

Computer Forensics Tool Testing— The Volatility Framework: WK8
Course Name: CYB624
Professor: Cynthia Gonnella
Date: 03/13/2020
Examiner Name: Brian T. Carr

Table of Contents

List of Illustrative Materials	3
List of Tables	3
List of Figures	3
File Samples.....	4
Project Scope	5
Methodology	5
Analysis	6
Discussion.....	14
Conclusion and Future Research Recommendations	14
References	16
Appendix	17
Appendix A: Examiner Workstation Specifications	17

List of Illustrative Materials

List of Tables

Table 1: Case evidence items	4
Table 2: Hash Comparison Table	13

List of Figures

Figure 1: Hash values of the memory images	6
Figure 2: Imageinfo results for memory.img.....	7
Figure 3: Imageinfo results for memory2.img.....	7
Figure 4: Imageinfo results for memory4.img.....	8
Figure 5: Pslist output for memory.img	9
Figure 6: Creation and comparison of validation output files related to 349f6a9bc1efbdc9ca024be701823604.....	9
Figure 7: Pslist output for memory2.img	10
Figure 8: Creation and comparison of validation output files related to f5a328a3d4bdba93a4116c3c57589fa1	10
Figure 9: Pslist output for memory4.img	11
Figure 10: Creation and comparison of validation output files related to ca713e8ba0d5e1b1387c6389cedab5bctt	11
Figure 11: MD5 hash values of all output files for comparison	12

File Samples

Table 1 outlines the file samples used to test the pslist functionality of The Volatility Framework.

Table 1: Case evidence items

Description	Designation	Filename	MD5 Hash
Memory Image File	File Sample	memory.img	349f6a9bc1efbdc9ca024be701823604
Memory Image File	File Sample	WC_memory.img	349f6a9bc1efbdc9ca024be701823604
Memory Image File	File Sample	PRES_memory.img	349f6a9bc1efbdc9ca024be701823604
Memory Image File	File Sample	memory2.img	f5a328a3d4bdba93a4116c3c57589fa1
Memory Image File	File Sample	WC_memory2.img	f5a328a3d4bdba93a4116c3c57589fa1
Memory Image File	File Sample	PRES_memory2.img	f5a328a3d4bdba93a4116c3c57589fa1
Memory Image File	File Sample	memory4.img	ca713e8ba0d5e1b1387c6389cedab5bc
Memory Image File	File Sample	WC_memory4.img	ca713e8ba0d5e1b1387c6389cedab5bc
Memory Image File	File Sample	PRES_memory4.img	ca713e8ba0d5e1b1387c6389cedab5bc

Project Scope

The scope of the project will be limited to the pslist plugin functionality of The Volatility Framework (VF). The validation exercise will be completed using the Volatility Foundation Volatility Framework 2.6. This tool was utilized from the command line of an Ubuntu Linux Virtual Machine (VM). The Analyst tested the consistency of the output of the pslist plugin by outputting the results to a text file and then by hashing the text file. By comparing the hashes of different output text files, the Analyst was able to determine if VF produced consistent output when provided with input files with consistent hash values and identical command syntax. Additionally, the Analyst was able to determine that when provided with input files with different hash values and identical command syntax, VF would produce inconsistent output. Finally, The Analyst was able to determine if VF was able to successfully process all nine of the memory image files.

