Detecting Malware Infected Machines with Digital Forensic Analysis

Sangeetha G,
Associate Professor, Department of Artificial Intelligence and Data Science,
R.M.K College of Engineering and Technology, R& D Tiruvallur District, Tamil Nadu, India.
gsangeethakarthik@gmail.com

Indumathi A,
Assistant Professor,
Department of Information Technology, Sri Venkateswara
College of Engineering, Sriperumbudur

Usha S,
Kongu Engineering, College Perundurai,India

Anithaashri TP,
Professor, Saveetha School of Engineering

R Anto Arockia,
Assistant Professor, SRM Institute of Science and Technology, Kattangulattur, Chennai

Rosaline Balamurugan AG,
Assistant Professor, Department of Computer Science Vel Tech Rangarajan,
Dr. S. Agunthala Institute of Scienceand Technology
balamurugan.cse.ap@gmail.com

Article Info
Page Number:907 - 916
Publication Issue:
Vol 70 No. 2 (2021)

Article History
***Article Received**: 05 September 2021*
***Revised**: 09 October 2021*
***Accepted**: 22 November 2021*
***Publication**: 26 December 2021*

Abstract

The majority of significant intimidation on the Internet is malware which means malicious software. The Third-party (or attacker) will install the malware software program on the machine without the awareness of the owner to steal their private data. Day-by-Day the third party launches new malware, which leads to a great challenge to the malware detectors. Man-in-the-Browser (MB) attack is one of the special attacks in Man-in-the-Middle (MM), which targets the Internet backing customers. This work examined forensic analysis of Random Access Memory (RAM) and Volatile data infected machines. By using open source tools, the activities of the malware, the cause of the attacks and time periods are identified.

Keywords: Man-in-the-Browser, Random Access Memory, Man-in-the-Middle, Volatile data

1. Introduction

Digital Forensics science develops as an effect of the growth of technology and as such should proceed to progress to covert the investigation of new use cases for the prosecution of cybercriminals. For illustration, the situation of the Internet of Things (IoT) standard takes to the cybercrime incident infinite various devices for which there are not precise digital forensics techniques to extend and examine the digital evidence. Some clarifications have been issued in the earlier period, but they are still very limited and hard to work as a general structure for the digital forensic community. The conventional method to digital forensics is stagnant analysis. This method is mostly applied, has grooved methods, and formed official legality of proof gathered. In stagnant analysis, a forensically legal photocopy of every storage space medium of the shout downed system is ready made. Gears for medium investigation are victimized to seek for digital proof. The specified tools are high-quality at positioning files and finding their substance. File formation and alteration times are able to constitute. Removed files frequently may be found to various extents. Additional exciting information like email records, browsing history, and mounted script also is found. Stagnant analysis has some drawbacks. The major one thing is it cannot afford an absolute image of events. In Man-in-the-Browser, the performers install a Trojan horse on the system of victims which will modify the web transaction of the victim user. The man-in-the-middle attack is also a kind of cyber attack. In this the aggressors furtively interrupts and pass information among two persons who think they are conversing directly with each one.

The research work [8] states the most important confining features. To do stagnant analysis the end system wants to be locked up. This activity can be accomplished with a correct halt progression or just by dragging a power plug. The next method breaks feasible implementation of codes set up to take away the proof in a case of the halt progression. It could affect incompatible data and disk status in the write down cache.

In the above suit of shutdown the intention method, its active status is unavoidably mislaid. The information of volatile memory is not cured, but for potential information sheeted to disk. Additional applicable active information like established network connections, process list, installed kernel modules and open network ports, may not be analyzed with stagnant analysis. As all the mentioned things could apply to the analysis being accomplished, stagnant analysis offers partial proof. Encryption builds contact with information hoarded in volumes or encrypted files, much difficult if not impractical. Encryption keys, exploited in the system process for fetching encrypted data, are deserted; formerly the system is halted.

An often ignored disadvantage of static analysis is that it is difficult for a normal user. A system cannot be utilized even as forensic replicas of medium are completed. For some systems that demand more accessibility, this might entirely stop stagnant analysis. A secondary to stagnant analysis, or to be an additional fine harmonizing method, is live investigation. In this environment, proofs are gathered while the system is in progress. Live analysis works on several of the problems of stagnant analysis. Conversely, there are various questions in live analysis. The significant problem is that the live analysis of some action the forecaster implements originate from permanent modification of the examined system status [9]. The transform of the system state is beside recognized ethics in forensics that proof must not be modified and could origin assessment effects not to be provable and repeatable [10]. Constraints inflicted on the suitability of proof gathered by live analysis based on the valid authorized system.

There are few problems with live analysis. Experts could not have a proper stage of rights to the examined system. In the committee system, its reliability is doubtful. An assailant might have altered the structure in a method that holes finding of assault and alterations. This is particularly correct if the examined user interface is utilized. Still exploitation of recognized binaries from CD/DVD or extra trustworthy medium might not show the correct information because some of the forensic tools rely on the information given aside the kernel or the file system that could have been fiddled with.

2. Related work

Zhang[1] proposed a technique to prevent digital forensics action, a new anti-memory forensic technique that has been developed by misusing two architectural features of modern computers. One is PA(Physical Address) remapping is abused in Hidden in I/O Space (HIveS) and the other one is Hardware aided protected containers are abused in Malicious Enclave Software (Malclaveware) to give memory introspection protection for malware and encryption. To defend the ajar HIveS memory beside memory forensics, two new methods had been planned: TLB Camouflage and Blackbox Write. Blackbox Write modifies only write down right to the HIveS memory by building unsymmetrical write and read goals among the I/O and the memory space. TLB Camouflage efforts TLB cache incoherency along with multi-core processors to assure privileged write and read rights for a single processor core to the HIveS memory. They applied the idea of Malclaveware to ransomware to defend file encryption key which is not achievable to detect key by any advanced privilege forensics subsystem.

Most of the security sandbox technology is extremely interdependent on hypervisor types and their variants. Tien [2] proposed a novel sandbox system, using memory forensics methods, to give a cause-less sandbox result that is autonomous of the hypervisor. The VM introspection system is used to supervise malware progress memory information exterior the VM and investigate its model activities such as registry, file, network activities, and process. It compares the guest VM memory status after and before the malware implementation.

