INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 4, Issue 4)
Available online at: www.ijariit.com

Hunting advanced volatile threats using memory forensics

Priya B Gadgil
p.gadgil@somaiya.edu
K. J. Somaiya College of Engineering, Mumbai, Maharashtra

Sangeeta Nagpure
sangeetanagpure@somaiya.edu
K. J. Somaiya College of Engineering, Mumbai, Maharashtra

ABSTRACT

Due to continuous growth in malware attacks, memory forensics has become very crucial as it contains many forensic artifacts that cyber forensic investigators cannot get through the traditional disk forensics. Forensic Analysis of a memory dump of victim's machine provides a detailed analysis of malware, checking traces of malware that have been created while running in the machine. Moreover, recent malware techniques also use stealthy methods to go undetected in typical disk forensics. Such techniques always execute exclusively from the memory or hide in the legitimate process to avoid the typical signature-based antivirus detection. Many of the recent studies also show that the percentage of such attacks have increased drastically. It is also estimated that the same trend will continue in the future and advanced threat like file less malware will become the major concern for the organizations as well as security researchers. This paper analyses memory forensics in the context of designing a forensic approach which will help to detect such advance malware threats. In this paper, we are analyzing a sample memory image infected by a malware. This paper proposes a generalized framework for doing step by step analysis of memory image for detecting fileless malware attacks.

Keywords— *Memory forensics, Advanced volatile threat*

1. INTRODUCTION

Computer forensic involves acquiring, processing, analyzing digital information and traces which can be used as evidence in administrative, civil or legal cases. Typically while working in the field of cyber forensics investigations one needs to unplug the suspect machine, make a forensically sound duplicate copy and work on the same copy. It is because the basic principle of digital forensics says ‘Never work on the Original Media'. Attackers carry out lifeless attacks that don’t drop malware on a victim’s system in order to work, and so easily evade detection. So to analyze these types of fileless attacks, one needs to carry out Memory Forensics.

In memory forensics, we need to capture the memory of the live suspect machine and ~~analyses~~ it to get the crucial information about the system. In this paper, one sample memory image of the infected system is analyzed using Memory Forensic tool ‘Volatility'. This is to demonstrate an analytical approach which can be helpful in detecting the advanced volatile threats. The main objective of our approach would be to find out the **‘Indicators of the Compromise’ (IOC’s) for the conformity of the malware detection.**

2. RESOURCES NEEDED

(i) Volatility Memory Forensic tool installed.
(ii) Captured Memory of the suspected machine.

3.DEMONSTRATION

A sample memory image is chosen to demonstrate how memory forensics can be useful in digging the traces of the malware. In the example, we will be analyzing to get enough Indicators of Compromises (IOC's). Using the tool ‘Volatility' following analysis was performed on the disk image.

Methodology for memory analysis for advanced volatile search as follows using volatility:

1. Identify Rogue Processes

a. Pslist
b. Psscan
c. Pstree
d. Pstotal

2. Analyze process DLLs and handles

a. Dlllist
b. Cmdline
c. Getsids
d. Handles
e. Filescan
f. Mutantscan
g. Svcscan
h. Cmdscan
i. Consoles

3. Review network artefacts

a. Connections
b. Connscan
c. Sockets
d. Sockscan
e. Netscan

4. Look for evidence of code injection

a. Malfind
b. Ldrmodules

5. Checks for signs of a rootkit

a. SSDT
b. Psxview
c. Modscan
d. Apihooks
e. Driverirp
f. Idt

6. Dump suspicious processes and drivers

a. Dlldump
b. Moddump
c. Procdump
d. Memdump
e. Dumpfiles
f. filescan

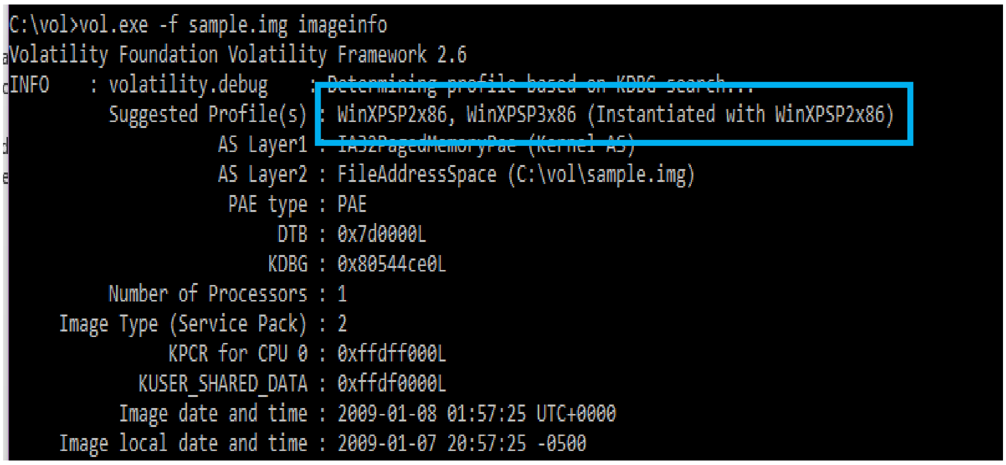
Above is the general framework that we can use to dig deep into the search of malware especially the volatile threats. To detect such threats is very challenging sometimes as it evades itself from many antivirus solutions as well as network level protection solutions. For the demonstration, only relevant output snaps are shown to prove the maliciousness of the captured memory. We should also note that every plugin listed above in each section will not work for every memory capture as some of the plugins will be the **'profile specific'.**

