditional Analysis Compared to Forensic Analysis of RAM

Memory forensics is the process of capturing the execution memory of a device and then analyzing the captured output for the presence of malware. Unlike hard drive forensics, where a device's file system is cloned and every file on the drive can be retrieved and analyzed, memory forensics focuses on the programs in question. Running on a device while capturing the memory dump.

In practice, the first step in memory investigation is to obtain a copy of the contents of volatile memory called Memory Dump. Once this step has been executed successfully, the dump should be analyzed. In this case, evidence is extracted by analyzing and interpreting operating system constructs.

The traditional investigation approach poses a certain number of difficulties:

- The difficulty linked to the identification of the traces left by the virus in the RAM;

- The lack of an appropriate technique for examining network connections;
- The difficulty of reconstructing the chronological sequence of malicious actions;
- The difficulty associated with extracting relevant data without altering the integrity of the digital evidence;
- The difficulty linked to traditional analysis which requires large storage capacity.

In view of all these difficulties and the growing threat of virus, it is appropriate to integrate forensic analysis of RAM on computers compromised during computer attacks and especially with regard to virus. Forensic Investigations can be difficult with hard drive [4].

The advantages of forensic analysis are:

- Forensic analysis makes it possible to extract relevant data from RAM without altering the integrity of the digital evidence;
- Forensic analysis of the RAM of a virus infected system can reveal indicators to identify specific processes related to malware activity and operation;
- Analyzing network connections after a virus attack can help determine outgoing communications to the command server.

Malware is one of the most advanced malware which uses computer resources and services to encrypt system data once it infects a system and causes large financial data losses to the organization and individuals [5].

4. Setting Up the Attack System

We install the LiME tool on Ubuntu 18.04.6 in order to perform a memory dump of the victim machine.

- We will first update the packages

  sudo apt update sudo apt upgrade
- Clone the repository on the machine with the following github* link followed by the compilation of LiME

  git clone https://github.com/504ensicsLabs/LiME.git

Figure 2 shows the cloning of the LiME git repository on the victim Ubuntu machine.

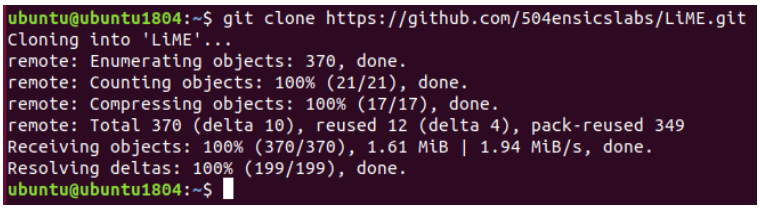


Figure 2. Cloning the LiME repository on the victim machine.

cd LiME/src && make

We now install Volatility Using pip

sudo python3 setup.py install

Regarding the work of this research, we had extracted the malicious virus in a victim's computer and then executed it in the first Ubuntu virtual machine so that

they were infected. We then reproduced the infection in the other Ubuntu virtual machines while remaining in our controlled environment. During the attack, one of the prerequisites was to maintain an internet connection so that the malware could communicate with the C&C server and thus describe its true behavior.

Figure 3 shows the Ubuntu 18.04.6 virtual machine before the malware attack. The next step is to infect our virtual machine.