The JVM transcript objects from the location to another place in the garbage collection and fails to copy the existing old data on time. Adam Pridgen [3] projected a memory investigation method for restricted environments, specifically HotSpot JVM. It is a memory investigation model and also assists for digital forensics.

Most of the security sandbox technology is extremely dependent on the VM hypervisor variety and their versions. Chin-Wei Tien[4] excused a new security sandbox system enclosed based on memory forensics method, which is in-depend of VM hypervisor and it does not demand the system calls. By using the VM introspection technique, malware executing memory data is observed and investigated its system behaviors like registry, process, network, and file in outside VM behavior. Two experiments have been done to design a secure sandbox. In the first experiment, advanced malware samples (including portable document format files, Microsoft Office documents, and executable files) have been seized and proved with log files of hypervisor-based sandbox CIA. In the next experimentation, chosen 8 malware (WSF), and HTML Application file style proved the log files with Cuckoo Sandbox and CaptureBAT. But, due to the fast vanishing process and network activities,the detection rate of scripting files is 75%.

A model had formed [5] to analyze complex malware by incorporating both memory forensic and stagnant investigation techniques. The model assessed three hundred valid malware instances and fetched a 95% identification rate. The proposed model analyzes malware in three phases. These phases are typical memory investigation, trigger-based memory investigation and interval-based memory investigation. Typical memory investigation: After executing the malware in a restricted setting the applicable dump files are gathered for analysis. The malware unsubtle memory section before it changes and doesn’t leave any traces of the malware, so the classic memory analysis cannot discover any important data in its analysis. Trigger-based memory investigation: Contained setting is organized for Instrumentation, API, and Performance based implementation levels. Interval-based memory investigation: In every 30 seconds in 3 minutes the memory dumps are gathered for investigation. This technique also does not time off any outlines to tricks of the malware. In the virtual environment, static analysis is executed on malware samples and created Memory.dmp using cuckoo sandbox. The result is compared with the result of www.virustotal.com which is an online memory analysis tool. It is difficult to set up the situation and make memory dumps, the volume of “memory.dmp” may exceed the margin of 1GB for a few cases.

In the memory investigation, pool tag scanning [6] is utilized to determine kernel object allotment. Pool quick scanning will be done to only physical memory pages which are modules of memory team allocation. It decreases the bandwidth necessities of achieving live memory investigation on a suspected system over a system. But, this framework cannot be used to place allotments that are not correlated in the virtual address space of the kernel.

New hibernation file system [7] is defined and offered in Windows 8, 8.1, and 10. Some modifications in shutdown behavior of Windows and hibernation have been discussed. This hiberfil.sys file will be useful sources of evidence in digital forensic. Systems that are powered off in this method or by shutdown /s /hybrid statement will hold limited hibernation information. We have to take extra care when shutting down a suspected computer to guarantee that the hibernation information is presented.

3. VOLATILE MEMORY INVESTIGATION

A modern future to live analysis is an outlook to build a junk of a volatile memory for offline investigation. Volatile memory investigation displays an outlook in that the simple basis of proof is a memory dump. A specialist can then make the case by investigating the physical memory junk in a separate situation that is reticent to the proof. This movement codes for a few problems in live analysis. It controls importance to the hazarded structure, the investigation is observable and it is achievable to enquire novel queries shortly. Besides, offline volatile memory investigation does not believe on the OS of probably cooperation machines. The necessity for the particular device that should be suitable to the structure being secured ahead of something bad occurring creates this method infeasible in the universal case, but it had begun a lot of study committed to the investigation of memory junks. Some reports are dedicated to Windows memory analysis [12][13][14][15].

Memory Forensics is the investigation of images of memory retrieved from the jetting system. In the paper, we study in what way to utilize Memory Forensic Toolkits like Volatility to investigate the memory units with realistic forensic scenarios.

Memory forensics can assist in fetching forensics units from memory units of a system such as network connections, loaded modules, running process, etc. It is able to assist in rootkit detection, reverse engineering, and unpacking. The steps engaged in memory forensics:

Memory Acquisition: It needs dumping memory of the suspected machine. Acquiring memory images from a physical machine by using tools such as Win32dd/Win64dd, FastDump, DumpIt , and Memoryze is easy. It can be done by hanging the VM and seizing the ".vmem" file.

Memory Investigation: After a memory copy is fetched, then investigate the seized memory for forensic artifact tools such as Memoryze and Volatility may be utilized to investigate the memory.

Memory forensic assists to collect the subsequent information: (i) Listing the current and past network links (ii) Listing the all currently executing process (iii) Listing all DLL’s loaded (iv) Keystrokes pressed (v) unfilled / Unencrypted edition of the malware file (vi) listing files related to a process (vii) listing registry keys related to a process (viii) Listing Kernel sections (ix) Script inoculation (x) Rootkit recognition and (xi) Discover unknown artifacts

These images may be in any format like: (i) Page File (ii) Crash Dump (iii) Raw Format (iv) Hibernation File etc. There are different software/tools existing such as MoonSols, Belkasoft RAM, and seizure that support the attainment of the image. In the page files, recollect which can be utmost 16-page files in a computer, as a result formerly the image is retrieved the forecaster should verify all acquirable page files. In the Hibernation file, prior to the forecaster beginning investigating the image, it is necessary to be uncompressed. In addition, for fetching the image of a VM a snap is the greatest method to begin, but remember that there are other files other than the snapshot which might contain some relevant data.

After the image is fetched the next need to guarantee that the image detection is done. In general software packages such as Volatility check for KDBG chunks to discover the image Service Pack and OS. Since this chunk trails loaded modules and process list, you can also discover data like the number of loaded modules directly from such a high level, number of active processes. After the outline is taken, next we begin determining further objects from the image like the currently executing process, loaded dlls, active network links at the moment of image acquisition. More footprints are collected at the time. Volatility is a precocious memory forensic model developed in python. It is comfortable on multiple operating systems (Linux, Mac OS X, Windows,)

4. Experimental Result

When the Digital Forensics group met the victim's party the alleged contaminated system was in running mode, the group investigated it in two steps [11] : Step 1: Gathering digital proofs Step 2: Investigation of gathered digital proof.