Methodology

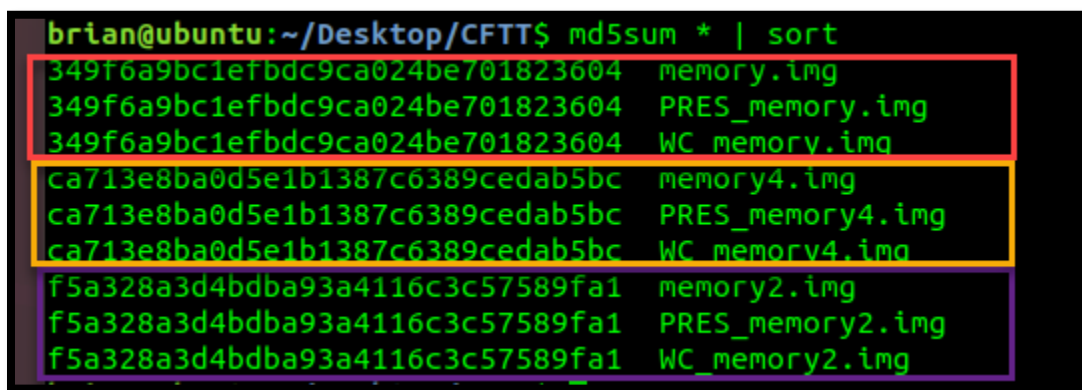
The Analyst chose to utilize the National Institute for Standards and Technology's (NIST) Computer Forensic Tool Testing (CFTT) methodology. This methodology has two parts. The first part is the selection development process, and the second portion is the tool test process. (National Institute for Standards and Technology, 2018) The development processes consisted of the Analyst selecting the VF as the tool which to use for the project and the pslist plugin as the functionality of that tool which was to be tested.

The tool test process began with the Analyst installing VF within an Ubuntu Linux virtual machine which was used for tool testing validation. The Analyst was able to install VF by utilizing the APT package manager. The Analyst utilized the installation command: `sudo apt install volatility`. The Analyst then reviewed the documentation before developing a plan to test

the pslist function of VF. The Analyst then carried out the CFTT plan and documented the results. Finally, the Analyst submitted the analysis results.

Analysis

The analysis began by first obtaining the hash values of each memory image that was to be processed by the pslist plugin of VF. The Analyst obtained each of the nine files' MD5 hash value by using the md5sum utility and then by sorting the output. There are three unique hash values among the nine files. Each hash value can be associated with three unique files. This information can be seen as outlined in *Figure 1*.



```
brian@ubuntu:~/Desktop/CFTT$ md5sum * | sort
349f6a9bc1efbdc9ca024be701823604 memory.img
349f6a9bc1efbdc9ca024be701823604 PRES_memory.img
349f6a9bc1efbdc9ca024be701823604 WC_memory.img
ca713e8ba0d5e1b1387c6389cedab5bc memory4.img
ca713e8ba0d5e1b1387c6389cedab5bc PRES_memory4.img
ca713e8ba0d5e1b1387c6389cedab5bc WC_memory4.img
f5a328a3d4bdba93a4116c3c57589fa1 memory2.img
f5a328a3d4bdba93a4116c3c57589fa1 PRES_memory2.img
f5a328a3d4bdba93a4116c3c57589fa1 WC_memory2.img
```

Figure 1: Hash values of the memory images

Next, the Analyst used the imageinfo plugin of VF to obtain the proper profile information for one file for each unique hash value. The Imageinfo results for the memory.img memory image revealed that there were a few suggested profiles. The suggested profiles for memory.img can be seen outlined in *Figure 2*. The Analyst then used the imageinfo plugin again, but this time on memory2.img. This time, VF only suggested two different profiles. The imageinfo results for memory2.img can be seen in *Figure 3*. The two suggested profiles for memory2.img can also be seen outlined in *Figure 3*. The Analyst then repeated this process of using the imageinfo VF plugin to obtain the suggested profiles for the memory4.img memory

capture. The suggested profiles for the memory4.img memory capture can be seen outlined in Figure 4.

```
brian@ubuntu:~/Desktop/CFTT$ volatility -f memory.img imageinfo
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64 23418, Win2008R2SP1x64, Win7SP1x64 23418
AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
AS Layer2 : FileAddressSpace (/home/brian/Desktop/CFTT/memory.img)
PAE type : No PAE
DTB : 0x187000L
KDBG : 0xf80002e020f0L
Number of Processors : 1
Image Type (Service Pack) : 1
KPCR for CPU 0 : 0xffffffff80002e03d00L
KUSER_SHARED_DATA : 0xffffffff78000000000L
Image date and time : 2015-10-12 11:41:31 UTC+0000
Image local date and time : 2015-10-12 07:41:31 -0400
brian@ubuntu:~/Desktop/CFTT$
```