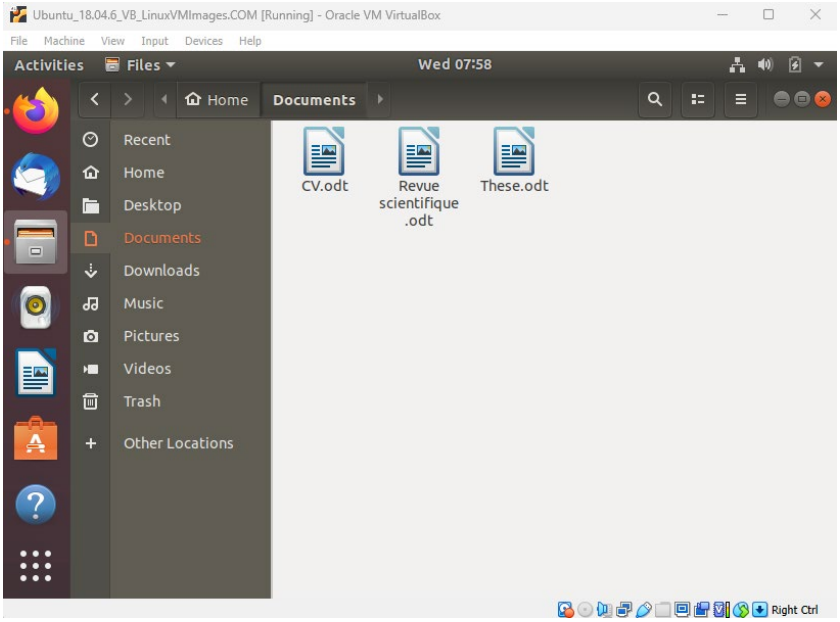


Figure 3. Capture of the victim machine before the virus attack.

Figure 4 comes from the victim machine after infecting the machine with malware. We find that our files have been encrypted and a ransom message has been dropped by the cybercriminal namely the readme_for_unlock.txt file.

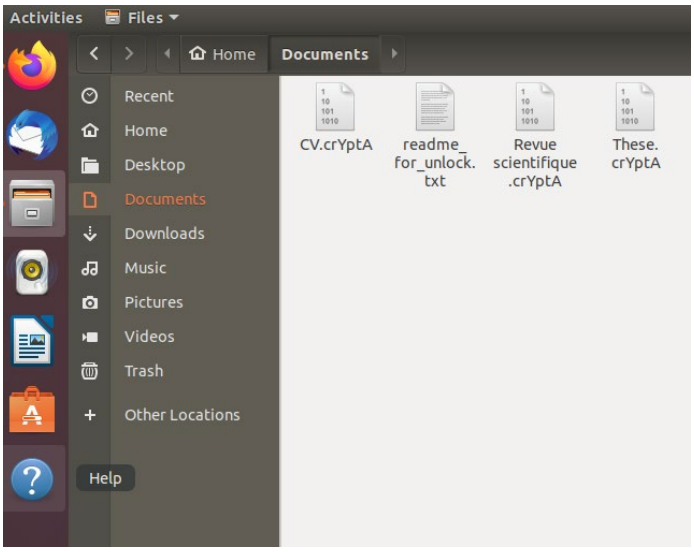


Figure 4. Capture of the victim machine after attack.

5. Result and Discussion

5.1. Memory Extraction Result

Memory forensics, where you have a chance to analyze and determine if a given sample is malware or not without going for complex reverse engineering techniques is the best solution [6]. When a computer is infected with irus, the dump memory process can be tricky because the ransomware can interfere with normal system operations. However, the above process could be considered. In the case of a computer infected with virus, the priority is to contain and isolate the infection as best as possible to prevent its spread. Memory dump can be difficult to perform due to the destructive or intrusive nature of the virus. The memory of the infected machine was extracted with LiME using the following command:

sudo insmod lime-$(uname -r).ko "path=/path/to/save/destination format=lime"

Figure 5 illustrates the result of the memory capture of our infected machine. As shown in the image above, our memory dump is 4 gigabytes which is exactly the size of the RAM of the Ubuntu machine. Memory forensics has come to the forefront as a formidable instrument in the ongoing struggle against malware, providing researchers with the means to examine these threats within their operational environment and extract valuable insights from the behaviors they exhibit while active [7].

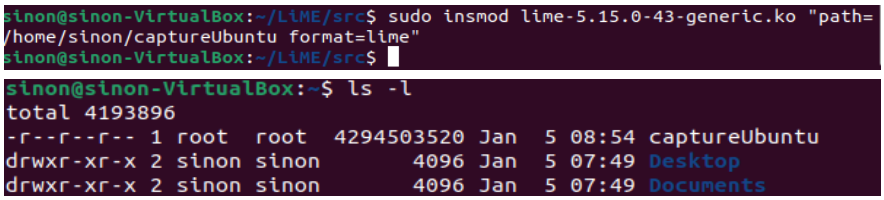
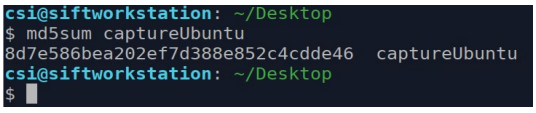


Figure 5. Memory dump capture.