4.1 Gathering digital proofs

The Digital Forensic group initially analyzed the RAM in.dd structure onto a forensically infertile medium by FTK Imager. The group also gathered event log files, internet history, volatile data and registry files from the alleged system such as the running processes /programs, list of network links, and dll files moved from the suspected system. The reliability of digital proofs is reserved during the full examination by creating the hash value, the digital fingerprint of the proofs. The software exploited for gathering the proofs are the Digital Evidence Forensic Tool kit (DEFT) and FTK Imager.

4.2 Investigation of gathered digital proofs

Volatility is the greatest open source software for examining RAM in 64 bit/32 bit systems. It assists investigation for Windows, Linux, Android systems, and Mac. It is founded on Python and may be executed on Linux, Windows, and Mac systems. It can examine raw VMware dumps (.vmem), dumps, virtual box dumps, crash dumps, and many others.

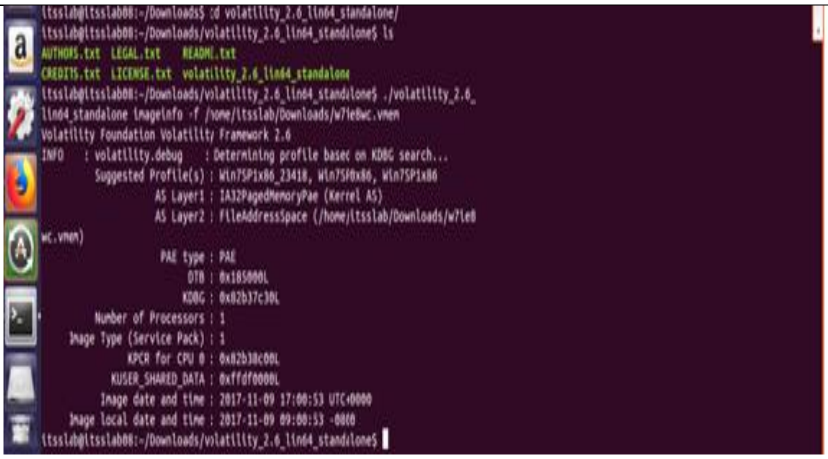
Volatility is necessary to identify what kind of system your memory dump arrived from, so it undergoes which algorithms, data structures, and representations to use. A default summary of WinXPSP2x86 is located internally. These commands are used to denote the service pack, operating system, and system design structure (32 or 64 bit), but it also includes additional practical information such as the timestamp of sample evidence and DTB address were collected.

```
* To list of an existing plugins
                python vol.py –h
* To illustrate the memory dump for examining
python vol.py -f mem.dmp --profile=WinXPSP3x86
```

```
* To identify the --profile information
 python vol.py -f mem.dmp imageinfo
```

Step 1: Identification of hardware and software information

```
./volatility_2.6_lin64_standalone  imageinfo  –f /home/itsslab/Downloads/w7ie8wc.vmem
```



```
itsslab@itsslab08:~/Downloads$ cd volatility_2.6_lin64_standalone/
itsslab@itsslab08:~/Downloads/volatility_2.6_lin64_standalone$ ls
AUTHORS.txt  LEGAL.txt    README.txt
CREDITS.txt  LICENSE.txt  volatility_2.6_lin64_standalone
itsslab@itsslab08:~/Downloads/volatility_2.6_lin64_standalone$ ./volatility_2.6_
lin64_standalone imageinfo -f /home/itsslab/Downloads/w7ie8wc.vmem
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/itsslab/Downloads/w7ie8
wc.vmem)
                      PAE type : PAE
                           DTB : 0x185000L
                          KDBG : 0x82b37c30L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0x82b38c00L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2017-11-09 17:00:53 UTC+0000
     Image local date and time : 2017-11-09 09:00:53 -0800
itsslab@itsslab08:~/Downloads/volatility_2.6_lin64_standalone$
```

Figrue. 1 Identification of hardware and software information

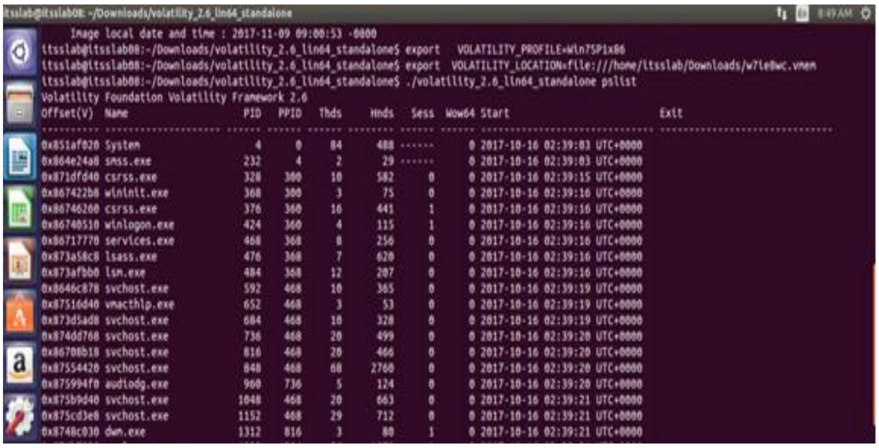
Volatility is necessary to know what kind of system memory dump used, so it undergoes which algorithms, data structures, and representations to use. A default summary of WinXPSP2x86 is located internally. This rule is applied to set the service pack, system design structure (32 or 64 bit), operating system, and but it also includes additional practical information such as the timestamp of sample evidence and DTB address were collected.