3.1 Step 1. Know the profile of the machine

It is very important first to determine the Profile (Operating system) of the system from where the memory image was captured, as the memory structure is different for different Windows Operating System.

The image info plugin identifies the Windows operating system version, the service pack, and the system architecture by locating the KDBG (Kernel Debugging Data Block) within the memory image.

The image info plugin also shows the date and time when the memory sample was collected.



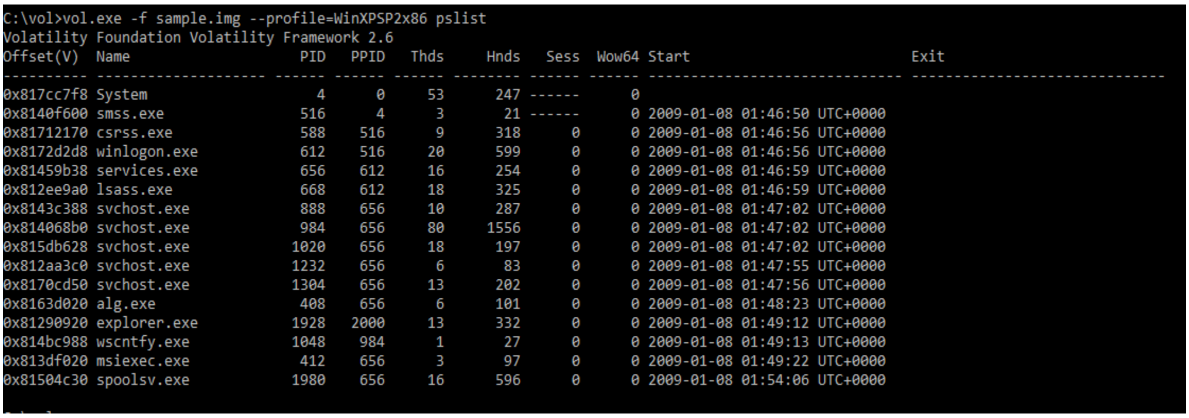
```
C:\vol>vol.exe -f sample.img imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (C:\vol\sample.img)
                      PAE type : PAE
                           DTB : 0x7d0000L
                          KDBG : 0x80544ce0L
          Number of Processors : 1
     Image Type (Service Pack) : 2
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2009-01-08 01:57:25 UTC+0000
     Image local date and time : 2009-01-07 20:57:25 -0500
```

Multiple profiles are suggested because the identified operating systems share many features that are common.

Using the Suggested Profiles and the Image Type field in the output, we can summaries that the correct profile to use is WinXPSP2x86.

3.2 Step 2: Collect all the processes list for the machine

Collecting the processes those were running on the machine is the first step in the memory forensics to start the analysis as it will give us the information about all the running tasks in the memory.