It is crucial to note that memory capture may require elevated privileges and should be done carefully to avoid modifying the target system's memory. It is also possible to do the Memory Dump via the network with LiME. Verifying the integrity of the memory dump to ensure that it was not altered during capture was carried out by calculating the MD5 hash of the original memory dump and that of the extracted file. The two operations are illustrated in the figures below. The result gives the same Hash indicating that the dump has not been altered.

Calculation of the Md5 hash from memory directly on the ubuntu machine

Calculating the Md5 hash from memory directly on the CSI Linux machine



These figures show the hash calculations just after the dump in the victim machine then in the CSI machine which will be used for the analysis. The observation made is that the two hashes are identical, which makes it possible to verify the integrity of the dump.

5.2. Result Based on Analysis of Running Processes

Examining running processes involves observing the programs and tasks currently operating within a computer system. The graphic representation below details our use of the Volatility tool to explore the RAM of our computer, affected by a virus attack. Volatility allows to find several key artifacts including different ways of listing processes, finding network connections, and using the module malfind that can detect suspicious instructions. Looking at memory analysis for use as a part of incident response, it usually comes down to finding signs of intrusions or malicious code [8].

Using "volatility psscan", we were able to examine and identify suspicious or malicious processes that could be linked to virus activities.

python vol.py --file=/home/csi/Desktop/E001.raw --profile=Linuxubuntu_5_4_0-91-genericx64 linux_psscan

Figure 6 shows the process structures and the list of processes running at the time the memory capture was performed. It provides information on active processes, their identifiers, process owners and execution times.