Step 2: Setting Environment variables

```
Export   VOLATILITY_PROFILE=Win7SP1x86

export  VOLATILITY_LOCATION=file:///home/itsslab/Downloads/w7ie8wc.vmem

./volatility_2.6_lin64_standalone  pslist
```



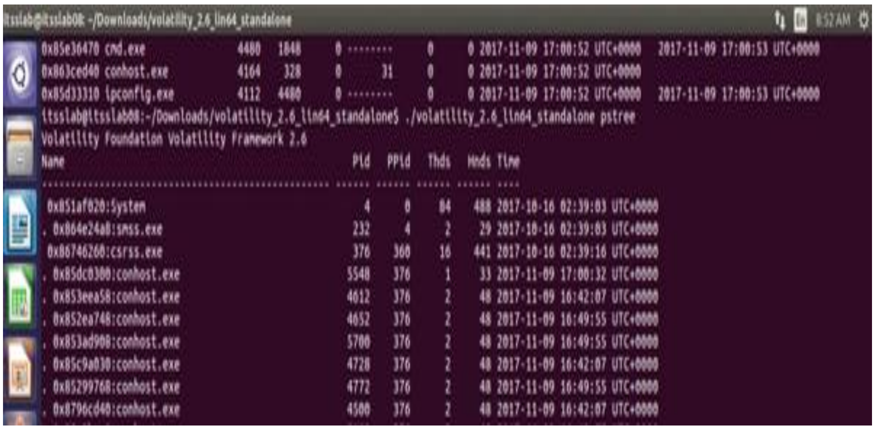
```
itsslab@itsslab08: ~/Downloads/volatility_2.6_lin64_standalone
     Image local date and time : 2017-11-09 09:00:53 -0800
itsslab@itsslab08:~/Downloads/volatility_2.6_lin64_standalone$ export   VOLATILITY_PROFILE=Win7SP1x86
itsslab@itsslab08:~/Downloads/volatility_2.6_lin64_standalone$ export  VOLATILITY_LOCATION=file:///home/itsslab/Downloads/w7ie8wc.vmem
itsslab@itsslab08:~/Downloads/volatility_2.6_lin64_standalone$ ./volatility_2.6_lin64_standalone pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x851af020 System                    4      0     84      488 ------      0 2017-10-16 02:39:03 UTC+0000
0x864e24a8 smss.exe                232      4      2       29 ------      0 2017-10-16 02:39:03 UTC+0000
0x871dfd40 csrss.exe               328    300     10      582      0      0 2017-10-16 02:39:15 UTC+0000
0x867422b8 wininit.exe             368    300      3       75      0      0 2017-10-16 02:39:16 UTC+0000
0x86746260 csrss.exe               376    360     16      441      1      0 2017-10-16 02:39:16 UTC+0000
0x86740510 winlogon.exe            424    360      4      115      1      0 2017-10-16 02:39:16 UTC+0000
0x86717770 services.exe            468    368      8      256      0      0 2017-10-16 02:39:16 UTC+0000
0x873a58c8 lsass.exe               476    368      7      620      0      0 2017-10-16 02:39:16 UTC+0000
0x873afbb0 lsm.exe                 484    368     12      207      0      0 2017-10-16 02:39:16 UTC+0000
0x8646c878 svchost.exe             592    468     10      365      0      0 2017-10-16 02:39:19 UTC+0000
0x87516d40 vmacthlp.exe            652    468      3       53      0      0 2017-10-16 02:39:19 UTC+0000
0x873d5ad8 svchost.exe             684    468     10      328      0      0 2017-10-16 02:39:19 UTC+0000
0x874dd768 svchost.exe             736    468     20      499      0      0 2017-10-16 02:39:20 UTC+0000
0x86708b18 svchost.exe             816    468     20      466      0      0 2017-10-16 02:39:20 UTC+0000
0x87554420 svchost.exe             848    468     68     2760      0      0 2017-10-16 02:39:20 UTC+0000
0x875994f0 audiodg.exe             960    736      5      124      0      0 2017-10-16 02:39:20 UTC+0000
0x875b9d40 svchost.exe            1048    468     20      663      0      0 2017-10-16 02:39:21 UTC+0000
0x875cd3e8 svchost.exe            1152    468     29      712      0      0 2017-10-16 02:39:21 UTC+0000
0x8748c030 dwm.exe                1312    816      3       80      1      0 2017-10-16 02:39:21 UTC+0000
```

Figrue. 2 Listing pslist

This plug-in provides us the option to display all running processes on the particular system during which the memory dump was seized. Some essential files have been found in the analysis of Explores.exe,Wanacry, and Tasksche.

Step 3: Listing pstree

```
./volatility_2.6_lin64_standalone  pstree
```

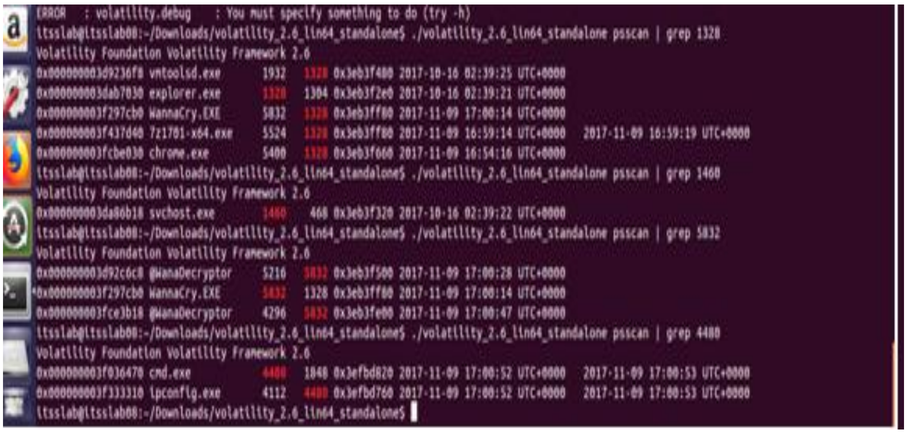


```
ltsslab@ltsslab08: ~/Downloads/volatility_2.6_lin64_standalone                8:52 AM
0x85e36470 cmd.exe          4480   1848    0 --------     0     0 2017-11-09 17:00:52 UTC+0000   2017-11-09 17:00:53 UTC+0000
0x863ced40 conhost.exe      4164    328    0       31     0     0 2017-11-09 17:00:52 UTC+0000
0x85d33310 ipconfig.exe     4112   4480    0 --------     0     0 2017-11-09 17:00:52 UTC+0000   2017-11-09 17:00:53 UTC+0000
ltsslab@ltsslab08:~/Downloads/volatility_2.6_lin64_standalone$ ./volatility_2.6_lin64_standalone pstree
Volatility Foundation Volatility Framework 2.6
Name                                          Pid   PPid   Thds   Hnds Time
-------------------------------------------- ------ ------ ------ ------ ----
 0x851af020:System                              4      0     84    488 2017-10-16 02:39:03 UTC+0000
. 0x864e24a0:smss.exe                         232      4      2     29 2017-10-16 02:39:03 UTC+0000
 0x86746260:csrss.exe                         376    360     16    441 2017-10-16 02:39:16 UTC+0000
. 0x85dc0300:conhost.exe                     5548    376      1     33 2017-11-09 17:00:32 UTC+0000
. 0x853eea58:conhost.exe                     4612    376      2     48 2017-11-09 16:42:07 UTC+0000
. 0x852ea748:conhost.exe                     4652    376      2     48 2017-11-09 16:49:55 UTC+0000
. 0x853ad908:conhost.exe                     5700    376      2     48 2017-11-09 16:49:55 UTC+0000
. 0x85c9a030:conhost.exe                     4728    376      2     48 2017-11-09 16:42:07 UTC+0000
. 0x85299768:conhost.exe                     4772    376      2     48 2017-11-09 16:49:55 UTC+0000
. 0x8796cd40:conhost.exe                     4500    376      2     48 2017-11-09 16:42:07 UTC+0000
```