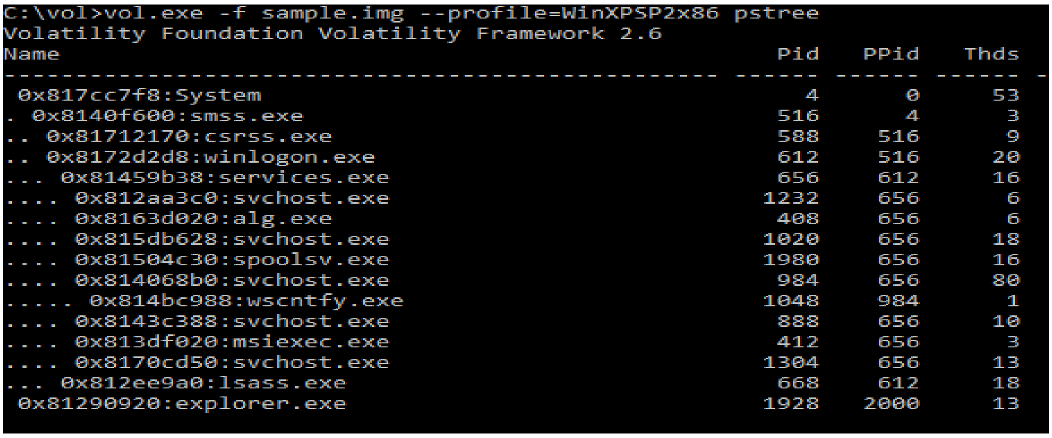


```
C:\vol>vol.exe -f sample.img --profile=WinXPSP2x86 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x817cc7f8 System                    4      0     53      247 ------      0
0x8140f600 smss.exe                516      4      3       21 ------      0 2009-01-08 01:46:50 UTC+0000
0x81712170 csrss.exe               588    516      9      318      0      0 2009-01-08 01:46:56 UTC+0000
0x8172d2d8 winlogon.exe            612    516     20      599      0      0 2009-01-08 01:46:56 UTC+0000
0x81459b38 services.exe            656    612     16      254      0      0 2009-01-08 01:46:59 UTC+0000
0x812ee9a0 lsass.exe               668    612     18      325      0      0 2009-01-08 01:46:59 UTC+0000
0x8143c388 svchost.exe             888    656     10      287      0      0 2009-01-08 01:47:02 UTC+0000
0x814068b0 svchost.exe             984    656     80     1556      0      0 2009-01-08 01:47:02 UTC+0000
0x815db628 svchost.exe            1020    656     18      197      0      0 2009-01-08 01:47:02 UTC+0000
0x812aa3c0 svchost.exe            1232    656      6       83      0      0 2009-01-08 01:47:55 UTC+0000
0x8170cd50 svchost.exe            1304    656     13      202      0      0 2009-01-08 01:47:56 UTC+0000
0x8163d020 alg.exe                 408    656      6      101      0      0 2009-01-08 01:48:23 UTC+0000
0x81290920 explorer.exe           1928   2000     13      332      0      0 2009-01-08 01:49:12 UTC+0000
0x814bc988 wscntfy.exe            1048    984      1       27      0      0 2009-01-08 01:49:13 UTC+0000
0x813df020 msiexec.exe             412    656      3       97      0      0 2009-01-08 01:49:22 UTC+0000
0x81504c30 spoolsv.exe            1980    656     16      596      0      0 2009-01-08 01:54:06 UTC+0000
```

Now, looking at the output of the process list it is very tedious to recognize whether there is something fishy in any process. The important parameters to look for are process ID, Parent Process ID, and the Timestamp information. To see the processes in the parent-child format lets now arrange the processes in the more structured way using command **pstree.**

3.3 Step 3: Dig dip into the analysis to find out the indicators of compromise

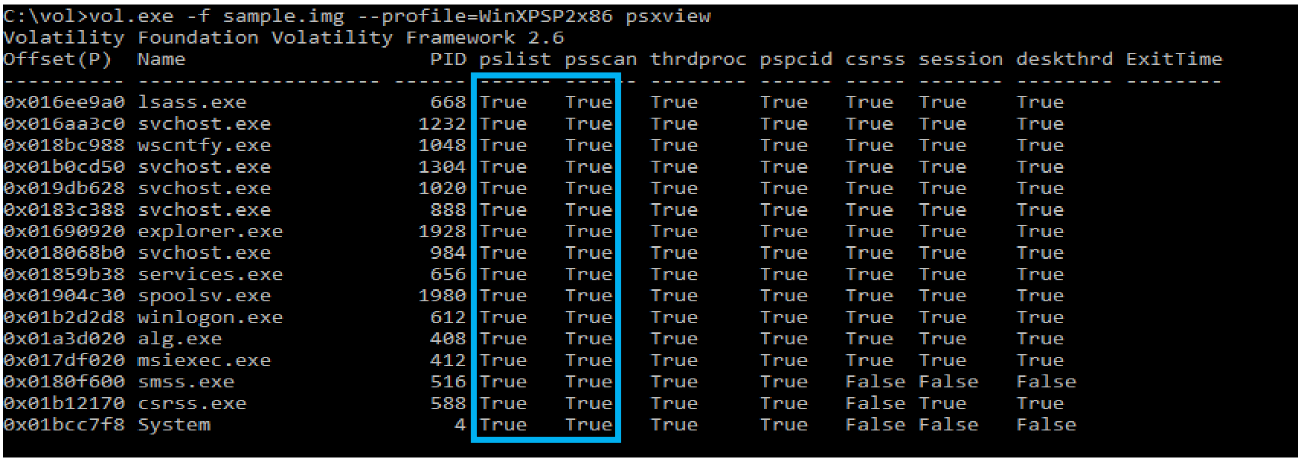
We will start to search the Indicators by identifying the parent and the child process. Even with the help of a process list, it is very difficult to know the parent process for the specific child process.



```
C:\vol>vol.exe -f sample.img --profile=WinXPSP2x86 pstree
Volatility Foundation Volatility Framework 2.6
Name                                                  Pid   PPid   Thds
-------------------------------------------------- ------ ------ ------ -
 0x817cc7f8:System                                      4      0     53
. 0x8140f600:smss.exe                                 516      4      3
.. 0x81712170:csrss.exe                               588    516      9
.. 0x8172d2d8:winlogon.exe                            612    516     20
... 0x81459b38:services.exe                           656    612     16
.... 0x812aa3c0:svchost.exe                          1232    656      6
.... 0x8163d020:alg.exe                               408    656      6
.... 0x815db628:svchost.exe                          1020    656     18
.... 0x81504c30:spoolsv.exe                          1980    656     16
.... 0x814068b0:svchost.exe                           984    656     80
..... 0x814bc988:wscntfy.exe                         1048    984      1
.... 0x8143c388:svchost.exe                           888    656     10
.... 0x813df020:msiexec.exe                           412    656      3
.... 0x8170cd50:svchost.exe                          1304    656     13
... 0x812ee9a0:lsass.exe                              668    612     18
 0x81290920:explorer.exe                             1928   2000     13
```