Show running processes with Pslist tool: This is a command designed to scan the memory of an operating system and list all active processes at the time of memory capture. With "Pslist" we scanned the RAM to extract information about the running processes such as process identifiers (PIDs), process names, session IDs, users associated with these processes.

```
csi@siftworkstation: ~/volatility
$ python vol.py --file=/home/csi/Desktop/E001.raw --profile=Linuxubuntu_5_4_0-91-genericx64 linux_psscan
```

```
Offset             Name                 Pid             PPid            Uid             Gid    DTB                Start Time
------------------ -------------------- --------------- --------------- --------------- ------ ------------------ ----------
0x000000002d408000 TaskCon~ller #0      11212           -               -1              -1     ------------------ -
0x000000002d409740 URL Classifier       10359           -               -1              -1     ------------------ -
0x000000002d40ae80 DOM Worker           10360           -               -1              -1     ------------------ -
0x000000002d40c5c0 MemoryPoller         10363           -               -1              -1     ------------------ -
0x000000002d40dd00 FS Broker 11229      11231           -               -1              -1     ------------------ -
0x000000002d4f8000 TaskCon~ller #2      10318           -               -1              -1     ------------------ -
0x000000002d4f9740 TaskCon~ller #0      10316           -               -1              -1     ------------------ -
0x000000002d4fae80 TaskCon~ller #1      10317           -               -1              -1     ------------------ -
0x000000002d4fc5c0 Worker Launcher      10315           -               -1              -1     ------------------ -
0x000000002d4fdd00 TaskCon~ller #3      10319           -               -1              -1     ------------------ -
0x000000002d540000 dircolors            11487           -               -1              -1     ------------------ -
0x000000002d541740 lesspipe             11483           -               -1              -1     ------------------ -
0x000000002d542e80 lesspipe             11485           -               -1              -1     ------------------ -
0x000000002d5445c0 timer                7841            -               -1              -1     ------------------ -
0x000000002d545d00 SwComposite          10253           -               -1              -1     ------------------ -
0x000000002d548000 avml                 11502           -               -1              -1     ------------------ -
0x000000002d549740 pool-caja            11471           -               -1              -1     ------------------ -
0x000000002d54ae80 pool-caja            11474           -               -1              -1     ------------------ -
0x000000002d54c5c0 sudo                 11501           -               -1              -1     ------------------ -
0x000000002d54dd00 pool-caja            11472           -               -1              -1     ------------------ -
0x000000002d558000 mozStorage #5        10429           -               -1              -1     ------------------ -
0x000000002d559740 RemVidChild          11210           -               -1              -1     ------------------ -
0x000000002d55ae80 ImageIO              11211           -               -1              -1     ------------------ -
```

```
0x0000000198ad45c0 cpuhp/3              27              -               -1              -1      ------------------ -
0x0000000198ad5d00 ksoftirqd/2          24              -               -1              -1      ------------------ -
0x0000000198ae0000 ksoftirqd/3          30              -               -1              -1      ------------------ -
0x0000000198ae2e80 kworker/3:0          11404           -               -1              -1      ------------------ -
0x0000000198ae45c0 kworker/3:0H         32              -               -1              -1      ------------------ -
0x0000000198b38000 kauditd              36              -               -1              -1      ------------------ -
0x0000000198b39740 netns                34              -               -1              -1      ------------------ -
0x0000000198b3ae80 khungtaskd           37              -               -1              -1      ------------------ -
0x0000000198b3c5c0 kdevtmpfs            33              -               -1              -1      ------------------ -
0x0000000198b3dd00 rcu_tasks_kthre      35              -               -1              -1      ------------------ -
0x0000000198bf8000 kworker/1:1H         218             -               -1              -1      ------------------ -
0x0000000198bf9740 cups-browsed         7433            -               -1              -1      ------------------ -
0x0000000198bfae80 ModemManager         878             -               -1              -1      ------------------ -
0x0000000198bfdd00 ttm_swap             217             -               -1              -1      ------------------ -
0x0000000198a42d18                      290             -               -1              -1      ------------------ -
0x0000000198a345c0 idle_inject/0        13              -               -1              -1      ------------------ -
0x0000000198a78000 idle_inject/1        16              -               -1              -1      ------------------ -
0x0000000198a8ae80 idle_inject/2        22              -               -1              -1      ------------------ -
0x0000000198ae1740 idle_inject/3        28              -               -1              -1      ------------------ -
0x0000000198bfc5c0 irq/18-vmwgfx        216             -               -1              -1      ------------------ -
0x0000000198a32e80 migration/0          12              -               -1              -1      ------------------ -
0x0000000198a7ae80 migration/1          17              -               -1              -1      ------------------ -
0x0000000198ad1740 migration/2          23              -               -1              -1      ------------------ -
0x0000000198ae5d00 migration/3          29              -               -1              -1      ------------------ -
csi@siftworkstation: ~/volatility
```