Figrue. 3 Listing pstree

Step 4 : Identification of particular process _id

```
./volatility_2.6_lin64_standalone  Psscan | grep   pid
```

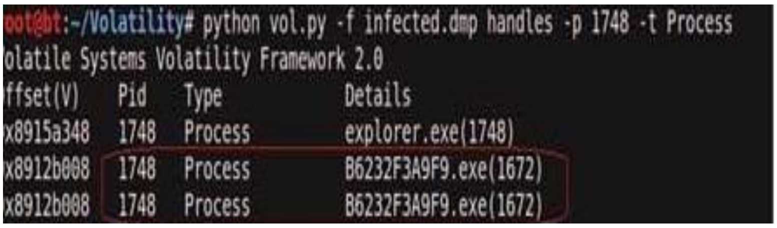


```
ERROR   : volatility.debug    : You must specify something to do (try -h)
ltsslab@ltsslab08:~/Downloads/volatility_2.6_lin64_standalone$ ./volatility_2.6_lin64_standalone psscan | grep 1328
Volatility Foundation Volatility Framework 2.6
0x000000003d9236f8 vmtoolsd.exe      1932   1328 0x3eb3f480 2017-10-16 02:39:25 UTC+0000
0x000000003dab7030 explorer.exe      1328   1304 0x3eb3f2e0 2017-10-16 02:39:21 UTC+0000
0x000000003f297cb0 WannaCry.EXE      5832   1328 0x3eb3ff80 2017-11-09 17:00:14 UTC+0000
0x000000003f437d40 7z1701-x64.exe    5524   1328 0x3eb3ff80 2017-11-09 16:59:14 UTC+0000   2017-11-09 16:59:19 UTC+0000
0x000000003fcbe030 chrome.exe        5400   1328 0x3eb3f660 2017-11-09 16:54:16 UTC+0000
ltsslab@ltsslab08:~/Downloads/volatility_2.6_lin64_standalone$ ./volatility_2.6_lin64_standalone psscan | grep 1460
Volatility Foundation Volatility Framework 2.6
0x000000003da86b18 svchost.exe       1460    468 0x3eb3f320 2017-10-16 02:39:22 UTC+0000
ltsslab@ltsslab08:~/Downloads/volatility_2.6_lin64_standalone$ ./volatility_2.6_lin64_standalone psscan | grep 5832
Volatility Foundation Volatility Framework 2.6
0x000000003d92c6c8 @WanaDecryptor    5216   5832 0x3eb3f500 2017-11-09 17:00:28 UTC+0000
0x000000003f297cb0 WannaCry.EXE      5832   1328 0x3eb3ff80 2017-11-09 17:00:14 UTC+0000
0x000000003fce3b18 @WanaDecryptor    4296   5832 0x3eb3fe00 2017-11-09 17:00:47 UTC+0000
ltsslab@ltsslab08:~/Downloads/volatility_2.6_lin64_standalone$ ./volatility_2.6_lin64_standalone psscan | grep 4480
Volatility Foundation Volatility Framework 2.6
0x000000003f036470 cmd.exe           4480   1848 0x3efbd820 2017-11-09 17:00:52 UTC+0000   2017-11-09 17:00:53 UTC+0000
0x000000003f333310 ipconfig.exe      4112   4480 0x3efbd760 2017-11-09 17:00:52 UTC+0000   2017-11-09 17:00:53 UTC+0000
ltsslab@ltsslab08:~/Downloads/volatility_2.6_lin64_standalone$
```

Figrue. 4 Identification of particular process _id

Step 5: Handles of explorer.exe

The explorer.exe was establishing links to the malevolent IP, there is a hypothesis that explorer.exe is abscessed. It is one of the Operating System processes. Let's look at the procedure of explorer.exe. The following screen displays Explorer.exe opens a handle to the B623F3A9F9 (1672), saying explorer.exe could have made that process, which could also be malevolent.

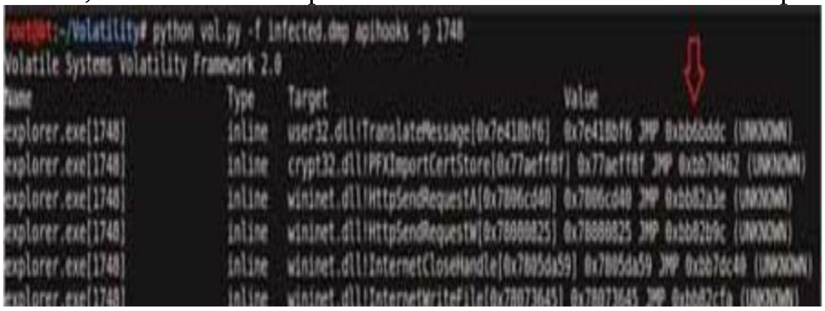


```
oot@bt:~/Volatility# python vol.py -f infected.dmp handles -p 1748 -t Process
olatile Systems Volatility Framework 2.0
ffset(V)    Pid    Type       Details
x8915a348   1748   Process    explorer.exe(1748)
x8912b008   1748   Process    B6232F3A9F9.exe(1672)
x8912b008   1748   Process    B6232F3A9F9.exe(1672)
```

Figrue. 5. Detection of explorer.exe

Step 6 : API hooks

API hooks unit illustrate, API hooks in explorer.exe and move to an indefinite place