Processes are arranged in the logical manner i.e. Parent process is on the Top of the list. And corresponding child processes are listed below. With this output we can check from the process list parameters are appropriate.

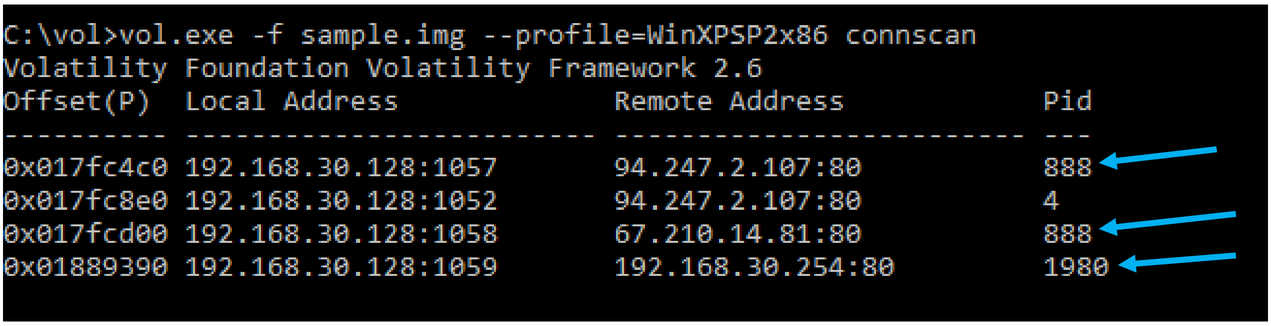
Now let's check whether there are any terminated processes are present in the memory with the help of **psxview.**



```
C:\vol>vol.exe -f sample.img --profile=WinXPSP2x86 psxview
Volatility Foundation Volatility Framework 2.6
Offset(P)  Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x016ee9a0 lsass.exe               668 True   True   True     True   True  True    True
0x016aa3c0 svchost.exe            1232 True   True   True     True   True  True    True
0x018bc988 wscntfy.exe            1048 True   True   True     True   True  True    True
0x01b0cd50 svchost.exe            1304 True   True   True     True   True  True    True
0x019db628 svchost.exe            1020 True   True   True     True   True  True    True
0x0183c388 svchost.exe             888 True   True   True     True   True  True    True
0x01690920 explorer.exe           1928 True   True   True     True   True  True    True
0x018068b0 svchost.exe             984 True   True   True     True   True  True    True
0x01859b38 services.exe            656 True   True   True     True   True  True    True
0x01904c30 spoolsv.exe            1980 True   True   True     True   True  True    True
0x01b2d2d8 winlogon.exe            612 True   True   True     True   True  True    True
0x01a3d020 alg.exe                 408 True   True   True     True   True  True    True
0x017df020 msiexec.exe             412 True   True   True     True   True  True    True
0x0180f600 smss.exe                516 True   True   True     True   False False   False
0x01b12170 csrss.exe               588 True   True   True     True   False True    True
0x01bcc7f8 System                    4 True   True   True     True   False False   False
```

From the above output, we can say that none of the processes was terminated as all the processes are turning up in the **pslist** as well as **psscan** output. The terminated processes would have flagged as false in the process list. That means such processes were terminated before capturing the memory of the machine.