Figure 2: Imageinfo results for memory.img

```
brian@ubuntu:~/Desktop/CFTT$ volatility -f memory2.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 (/home/brian/Desktop/CFTT/memory2.img)
PAE type : PAE
DTB : 0x334000L
KDBG : 0x80545ae0L
Number of Processors : 1
Image Type (Service Pack) : 3
KPCR for CPU 0 : 0xffdf000L
KUSER_SHARED_DATA : 0xffdf000L
Image date and time : 2015-10-12 17:06:17 UTC+0000
Image local date and time : 2015-10-12 13:06:17 -0400
brian@ubuntu:~/Desktop/CFTT$
```

Figure 3: Imageinfo results for memory2.img

```
brian@ubuntu:~/Desktop/CFTT$ volatility -f memory4.img imageinfo
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_
23418, Win2008R2SP1x64, Win7SP1x64_23418
AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
AS Layer2 : FileAddressSpace (/home/brian/Desktop/CFTT/memory4.img)
PAE type : No PAE
DTB : 0x187000L
KDBG : 0xf80002bf80a0L
Number of Processors : 1
Image Type (Service Pack) : 1
KPCR for CPU 0 : 0xffffffff80002bf9d00L
KUSER_SHARED_DATA : 0xffffffff78000000000L
Image date and time : 2015-10-12 18:40:43 UTC+0000
Image local date and time : 2015-10-12 14:40:43 -0400
```