Figure 6. Result of the psscan command.

```
csi@siftworkstation: ~/volatility
$ python vol.py --file=/home/csi/Desktop/E001.raw --profile=Linuxubuntu_5_4_0-91-genericx64 linux_pslist
```

```
Offset             Name                 Pid             PPid            Uid             Gid    DTB                Start Time
------------------ -------------------- --------------- --------------- --------------- ------ ------------------ ----------
0xffff998d18a145c0 systemd              1               0               0               0      0x00000001971be000 2022-05-28 21
:48:44 UTC+0000
0xffff998d18a11740 kthreadd             2               0               0               0      ------------------ 2022-05-28 21
:48:44 UTC+0000
0xffff998d18a15d00 rcu_gp               3               2               0               0      ------------------ 2022-05-28 21
:48:44 UTC+0000
0xffff998d18a10000 rcu_par_gp           4               2               0               0      ------------------ 2022-05-28 21
:48:44 UTC+0000
0xffff998d18a2ae80 kworker/0:0H         6               2               0               0      ------------------ 2022-05-28 21
:48:44 UTC+0000
0xffff998d18a2dd00 mm_percpu_wq         9               2               0               0      ------------------ 2022-05-28 21
:48:44 UTC+0000
0xffff998d18a28000 ksoftirqd/0          10              2               0               0      ------------------ 2022-05-28 21
:48:44 UTC+0000
0xffff998d18a30000 rcu_sched            11              2               0               0      ------------------ 2022-05-28 21
:48:44 UTC+0000
0xffff998d18a32e80 migration/0          12              2               0               0      ------------------ 2022-05-28 21
:48:44 UTC+0000
0xffff998d18a345c0 idle_inject/0        13              2               0               0      ------------------ 2022-05-28 21
:48:44 UTC+0000
0xffff998d18a79740 cpuhp/0              14              2               0               0      ------------------ 2022-05-28 21
:48:44 UTC+0000
```

```
0xffff998d18a89740 scsi_tmf_3           11409           2               0               0      ------------------ 2022-05-29 02
:11:08 UTC+0000
0xffff998d1818dd00 usb-storage          11410           2               0               0      ------------------ 2022-05-29 02
:11:08 UTC+0000
0xffff998d180e1740 uas                  11413           2               0               0      ------------------ 2022-05-29 02
:11:08 UTC+0000
0xffff998d180e2e80 kworker/u8:1         11417           2               0               0      ------------------ 2022-05-29 02
:11:09 UTC+0000
0xffff998d10641740 kworker/2:1          11423           2               0               0      ------------------ 2022-05-29 02
:11:09 UTC+0000
0xffff998d180d0000 jbd2/sdb1-8          11461           2               0               0      ------------------ 2022-05-29 02
:11:11 UTC+0000
0xffff998d180d2e80 ext4-rsv-conver      11462           2               0               0      ------------------ 2022-05-29 02
:11:11 UTC+0000
0xffff998d1812dd00 ext4lazyinit         11463           2               0               0      ------------------ 2022-05-29 02
:11:11 UTC+0000
0xffff998c19cec5c0 mate-terminal        11477           1               1000            1000   0x000000018d608000 2022-05-29 02
:11:17 UTC+0000
0xffff998d09badd00 bash                 11482           11477           1000            1000   0x000000018a950000 2022-05-29 02
:11:17 UTC+0000
0xffff998bad54c5c0 sudo                 11501           11482           0               0      0x00000001912f8000 2022-05-29 02
:13:57 UTC+0000
0xffff998bad548000 avml                 11502           11501           0               0      0x0000000192848000 2022-05-29 02
:13:57 UTC+0000
csi@siftworkstation: ~/volatility
```

Figure 7. Result of the pslist command.

python vol.py --file=/home/csi/Desktop/E001.raw --profile=Linuxubuntu_5_4_0-91-genericx64 linux_pslist

Figure 7 shows an overview of the active processes at the time of memory capture. This was necessary to detect suspicious activities. Unlike pslist which focuses on active processes at the time of memory capture, psscan performs a deeper scan of memory to look for process structures, even those that might be hidden or deleted.

Displaying malicious processes with the volatility malfind tool: This command is designed to detect suspicious or potentially malicious structures in the memory of the system being scanned. In our research, it was useful for examining system memory and spotting indicators of compromise by looking for typical malware characteristics. Alternatively, it can be used to search for code injected into processes, hooks or abnormal memory modifications, signs of suspicious activity or artifacts related to attacks.

Figure 8 shows suspicious or potentially malicious memory sections. It particularly illustrates malicious processes.