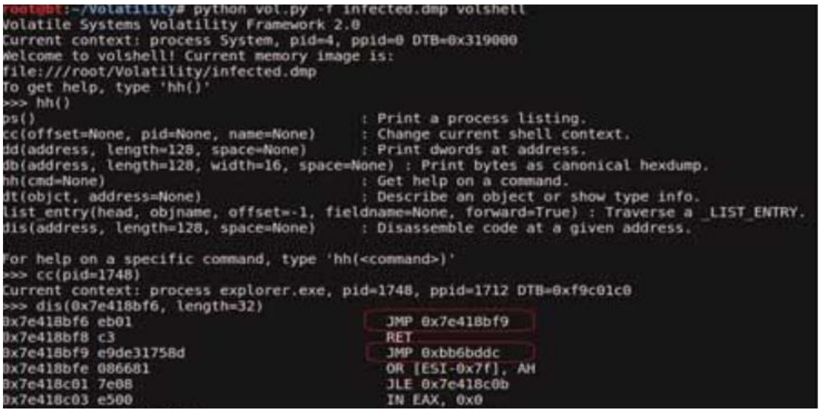


```
root@bt:~/Volatility# python vol.py -f infected.dmp apihooks -p 1748
Volatile Systems Volatility Framework 2.0
Name                  Type    Target                                        Value
explorer.exe[1748]    inline  user32.dll!TranslateMessage[0x7e418bf6]  0x7e418bf6 JMP 0xbb6bddc (UNKNOWN)
explorer.exe[1748]    inline  crypt32.dll!PFXImportCertStore[0x77aeff8f] 0x77aeff8f JMP 0xbb70462 (UNKNOWN)
explorer.exe[1748]    inline  wininet.dll!HttpSendRequestA[0x7806c040] 0x7806c040 JMP 0xbb82a3e (UNKNOWN)
explorer.exe[1748]    inline  wininet.dll!HttpSendRequestW[0x78080825] 0x78080825 JMP 0xbb82b9c (UNKNOWN)
explorer.exe[1748]    inline  wininet.dll!InternetCloseHandle[0x7805da59] 0x7805da59 JMP 0xbb7dc40 (UNKNOWN)
explorer.exe[1748]    inline  wininet.dll!InternetWriteFile[0x78073645] 0x78073645 JMP 0xbb82cfa (UNKNOWN)
```

Figrue 6. API hooks

Step 7: Exploring the hooks

The hooks function (Translate Message) confirms a small jump and a lengthy jump to malware position



```
root@bt:~/Volatility# python vol.py -f infected.dmp volshell
Volatile Systems Volatility Framework 2.0
Current context: process System, pid=4, ppid=0 DTB=0x319000
Welcome to volshell! Current memory image is:
file:///root/Volatility/infected.dmp
To get help, type 'hh()'
>>> hh()
ps()                                     : Print a process listing.
cc(offset=None, pid=None, name=None)     : Change current shell context.
dd(address, length=128, space=None)      : Print dwords at address.
db(address, length=128, width=16, space=None) : Print bytes as canonical hexdump.
hh(cmd=None)                             : Get help on a command.
dt(objct, address=None)                  : Describe an object or show type info.
list_entry(head, objname, offset=-1, fieldname=None, forward=True) : Traverse a _LIST_ENTRY.
dis(address, length=128, space=None)     : Disassemble code at a given address.

For help on a specific command, type 'hh(<command>)'
>>> cc(pid=1748)
Current context: process explorer.exe, pid=1748, ppid=1712 DTB=0xf9c01c0
>>> dis(0x7e418bf6, length=32)
0x7e418bf6 eb01                          JMP 0x7e418bf9
0x7e418bf8 c3                            RET
0x7e418bf9 e9de31758d                    JMP 0xbb6bddc
0x7e418bfe 086681                        OR [ESI-0x7f], AH
0x7e418c01 7e08                          JLE 0x7e418c0b
0x7e418c03 e500                          IN EAX, 0x0
```

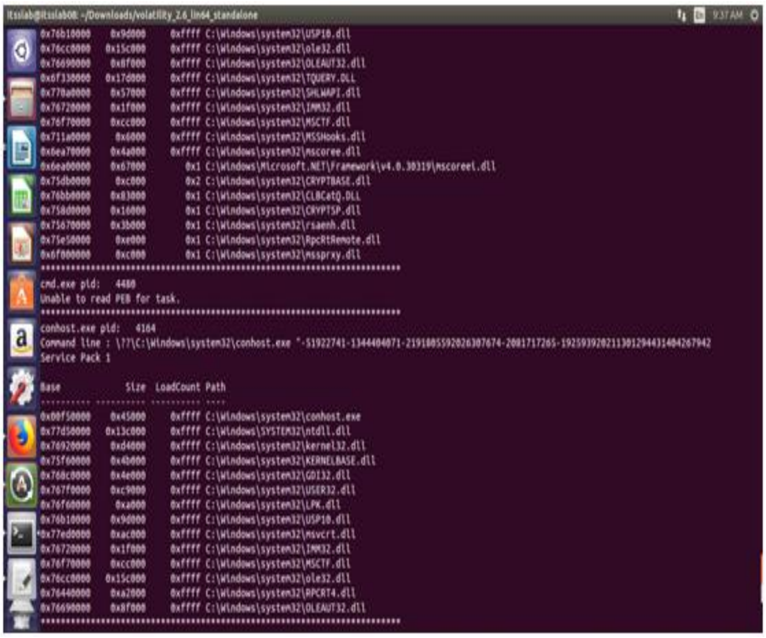
Figrue 7. Exploring the hooks

Writing the bytes at the hooked position, displays the occurrence of embedded executable file location in explorer.exe .