The Next step is to check whether any process is trying to connect the remote IP's. Command **connscan** will help us to find such a process.

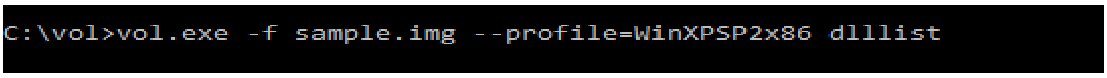


```
C:\vol>vol.exe -f sample.img --profile=WinXPSP2x86 connscan
Volatility Foundation Volatility Framework 2.6
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ---
0x017fc4c0 192.168.30.128:1057       94.247.2.107:80           888
0x017fc8e0 192.168.30.128:1052       94.247.2.107:80           4
0x017fcd00 192.168.30.128:1058       67.210.14.81:80           888
0x01889390 192.168.30.128:1059       192.168.30.254:80         1980
```

From the above results the important **points that should be noted are:**

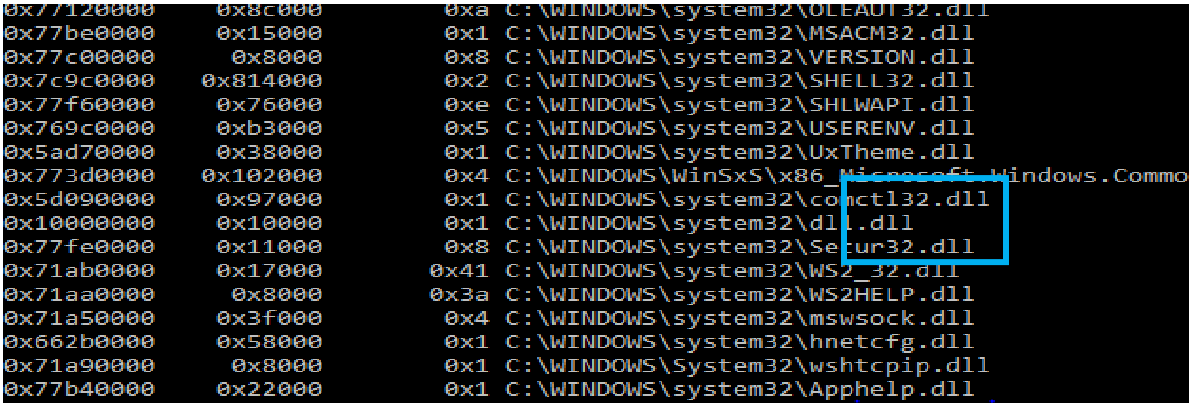
- Local IP 192.168.30.128 with Process ID 888 is trying to connect two remote addresses 94.247.2.107 & 67.210.14.81.
- Process with PID 888 is svchost.exe with Parent ID 656 i.e. services.exe

The next step is to check memory for suspicious dll's to check the probable hooking available in the victim machine inserted by the malicious process. Command **dlllist** displays all the available dll in the memory. Manual inspection is needed to check whether any suspicious dll is present or not.



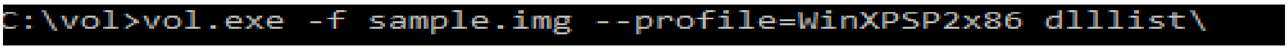
```
C:\vol>vol.exe -f sample.img --profile=WinXPSP2x86 dlllist
```

The output of the dlllist is quite big. In this example, we get suspicious dll entry as dll.dll in the path **C:\WINDOWS\system32\**