Figure 4: Imageinfo results for memory4.img

After the suggested profiles for each memory image were obtained, the Analyst proceeded to use them to analyze their associated memory image files. The output of the pslist plugin for memory.img can be seen in *Figure 5*. The syntax used to obtain this output can be seen outlined in *Figure 5*. The Analyst then proceeded to repeat this process, except this time the output was sent to a txt file. The Analyst repeated this for all the input files with an MD5 hash value of 349f6a9bc1efbdc9ca024be701823604. The Analyst then compared the hash values of each output file to determine if the pslist plugin of the VF produced consistent output results when provided with input files with consistent hash values. The Analyst can be seen creating and comparing the output files in *Figure 6*. Additionally, the results highlighted in *Figure 6* show that the output files all produced the same MD5 hash value. This confirms that when provided with multiple input files with the same hash value, the pslist plugin of VF will produce consistent output results.


```

brian@ubuntu:~/Desktop/CFITTS$ volatility -f memory.img --profile=Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)      Name          PID  PPID  Thds  Hnds  Sess  How64  Start          Exit
-----
0xffffffff8003cdd040 System        4      0    93   632  -----  0  2015-10-12 03:56:54 UTC+0000
0xffffffff80051d9d0 smss.exe     300    4     2    29  -----  0  2015-10-12 03:56:54 UTC+0000
0xffffffff8005cd98f0 avgrsa.exe  328   300  0  -----  0  2015-10-12 03:56:54 UTC+0000
0xffffffff8005cd5900 avgrsa.exe  340   328  52   530  -----  0  2015-10-12 03:56:54 UTC+0000
0xffffffff8006dcf060 avgcsrva.exe 388   340  19   276  -----  0  2015-10-12 03:56:56 UTC+0000
0xffffffff8004891b10 smss.exe     624   300  0  -----  0  2015-10-12 03:56:57 UTC+0000
0xffffffff80055a6930 csrss.exe   632   624  9   624  0  0  2015-10-12 03:56:57 UTC+0000
0xffffffff8006014710 smss.exe     676   300  0  -----  1  2015-10-12 03:56:57 UTC+0000
0xffffffff80052d0060 winlnit.exe 684   624  3    74  0  0  2015-10-12 03:56:57 UTC+0000
0xffffffff8003db4140 csrss.exe   696   676  11   634  1  0  2015-10-12 03:56:57 UTC+0000
0xffffffff8005d3c8f0 winlogon.exe 744   676  3   114  1  0  2015-10-12 03:56:57 UTC+0000
0xffffffff8005fc0940 services.exe 788   684  7   223  0  0  2015-10-12 03:56:57 UTC+0000
0xffffffff80052b0700 lsass.exe   796   684  6   799  0  0  2015-10-12 03:56:57 UTC+0000
0xffffffff80053a6b10 lsm.exe     804   684  10   169  0  0  2015-10-12 03:56:57 UTC+0000
0xffffffff8006355b10 svchost.exe 912   788  10   372  0  0  2015-10-12 03:56:58 UTC+0000
0xffffffff8006393b10 vmacthlp.exe 976   788  3    54  0  0  2015-10-12 03:56:58 UTC+0000
0xffffffff800639eb10 WtuSystemSupp 996   788  7   105  0  1  2015-10-12 03:56:58 UTC+0000
0xffffffff80063c7b10 svchost.exe 324   788  7   311  0  0  2015-10-12 03:56:58 UTC+0000
0xffffffff8006402b10 svchost.exe 652   788  21   517  0  0  2015-10-12 03:56:58 UTC+0000
0xffffffff80064ab060 svchost.exe 1056  788  17   441  0  0  2015-10-12 03:56:59 UTC+0000
0xffffffff80064e5870 svchost.exe 1108  788  12   316  0  0  2015-10-12 03:56:59 UTC+0000
0xffffffff80064f1060 svchost.exe 1152  788  41  1311  0  0  2015-10-12 03:56:59 UTC+0000
0xffffffff8006588b10 svchost.exe 1344  788  17   415  0  0  2015-10-12 03:56:59 UTC+0000
0xffffffff800633c260 spoolsv.exe 1476  788  14   361  0  0  2015-10-12 03:56:59 UTC+0000
0xffffffff8005ed3330 svchost.exe 1520  788  18   300  0  0  2015-10-12 03:57:00 UTC+0000
0xffffffff800644f060 armsvc.exe 1604  788  4    69  0  1  2015-10-12 03:57:00 UTC+0000
0xffffffff800660c060 avgldsaagent.ex 1628  788  30   453  0  1  2015-10-12 03:57:00 UTC+0000
0xffffffff80065d33c0 avgsvca.exe 1664  788  19   540  0  0  2015-10-12 03:57:00 UTC+0000
0xffffffff8006625a20 avgwdsvcx.exe 1712  788  44   969  0  1  2015-10-12 03:57:00 UTC+0000
0xffffffff80066af4b0 svchost.exe 1776  788  11   266  0  0  2015-10-12 03:57:00 UTC+0000
0xffffffff800682e650 taskhost.exe 1072  788  13   264  1  0  2015-10-12 03:57:02 UTC+0000
0xffffffff80067ccb10 userinit.exe 1452  744  0  -----  1  0  2015-10-12 03:57:02 UTC+0000
0xffffffff80067e9850 dwm.exe     1292  1056  5   159  1  0  2015-10-12 03:57:02 UTC+0000
0xffffffff80067f3060 explorer.exe 2056  1452  32   996  1  0  2015-10-12 03:57:02 UTC+0000
0xffffffff8006956b10 vmtoolsd.exe 2416  2056  6   276  1  0  2015-10-12 03:57:04 UTC+0000
0xffffffff8006960360 runonce.exe 2424  2056  0  -----  1  0  2015-10-12 03:57:04 UTC+0000
0xffffffff8006972b10 avgui.exe  2472  2424  23   562  1  1  2015-10-12 03:57:04 UTC+0000
0xffffffff800697b10 avgui.exe  2480  2424  23   562  1  1  2015-10-12 03:57:04 UTC+0000