Step 8: Listing DLL files

To show the DLLs for all presently running processes or a particular process we use this plug-in.

```
itsslab@itsslab08:~/Downloads/volatility_2.6_lin64_standalone$
./volatility_2.6_lin64_standalone -f
/home/itsslab/Downloads//w7ie8wc.vmem --profile=Win7SP1x86 dlllist >
dlllist.txt
```

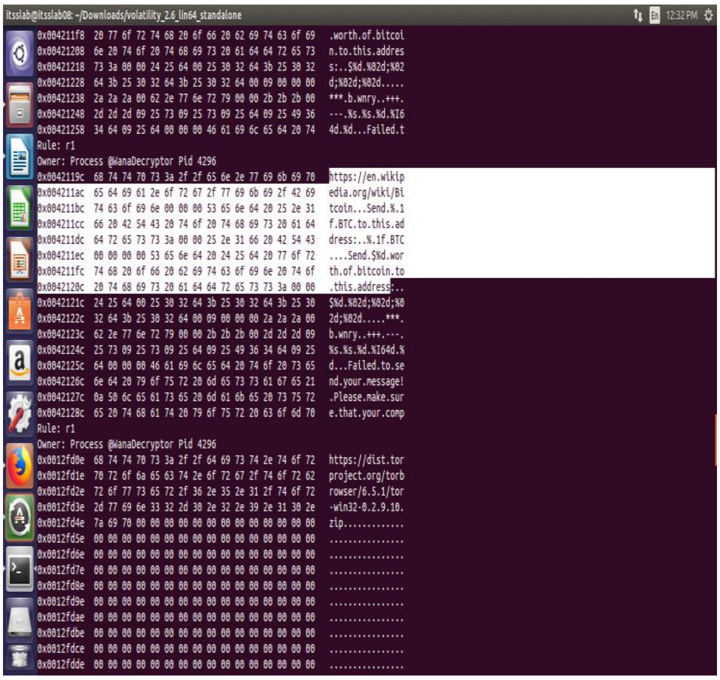


itsslab@itsslab08: ~/Downloads/volatility_2.6_lin64_standalone

```
0x76b10000    0x9d000    0xffff C:\Windows\system32\USP10.dll
0x76cc0000   0x15c000    0xffff C:\Windows\system32\ole32.dll
0x76690000    0x8f000    0xffff C:\Windows\system32\OLEAUT32.dll
0x6f330000   0x17d000    0xffff C:\Windows\system32\TQUERY.DLL
0x770a0000    0x57000    0xffff C:\Windows\system32\SHLWAPI.dll
0x76720000    0x1f000    0xffff C:\Windows\system32\IMM32.dll
0x76f70000    0xcc000    0xffff C:\Windows\system32\MSCTF.dll
0x711a0000     0x6000    0xffff C:\Windows\system32\MSSHooks.dll
0x6ea70000    0x4a000    0xffff C:\Windows\system32\mscoree.dll
0x6ea00000    0x67000       0x1 C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscoreel.dll
0x75db0000     0xc000       0x2 C:\Windows\system32\CRYPTBASE.dll
0x76bb0000    0x83000       0x1 C:\Windows\system32\CLBCatQ.DLL
0x758d0000    0x16000       0x1 C:\Windows\system32\CRYPTSP.dll
0x75670000    0x3b000       0x1 C:\Windows\system32\rsaenh.dll
0x75e50000     0xe000       0x1 C:\Windows\system32\RpcRtRemote.dll
0x6f000000     0xc000       0x1 C:\Windows\system32\mssprxy.dll
************************************************************************
cmd.exe pid:   4480
Unable to read PEB for task.
************************************************************************
conhost.exe pid:   4164
Command line : \??\C:\Windows\system32\conhost.exe "-51922741-1344404071-2191805592026307674-2081717265-192593920211301294431404267942
Service Pack 1

Base             Size  LoadCount Path
---------- ---------- ---------- ----
0x00f50000    0x45000    0xffff C:\Windows\system32\conhost.exe
0x77d50000   0x13c000    0xffff C:\Windows\SYSTEM32\ntdll.dll
0x76920000    0xd4000    0xffff C:\Windows\system32\kernel32.dll
0x75f60000    0x4b000    0xffff C:\Windows\system32\KERNELBASE.dll
0x768c0000    0x4e000    0xffff C:\Windows\system32\GDI32.dll
0x767f0000    0xc9000    0xffff C:\Windows\system32\USER32.dll
0x76f60000     0xa000    0xffff C:\Windows\system32\LPK.dll
0x76b10000    0x9d000    0xffff C:\Windows\system32\USP10.dll
0x77ed0000    0xac000    0xffff C:\Windows\system32\msvcrt.dll
0x76720000    0x1f000    0xffff C:\Windows\system32\IMM32.dll
0x76f70000    0xcc000    0xffff C:\Windows\system32\MSCTF.dll
0x76cc0000   0x15c000    0xffff C:\Windows\system32\ole32.dll
0x76440000    0xa2000    0xffff C:\Windows\system32\RPCRT4.dll
0x76690000    0x8f000    0xffff C:\Windows\system32\OLEAUT32.dll
************************************************************************
```