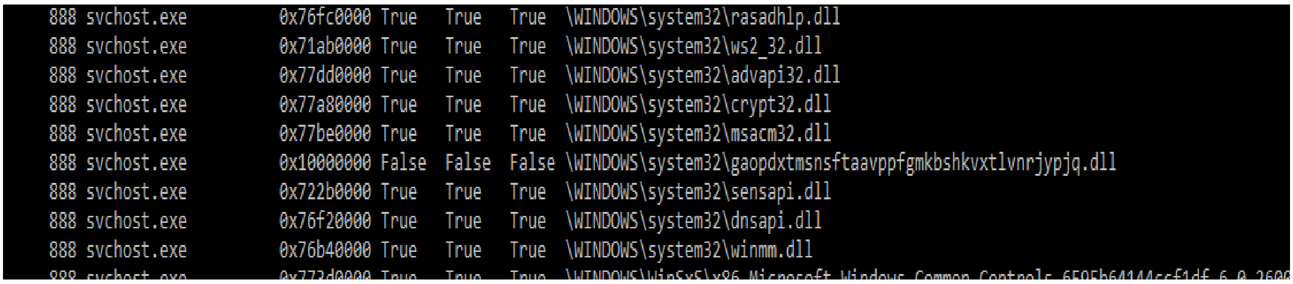


```
0x77120000     0x8c000        0xa C:\WINDOWS\system32\OLEAUT32.dll
0x77be0000     0x15000        0x1 C:\WINDOWS\system32\MSACM32.dll
0x77c00000      0x8000        0x8 C:\WINDOWS\system32\VERSION.dll
0x7c9c0000    0x814000        0x2 C:\WINDOWS\system32\SHELL32.dll
0x77f60000     0x76000        0xe C:\WINDOWS\system32\SHLWAPI.dll
0x769c0000     0xb3000        0x5 C:\WINDOWS\system32\USERENV.dll
0x5ad70000     0x38000        0x1 C:\WINDOWS\system32\UxTheme.dll
0x773d0000    0x102000        0x4 C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Commo
0x5d090000     0x97000        0x1 C:\WINDOWS\system32\comctl32.dll
0x10000000     0x10000        0x1 C:\WINDOWS\system32\dll.dll
0x77fe0000     0x11000        0x8 C:\WINDOWS\system32\Secur32.dll
0x71ab0000     0x17000       0x41 C:\WINDOWS\system32\WS2_32.dll
0x71aa0000      0x8000       0x3a C:\WINDOWS\system32\WS2HELP.dll
0x71a50000     0x3f000        0x4 C:\WINDOWS\system32\mswsock.dll
0x662b0000     0x58000        0x1 C:\WINDOWS\system32\hnetcfg.dll
0x71a90000      0x8000        0x1 C:\WINDOWS\System32\wshtcpip.dll
0x77b40000     0x22000        0x1 C:\WINDOWS\system32\Apphelp.dll
```

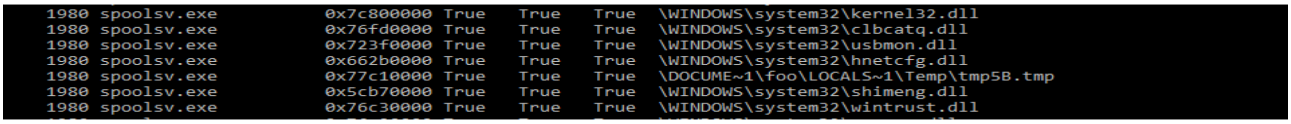
Since we have got suspicious dll entry which needs to be checked in detail later investigation with some more dll related commands will help us to find the traces of the infected part of the memory and the corresponding processes. The command ldrmodules detects the unlinked dll in the memory. Although the output of this command is very large the quick inspection can lead us to get the right traces.



```
C:\vol>vol.exe -f sample.img --profile=WinXPSP2x86 dlllist\
```



```
     888 svchost.exe          0x76fc0000 True   True   True  \WINDOWS\system32\rasadhlp.dll
     888 svchost.exe          0x71ab0000 True   True   True  \WINDOWS\system32\ws2_32.dll
     888 svchost.exe          0x77dd0000 True   True   True  \WINDOWS\system32\advapi32.dll
     888 svchost.exe          0x77a80000 True   True   True  \WINDOWS\system32\crypt32.dll
     888 svchost.exe          0x77be0000 True   True   True  \WINDOWS\system32\msacm32.dll
     888 svchost.exe          0x10000000 False  False  False \WINDOWS\system32\gaopdxtmsnsftaavppfgmkbshkvxtlvnrjypjq.dll
     888 svchost.exe          0x722b0000 True   True   True  \WINDOWS\system32\sensapi.dll
     888 svchost.exe          0x76f20000 True   True   True  \WINDOWS\system32\dnsapi.dll
     888 svchost.exe          0x76b40000 True   True   True  \WINDOWS\system32\winmm.dll
     888 svchost.exe          0x773d0000 True   True   True  \WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600
```



```
    1980 spoolsv.exe          0x7c800000 True   True   True   \WINDOWS\system32\kernel32.dll
    1980 spoolsv.exe          0x76fd0000 True   True   True   \WINDOWS\system32\clbcatq.dll
    1980 spoolsv.exe          0x723f0000 True   True   True   \WINDOWS\system32\usbmon.dll
    1980 spoolsv.exe          0x662b0000 True   True   True   \WINDOWS\system32\hnetcfg.dll
    1980 spoolsv.exe          0x77c10000 True   True   True   \DOCUME~1\foo\LOCALS~1\Temp\tmp5B.tmp
    1980 spoolsv.exe          0x5cb70000 True   True   True   \WINDOWS\system32\shimeng.dll
    1980 spoolsv.exe          0x76c30000 True   True   True   \WINDOWS\system32\wintrust.dll
```