```

Figure 5: Pslist output for memory.img

```

brian@ubuntu:~/Desktop/CFITTS$ volatility -f memory.img --profile=Win7SP1x64 pslist > memory_output1.txt
Volatility Foundation Volatility Framework 2.6
brian@ubuntu:~/Desktop/CFITTS$ volatility -f WC_memory.img --profile=Win7SP1x64 pslist > memory_output2.txt
Volatility Foundation Volatility Framework 2.6
brian@ubuntu:~/Desktop/CFITTS$ volatility -f PRES_memory.img --profile=Win7SP1x64 pslist > memory_output3.txt
Volatility Foundation Volatility Framework 2.6
brian@ubuntu:~/Desktop/CFITTS$ md5sum memory_*
39c1228cfc359fe19d33fabadb12247b memory_output1.txt
39c1228cfc359fe19d33fabadb12247b memory_output2.txt
39c1228cfc359fe19d33fabadb12247b memory_output3.txt
brian@ubuntu:~/Desktop/CFITTS$

```

Figure 6: Creation and comparison of validation output files related to 349f6a9bc1efbdc9ca024be701823604

The Analyst then repeated this process for the input files associated with the MD5 hash value, f5a328a3d4bdba93a4116c3c57589fa1. The output of the pslist plugin can be seen in Figure 7. The Analyst can then be seen outputting the results of the pslist plugin to text files in Figure 8. The hash values of the output files can be seen outlined in Figure 8. These results show that the output files all had the same hash value, showing that the pslist plugin was able to produce consistent results when provided with consistent input files.

```

brian@ubuntu:~/Desktop/CFTT$ volatility -f memory2.img --profile=WinXPSP2x86 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name                PID  PPID  Thds  Hnds  Sess  Wow64  Start                Exit
-----
0x865c6830 System                4    0     58   419   -----  0
0x864d9ca8 smss.exe             384  4      3    19   -----  0 2015-10-12 16:53:28 UTC+0000
0x865292e8 csrss.exe           540  384   10   487   0        0 2015-10-12 16:53:28 UTC+0000
0x8648c338 winlogon.exe         620  384   20   589   0        0 2015-10-12 16:53:28 UTC+0000
0x864fb978 services.exe        708  620   16   343   0        0 2015-10-12 16:53:28 UTC+0000
0x863435e0 lsass.exe            720  620   24   367   0        0 2015-10-12 16:53:28 UTC+0000
0x864ea748 vmacthlp.exe        876  708    1    25    0        0 2015-10-12 16:53:28 UTC+0000
0x864e3a88 svchost.exe      888  708   22   219   0        0 2015-10-12 16:53:28 UTC+0000
0x862845c0 svchost.exe          972  708   10   268   0        0 2015-10-12 16:53:29 UTC+0000
0x863c3468 svchost.exe     1056 708   84  1569   0        0 2015-10-12 16:53:29 UTC+0000
0x85ec3918 svchost.exe     1104 708   12    97   0        0 2015-10-12 16:53:29 UTC+0000
0x86275860 svchost.exe     1140 708   16   205   0        0 2015-10-12 16:53:29 UTC+0000
0x8627c020 explorer.exe     1472 1424   17   594   0        0 2015-10-12 16:53:30 UTC+0000
0x8626abf0 spoolsv.exe          1556 708   15   141   0        0 2015-10-12 16:53:30 UTC+0000
0x8626d768 vmtoolsd.exe         1756 1472    3   182   0        0 2015-10-12 16:53:31 UTC+0000
0x86234020 VGAuthService.e     520  708    2    60   0        0 2015-10-12 16:53:36 UTC+0000
0x8646b5d8 vmtoolsd.exe         784  708    8   269   0        0 2015-10-12 16:53:39 UTC+0000
0x86205020 wmioprse.exe         756  888   12   231   0        0 2015-10-12 16:53:40 UTC+0000
0x85e41630 TPAutoConnSvc.e     1400 708    5   106   0        0 2015-10-12 16:53:40 UTC+0000
0x85e33da0 alg.exe           1916 708    6   107   0        0 2015-10-12 16:53:40 UTC+0000
0x85e33630 TPAutoConnect.e     2044 1400    1    70   0        0 2015-10-12 16:53:41 UTC+0000
0x861f79e0 wuauclt.exe          1820 1056    4   137   0        0 2015-10-12 16:54:39 UTC+0000
0x861ed7d8 wscntfy.exe          1900 1056    1    28   0        0 2015-10-12 16:54:40 UTC+0000
0x85e447a8 wordpad.exe         1792 1472    3    81   0        0 2015-10-12 16:58:15 UTC+0000
0x85e87ae0 cmd.exe             264  1472    1    30   0        0 2015-10-12 16:58:22 UTC+0000
0x85ec8588 firefox.exe         1888 1472   45   562   0        0 2015-10-12 16:58:45 UTC+0000
0x85e70200 cmd.exe             3260 1472    1    32   0        0 2015-10-12 17:02:32 UTC+0000
0x8638ada0 FTK Imager.exe     3604 1472   10   280   0        0 2015-10-12 17:05:53 UTC+0000
brian@ubuntu:~/Desktop/CFTT$ █