Analysis of the contents of certain files with the linux_proc_maps tool: In a Linux operating system, the /proc/[PID]/maps file exposes information about the memory address space of a specific process where [PID] represents the process identifier. This information includes virtual memory ranges allocated to the process, access permissions (read, write, execute), path of loaded shared libraries (.so files), among other details.

```
csi@siftworkstation: ~/volatility
$ python vol.py -f /home/csi/Desktop/E001.raw --profile=Linuxubuntu_5_4_0-91-genericx64 linux_malfind
Volatility Foundation Volatility Framework 2.6.1
Process: networkd-dispat Pid: 772 Address: 0x7f1706a92000 File: Anonymous Mapping
Protection: VM_READ|VM_WRITE|VM_EXEC
Flags: VM_READ|VM_WRITE|VM_EXEC|VM_MAYREAD|VM_MAYWRITE|VM_MAYEXEC|VM_ACCOUNT|VM_CAN_NONLINEAR

0x007f1706a92000  00 00 00 00 00 00 00 00 43 00 00 00 00 00 00 00   ........C.......
0x007f1706a92010  4c 8d 15 f9 ff ff ff ff 25 03 00 00 00 0f 1f 00   L.......%.......
0x007f1706a92020  40 a1 a5 06 17 7f 00 00 08 3d 62 01 00 00 00 00   @........=b.....
0x007f1706a92030  20 7b c1 05 17 7f 00 00 f0 3c 62 01 00 00 00 00   .{.......<b.....



Process: mintmenu Pid: 1711 Address: 0x7fd0a70b7000 File: Anonymous Mapping
Protection: VM_READ|VM_WRITE|VM_EXEC
Flags: VM_READ|VM_WRITE|VM_EXEC|VM_MAYREAD|VM_MAYWRITE|VM_MAYEXEC|VM_ACCOUNT|VM_CAN_NONLINEAR

0x007fd0a70b7000  00 00 00 00 00 00 00 00 43 00 00 00 00 00 00 00   ........C.......
0x007fd0a70b7010  4c 8d 15 f9 ff ff ff ff 25 03 00 00 00 0f 1f 00   L.......%.......
0x007fd0a70b7020  40 b1 c5 a7 d0 7f 00 00 b8 24 95 01 00 00 00 00   @........$......
0x007fd0a70b7030  20 9b f8 a6 d0 7f 00 00 a0 24 95 01 00 00 00 00   .........$......



Process: mate-xapp-statu Pid: 1719 Address: 0x7f9953d33000 File: Anonymous Mapping
Protection: VM_READ|VM_WRITE|VM_EXEC
Flags: VM_READ|VM_WRITE|VM_EXEC|VM_MAYREAD|VM_MAYWRITE|VM_MAYEXEC|VM_ACCOUNT|VM_CAN_NONLINEAR

0x007f9953d33000  00 00 00 00 00 00 00 00 43 00 00 00 00 00 00 00   ........C.......
0x007f9953d33010  4c 8d 15 f9 ff ff ff ff 25 03 00 00 00 0f 1f 00   L.......%.......
0x007f9953d33020  40 81 52 54 99 7f 00 00 58 69 f7 01 00 00 00 00   @.RT....Xi......
0x007f9953d33030  20 ab 30 54 99 7f 00 00 40 69 f7 01 00 00 00 00   ..0T....@i......
```

```
Process: mintUpdate Pid: 2014 Address: 0x7febf03df000 File: Anonymous Mapping
Protection: VM_READ|VM_WRITE|VM_EXEC
Flags: VM_READ|VM_WRITE|VM_EXEC|VM_MAYREAD|VM_MAYWRITE|VM_MAYEXEC|VM_ACCOUNT|VM_CAN_NONLINEAR

0x007febf03df000  00 00 00 00 00 00 00 00 43 00 00 00 00 00 00 00   ........C.......
0x007febf03df010  4c 8d 15 f9 ff ff ff ff 25 03 00 00 00 0f 1f 00   L.......%.......
0x007febf03df020  40 11 5f f7 eb 7f 00 00 18 b4 cc 01 00 00 00 00   @._.............
0x007febf03df030  20 fb ad f6 eb 7f 00 00 00 b4 cc 01 00 00 00 00   ................



Process: mintreport-tray Pid: 2059 Address: 0x7f726ed32000 File: Anonymous Mapping
Protection: VM_READ|VM_WRITE|VM_EXEC
Flags: VM_READ|VM_WRITE|VM_EXEC|VM_MAYREAD|VM_MAYWRITE|VM_MAYEXEC|VM_ACCOUNT|VM_CAN_NONLINEAR

0x007f726ed32000  00 00 00 00 00 00 00 00 43 00 00 00 00 00 00 00   ........C.......
0x007f726ed32010  4c 8d 15 f9 ff ff ff ff 25 03 00 00 00 0f 1f 00   L.......%.......
0x007f726ed32020  40 f1 7c 6f 72 7f 00 00 18 b0 d7 02 00 00 00 00   @.|or...........
0x007f726ed32030  20 db bb 6e 72 7f 00 00 00 b0 d7 02 00 00 00 00   ...nr...........



csi@siftworkstation: ~/volatility
$
```

Figure 8. Result of the malfind command.