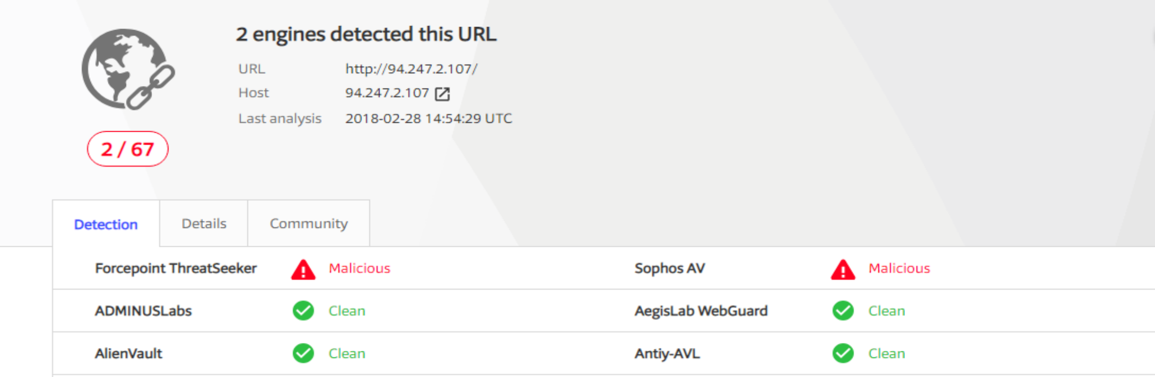
The command ldrmodules gives the two suspicious readings in this case. One **suspicious dll name in a system32 path and one temporary file that might have created** when malware was running in the memory of the machine.

4. ANALYSIS AND RESULTS

From the rough Analysis, we have collected the enough IOC's to point the malware infection. Further memory analysis will also lead us to trace the complete path of the memory and exact location in the memory. The following are the Indicators of malware Infections in the victim machine:

- Local IP 192.168.30.128 with Process ID 888 is trying to connect two remote addresses 94.247.2.107 & 67.210.14.81.
- Process with PID 888 is svchost.exe with Parent ID 656 i.e. services.exe
- Local IP 192.168.30.128 is connecting to itself through process ID 1980.
- Presence of suspicious dll name "gaopdxtmsnsftaavppfgmkbshkvxtlvnrjypjq.dll" in ldrmodules command confirms the detached dll which might have run on the system. The dll is again associated with PID 888(svchost.exe).

Now, to confirm the results from the experimental demonstration let's check the details of the remote IP addresses as well as malicious dll's. It is a possibility that remote IP's can be any legitimate application server IP.



2 engines detected this URL

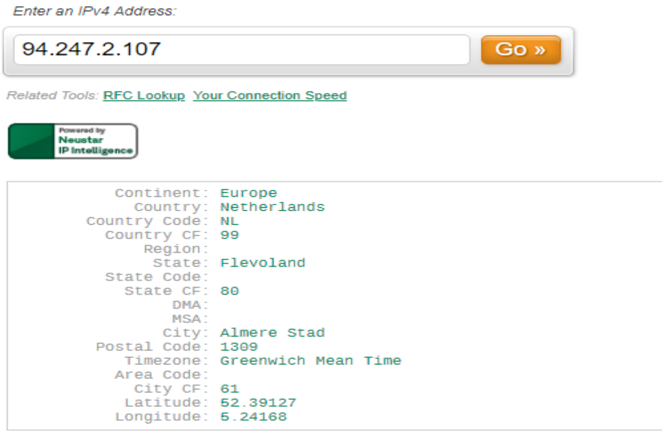
URL	http://94.247.2.107/
Host	94.247.2.107
Last analysis	2018-02-28 14:54:29 UTC

2 / 67

Detection | Details | Community

Forcepoint ThreatSeeker	Malicious	Sophos AV	Malicious
ADMINUSLabs	✓ Clean	AegisLab WebGuard	✓ Clean
AlienVault	✓ Clean	Antiy-AVL	✓ Clean

Two engines from the virus total detected this is the malicious IP. If we see the registration details of the IP from the **whois** database. When remote connection IP address checked on the **whois** database we got the following results:



Enter an IPv4 Address:

94.247.2.107 Go »

Related Tools: RFC Lookup Your Connection Speed

Powered by Neustar IP Intelligence

```
    Continent: Europe
      Country: Netherlands
 Country Code: NL
   Country CF: 99
       Region:
        State: Flevoland
   State Code:
     State CF: 80
          DMA:
          MSA:
         City: Almere Stad
  Postal Code: 1309
     Timezone: Greenwich Mean Time
    Area Code:
      City CF: 61
     Latitude: 52.39127
    Longitude: 5.24168
```

From the results, it is clear that IP is registered in the remote country and further analysis with threat intelligent solution leads us to the conclusion that IP address is related to malicious activities. We can get the similar conclusion after analyzing dumped suspicious dll and temp file on the online solution which supports the malware infection to the machine.