Figrue. 8 Listing DLL



itsslab@itsslab08: ~/Downloads/volatility_2.6_lin64_standalone

```
0x004211f8  20 77 6f 72 74 68 20 6f 66 20 62 69 74 63 6f 69   .worth.of.bitcoi
0x00421208  6e 20 74 6f 20 74 68 69 73 20 61 64 64 72 65 73   n.to.this.addres
0x00421218  73 3a 00 00 24 25 64 00 25 30 32 64 3b 25 30 32   s:..$%d.%02d;%02
0x00421228  64 3b 25 30 32 64 3b 25 30 32 64 00 09 00 00 00   d;%02d;%02d.....
0x00421238  2a 2a 2a 00 62 2e 77 6e 72 79 00 00 2b 2b 2b 00   ***.b.wnry..+++.
0x00421248  2d 2d 2d 09 25 73 09 25 73 09 25 64 09 25 49 36   ---.%s.%s.%d.%I6
0x00421258  34 64 09 25 64 00 00 00 46 61 69 6c 65 64 20 74   4d.%d...Failed.t
Rule: r1
Owner: Process @WanaDecryptor Pid 4296
0x0042119c  68 74 74 70 73 3a 2f 2f 65 6e 2e 77 69 6b 69 70   https://en.wikip
0x004211ac  65 64 69 61 2e 6f 72 67 2f 77 69 6b 69 2f 42 69   edia.org/wiki/Bi
0x004211bc  74 63 6f 69 6e 00 00 00 53 65 6e 64 20 25 2e 31   tcoin...Send.%.1
0x004211cc  66 20 42 54 43 20 74 6f 20 74 68 69 73 20 61 64   f.BTC.to.this.ad
0x004211dc  64 72 65 73 73 3a 00 00 25 2e 31 66 20 42 54 43   dress:..%.1f.BTC
0x004211ec  00 00 00 00 53 65 6e 64 20 24 25 64 20 77 6f 72   ....Send.$%d.wor
0x004211fc  74 68 20 6f 66 20 62 69 74 63 6f 69 6e 20 74 6f   th.of.bitcoin.to
0x0042120c  20 74 68 69 73 20 61 64 64 72 65 73 73 3a 00 00   .this.address:..
0x0042121c  24 25 64 00 25 30 32 64 3b 25 30 32 64 3b 25 30   $%d.%02d;%02d;%0
0x0042122c  32 64 3b 25 30 32 64 00 09 00 00 00 2a 2a 2a 00   2d;%02d.....***.
0x0042123c  62 2e 77 6e 72 79 00 00 2b 2b 2b 00 2d 2d 2d 09   b.wnry..+++.---.
0x0042124c  25 73 09 25 73 09 25 64 09 25 49 36 34 64 09 25   %s.%s.%d.%I64d.%
0x0042125c  64 00 00 00 46 61 69 6c 65 64 20 74 6f 20 73 65   d...Failed.to.se
0x0042126c  6e 64 20 79 6f 75 72 20 6d 65 73 73 61 67 65 21   nd.your.message!
0x0042127c  0a 50 6c 65 61 73 65 20 6d 61 6b 65 20 73 75 72   .Please.make.sur
0x0042128c  65 20 74 68 61 74 20 79 6f 75 72 20 63 6f 6d 70   e.that.your.comp
Rule: r1
Owner: Process @WanaDecryptor Pid 4296
0x0012fd0e  68 74 74 70 73 3a 2f 2f 64 69 73 74 2e 74 6f 72   https://dist.tor
0x0012fd1e  70 72 6f 6a 65 63 74 2e 6f 72 67 2f 74 6f 72 62   project.org/torb
0x0012fd2e  72 6f 77 73 65 72 2f 36 2e 35 2e 31 2f 74 6f 72   rowser/6.5.1/tor
0x0012fd3e  2d 77 69 6e 33 32 2d 30 2e 32 2e 39 2e 31 30 2e   -win32-0.2.9.10.
0x0012fd4e  7a 69 70 00 00 00 00 00 00 00 00 00 00 00 00 00   zip.............
0x0012fd5e  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0012fd6e  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0012fd7e  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0012fd8e  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0012fd9e  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0012fdae  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0012fdbe  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0012fdce  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0012fdde  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Figrue. 9. Identification of decryption information

Conclusion

Day-by-Day the third party launches new malware, which leads to a great challenge to the malware detectors. MB attack is one of the special attacks in Man-in-the-Middle, which targets the Internet backing customers. After observing forensic analysis of Random Access Memory and Volatile data of infected machines, the origin of the assault, timestamps (ctime and mtime) and the activities of the malware is identified by software tool Volatility. Memory forensics is an almighty method and along with the Volatility tool is able to discover and haul out the forensic objects from the memory that assists in incident response, malware analysis and reverse engineering.

Reference

[1] Ning Zhang, Ruide Zhang, Kun Sun, Wenjing Lou, Y. Thomas Hou, Sushil Jajodia" Memory Forensic Challenges Under Misused Architectural Features" IEEE Transactions on Information Forensics and Security (Volume: 13 , Issue: 9 , Sept. 2018).

[2] Chin-Wei Tien, Jian-Wei Liao, Shun-Chieh Chang, Sy-Yen Kuo," Memory forensics using virtual machine introspection for Malware analysis " 2017 IEEE Conference on Dependable and Secure Computing.

[3] Adam Pridgen, Simson Garfinkel, Dan S. Wallach," Picking up the trash: Exploiting generational GC for memory analysis" Digital Investigation 20 (2017).

[4] Chin-Wei Tien, Jian-Wei Liao, Shun-Chieh Chang, Sy-Yen Kuo," Memory forensics using virtual machine introspection for Malware analysis", 2017 IEEE Conference on Dependable and Secure Computing.

[5] Chathuranga Rathnayaka, Aruna Jamdagni," An Efficient Approach for Advanced Malware Analysis using Memory Forensic Technique" 2017 IEEE Trustcom/BigDataSE/ICESS.

[6] Joe Sylve, Vico Marziale, Golden Richard," Pool Tag: Quick Scanning for Windows Memory Analysis" DFRWS 2016 Europe — Proceedings of the Third Annual DFRWS Europe on Digital Investigation Volume 16, upplement, 29 March 2016,

[7] Joe T. Sylve, Vico Marziale, Golden G. Richard III, "Modern Windows Hibernation File Analysis", on Digital Investigation Volume 20, March 2017, Pages 16-22.

[8] B. Hay, M. Bishop, and K. Nance, "Live Analysis: Progress and Challenges," IEEE Security and Privacy, vol. 7, Mar. 2009, pp. 30-37.

[9] F. Adelstein, "Live forensics: diagnosing your system without killing it first," Commun. ACM, vol. 49, 2006, pp. 63-66.

[10] M.M. Pollitt, "Principles, practices, and procedures: an approach to standards in computer forensics," Second International Conference on Computer Evidence, 1995, pp. 10-15.

[11]. Kalaivani, K., & Sivakumar, R. (2016, September 1). A Novel Fuzzy Based Bio-Key Management scheme for Medical Data Security. Journal of Electrical Engineering and Technology. The Korean Institute of Electrical Engineers. https://doi.org/10.5370/jeet.2016.11.5.1509

[12] Schuster A, "Searching for processes and threads in Microsoft Windows memory dumps," Digital Investigation, vol. 3, Sep. 2006, pp. 10-16.

[13] A. Schuster, "Pool allocations as an information source in Windows memory forensics," International conference on IT-incident management and IT-forensics, 2006, pp. 104-115.

[14] Kornblum J.D, "Using every part of the buffalo in Windows memory analysis," Digital Investigation, vol. 4, Mar. 2007, pp. 24-29.

[15] Schuster A, "The impact of Microsoft Windows pool allocation strategies on memory forensics," Digital Investigation, vol. 5, 2008, pp. 58-64.