By examining the contents of this file with the linux_proc_maps command, we interpreted this data. This provided valuable information that helped understand the behavior of processes, the areas of memory they use, the shared library files they load.

Figure 9 shows a textual representation of the virtual memory of processes running on the system. It provides detailed information about how memory is allocated and used by these specific processes.

```
csi@siftworkstation: ~/volatility
$ python vol.py -f /home/csi/Desktop/E001.raw --profile=Linuxubuntu_5_4_0-91-genericx64 linux_proc_maps

0xffff998d099fdd00     1702 caja            0x00007f112372c000 0x00007f112372d000 ---              0x4e5000    253      0     915234 /usr/lib/x86_64-linux-gnu/libpython3.8.so.1.0
0xffff998d099fdd00     1702 caja            0x00007f112372d000 0x00007f1123733000 r--              0x4e5000    253      0     915234 /usr/lib/x86_64-linux-gnu/libpython3.8.so.1.0
0xffff998d099fdd00     1702 caja            0x00007f1123733000 0x00007f112377a000 rw-              0x4eb000    253      0     915234 /usr/lib/x86_64-linux-gnu/libpython3.8.so.1.0
0xffff998d099fdd00     1702 caja            0x00007f112377a000 0x00007f112379d000 rw-                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f112379d000 0x00007f112379e000 ---                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f112379e000 0x00007f1123f9e000 rw-                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f1123f9e000 0x00007f1124000000 r--                   0x0    253      0     524144 /usr/share/glib-2.0/schemas/gschemas.compiled
0xffff998d099fdd00     1702 caja            0x00007f1124000000 0x00007f1124025000 rw-                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f1124025000 0x00007f1128000000 ---                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f1128000000 0x00007f1128041000 rw-                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f1128041000 0x00007f112c000000 ---                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f112c000000 0x00007f112c021000 rw-                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f112c021000 0x00007f1130000000 ---                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f1130000000 0x00007f1130002000 r--                   0x0    253      0     287927 /usr/share/icons/Mint-Y-Brown/icon-theme.cache
0xffff998d099fdd00     1702 caja            0x00007f1130002000 0x00007f1130004000 r--                   0x0    253      0     291678 /usr/share/icons/Mint-Y-Orange/icon-theme.cache
0xffff998d099fdd00     1702 caja            0x00007f1130004000 0x00007f1130017000 r--                   0x0    253      0     923803 /usr/lib/x86_64-linux-gnu/girepository-1.0/Atk-1.0.t
ypelib
0xffff998d099fdd00     1702 caja            0x00007f1130017000 0x00007f1130026000 r--                   0x0    253      0     923815 /usr/lib/x86_64-linux-gnu/girepository-1.0/GObject-2
.0.typelib
0xffff998d099fdd00     1702 caja            0x00007f1130026000 0x00007f1130027000 ---                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f1130027000 0x00007f1130827000 rw-                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f1130827000 0x00007f113085e000 rw-                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f113085e000 0x00007f1130860000 r--                   0x0    253      0     291964 /usr/share/icons/Mint-Y-Yellow/icon-theme.cache
0xffff998d099fdd00     1702 caja            0x00007f1130860000 0x00007f1130869000 r--                   0x0    253      0     332695 /usr/share/icons/hicolor/icon-theme.cache
0xffff998d099fdd00     1702 caja            0x00007f1130869000 0x00007f1130886000 r--                   0x0    253      0     280262 /usr/share/icons/Adwaita/icon-theme.cache
0xffff998d099fdd00     1702 caja            0x00007f1130886000 0x00007f1130889000 r--                   0x0    253      0     920559 /usr/lib/x86_64-linux-gnu/girepository-1.0/GdkX11-3.
0.typelib
0xffff998d099fdd00     1702 caja            0x00007f1130889000 0x00007f11308aa000 rw-                   0x0      0      0          0
0xffff998d099fdd00     1702 caja            0x00007f11308aa000 0x00007f11308ab000 r--                   0x0    253      0     465438 /var/cache/fontconfig/d81b95d5-53a1-43de-af14-bdfbb9
f52820-le64.cache-7
0xffff998d099fdd00     1702 caja            0x00007f11308ab000 0x00007f11308bc000 r--                   0x0    253      0     923853 /usr/lib/x86_64-linux-gnu/girepository-1.0/Pango-1.0
```

Figure 9. Result of the linux_proc_maps command.