We can define the detection flow chart for such memory analysis for stealthy volatile attacks.

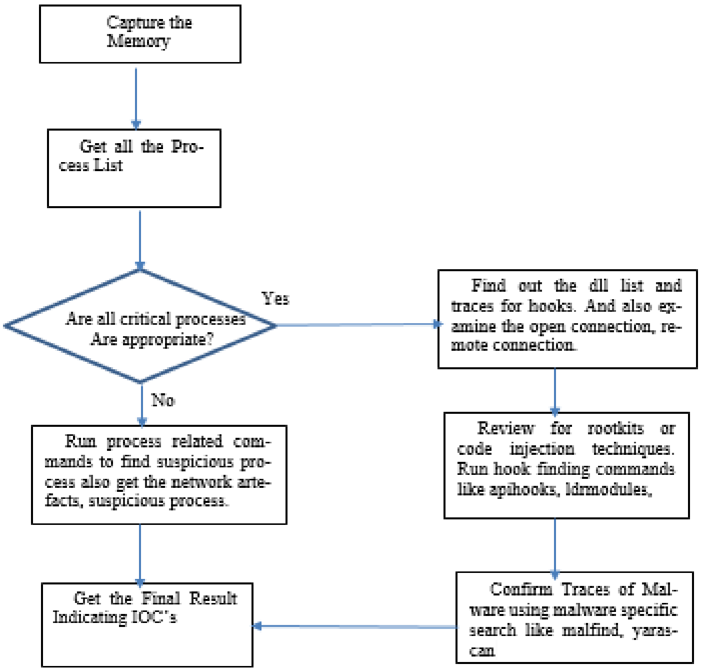

Capture the Memory
Get all the Process List
Are all critical processes Are appropriate?
Yes
No
Find out the dll list and traces for hooks. And also examine the open connection, remote connection.
Review for rootkits or code injection techniques. Run hook finding commands like apihooks, ldrmodules,
Confirm Traces of Malware using malware specific search like malfind, yarascan
Run process related commands to find suspicious process also get the network artefacts, suspicious process.
Get the Final Result Indicating IOC's

Fig. 1: Volatile threat detection flow

5. CONCLUSION

Memory Forensics can be extremely useful in conducting the stealthy volatile attacks which many a time reside only in memory or exclusively run from the machine memory. It can also modify the registry values which sometimes makes it be persistent even after the reboot of the machine. The example demonstrated shows us that examining the available process list is the very first thing investigation can be start off with. Apart from the process list, there are some other aspects who plays a very important role in analyzing such volatile malware threats. E.g. analyzing the open ports, examining remote connections, detailed examination of dll for hooking techniques etc. Indicators at these basic steps will guide investigators about the places those needs to be actually searched for the malware traces.

6. FUTURE SCOPE

Although we have presented a general brief flow and approach with which we can proceed with the analysis for finding out traces of advanced volatile threats residing in the memory or registry the use of the command line tool ‘volatility' requires the knowledge of command line tool as well as knowledge and methodology of static and dynamic malware analysis. Some work can be done in order to help the cyber investigators in detecting and analyzing malware from the RAM dump of the machine. Also, analysis using volatility needs all the commands to be remembered with the appropriate command format. Automating tasks up to some extent will help cyber investigators a lot in the detecting advanced volatile threats.

7. REFERENCES

[1] Thomas, Sunu, K. K. Sherly, and S. Dija. "Extraction of memory forensic artifacts from windows 7 ram image." In Information & Communication Technologies (ICT), 2013 IEEE Conference on, pp. 937-942. IEEE, 2013.

[2] Kumara, Ajay, and C. D. Jaidhar. "Execution time measurement of virtual machine volatile artifacts analyzers." In Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st International Conference on, pp. 314-319. IEEE, 2015.

[3] The Volatility Framework, https://code.google.com/p/Volatility/; 2014.

[4] Tomer Teller, Adi Hayon, "Enhancing Automated Malware Analysis Machines with Memory Analysis”, Blackhat Arsenal – 2014, p.no. 1-5.

[5] https://thebestvpn.com/cyber-security-statistics-2018/

[6] https://www.comparitech.com/blog/information-security/fileless-malware-attacks/