- Filescan: This Volatility "filescan" command was used to search and scan memory for file-like objects or file structures. It was useful when searching for file-related artifacts in memory.
- Timeliner: The "timeliner" command was used to create a timeline of events based on information retrieved from memory. She was able to create a temporal report organizing events or information retrieved from memory based on their timestamp, allowing us to better understand the chronological order of activities or changes in the system.

The results highlighted the critical importance of capturing the memory of a virus-infected system while detailing a thorough analysis of active processes and network connections to counter this threat. They demonstrate the superiority of the analysis of RAM compared to that of the hard drive in terms of efficiency and precision during investigations. Also, the method of extracting RAM using specific tools made it possible to identify the virus's activity patterns and analyze its operating modes. Volatile memory forensics are at the forefront of forensic [9].

6. Conclusion

This part of our investigations highlights the invaluable value of RAM analysis in combating malware attacks. By exploring memory capture, detailed analysis of running processes, and examination of network connections, this research demonstrates the relevance and effectiveness of this approach compared to traditional hard drive analysis. This approach focused on RAM analysis, validated by the results of this study, proves to be a crucial pillar in security investigations to detect, counter and neutralize malware attacks, thus strengthening response capabilities and protection of computer systems against these threats. To combat and identify the attacks, digital forensics plays a crucial role in cyber investigations. In particular, memory forensics helps by unhiding the tons of hidden secret information [10].

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Horan, C. and Saiedian, H. (2021) Cyber Crime Investigation: Landscape, Challenges, and Future Research Directions. *Journal of Cybersecurity and Privacy*, **1**, 580-596. https://doi.org/10.3390/jcp1040029

[2] Wu, Y., Xiang, D., Gao, J. and Wu, Y. (2019) Research on Investigation and Evidence Collection of Cybercrime Cases. *Journal of Physics*: *Conference Series*, **1176**, Article 042064. https://doi.org/10.1088/1742-6596/1176/4/042064

[3] Fernández-Fuentes, X., F. Pena, T. and Cabaleiro, J.C. (2022) Digital Forensic Analysis Methodology for Private Browsing: Firefox and Chrome on Linux as a Case Study. *Computers & Security*, **115**, Article 102626. https://doi.org/10.1016/j.cose.2022.102626

[4] Cox, J. and Bednar, P. (2018) Potential Difficulties during Investigations Due to Solid

State Drive (SSD) Technology. In: *Lecture Notes in Information Systems and Organisation*, Springer, 75-91. https://doi.org/10.1007/978-3-319-90503-7_7

[5] Chayal, N.M., Saxena, A. and Khan, R. (2022) A Review on Spreading and Forensics Analysis of Windows-Based Ransomware. *Annals of Data Science*, **11**, 1503-1524. https://doi.org/10.1007/s40745-022-00417-5

[6] Mohanta, A. and Saldanha, A. (2020) Memory Forensics with Volatility. In: *Malware Analysis and Detection Engineering*, Apress, 433-476. https://doi.org/10.1007/978-1-4842-6193-4_14

[7] Zhang, W., Li, X. and Zhu, T. (2023) Entropy and Memory Forensics in Ransomware Analysis: Utilizing LLaMA-7B for Advanced Pattern Recognition.

[8] Kävrestad, J., Birath, M. and Clarke, N. (2024) Malware Analysis. In: *Texts in Computer Science*, Springer, 227-238. https://doi.org/10.1007/978-3-031-53649-6_21

[9] Mohamed, A. and Saad, T. (2022) Automate Memory Forensics Investigation.

[10] Paul Joseph, D. and Norman, J. (2019) A Review and Analysis of Ransomware Using Memory Forensics and Its Tools. In: *Smart Innovation, Systems and Technologies*, Springer, 505-514. https://doi.org/10.1007/978-981-13-9282-5_48