```

Figure 7: Pslist output for memory2.img

```

brian@ubuntu:~/Desktop/CFTT$ volatility -f memory2.img --profile=WinXPSP2x86 pslist > memory2_output1.txt
Volatility Foundation Volatility Framework 2.6
brian@ubuntu:~/Desktop/CFTT$ volatility -f WC_memory2.img --profile=WinXPSP2x86 pslist > memory2_output2.txt
Volatility Foundation Volatility Framework 2.6
brian@ubuntu:~/Desktop/CFTT$ volatility -f PRES_memory2.img --profile=WinXPSP2x86 pslist > memory2_output3.txt
Volatility Foundation Volatility Framework 2.6
brian@ubuntu:~/Desktop/CFTT$ md5sum memory2_output*
96db1918f0104262b34ca36494924ead memory2_output1.txt
96db1918f0104262b34ca36494924ead memory2_output2.txt
96db1918f0104262b34ca36494924ead memory2_output3.txt
brian@ubuntu:~/Desktop/CFTT$ *

```

Figure 8: Creation and comparison of validation output files related to f5a328a3d4bdba93a4116c3c57589fa1

The Analyst repeated this process for the third time to validate the pslist output for the input files related to ca713e8ba0d5e1b1387c6389cedab5bc MD5 hash value. The VF pslist plugin output for memory4.img can be seen in *Figure 9*. The Analyst then proceeded to output the results of the pslist VF plugin to text files, this can be seen in *Figure 10*. The Analyst then obtained the MD5 hash value of each output file in order to determine if they were consistent. The hash values of each output file associated with ca713e8ba0d5e1b1387c6389cedab5bc can be

seen in *Figure 10*. The hash values in *Figure 10* were all consistent, which validated that the output was consistent for input files with consistent hash values.

```
brian@ubuntu:~/Desktop/CFTT$ volatility -f memory4.img --profile=Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)      Name                PID  PPID  Thds  Hnds  Sess  Wow64  Start                Exit
-----
0xffffffff80018b39e0 System              4    0     89   565   -----  0 2015-10-12 18:34:06 UTC+0000
0xffffffff8001e8bb30 smss.exe            252  4     2    29   -----  0 2015-10-12 18:34:06 UTC+0000
0xffffffff8002cad510 csrss.exe           336  320   9    565   0        0 2015-10-12 18:34:07 UTC+0000
0xffffffff8002eb30 wininit.exe         388  320   3    75   0        0 2015-10-12 18:34:07 UTC+0000
0xffffffff8002eb8b30 csrss.exe           396  380  10   345   1        0 2015-10-12 18:34:07 UTC+0000
0xffffffff8002ef5060 winlogon.exe        432  380   3    114   1        0 2015-10-12 18:34:07 UTC+0000
0xffffffff8002ef9b30 services.exe        488  388   7    252   0        0 2015-10-12 18:34:07 UTC+0000
0xffffffff8002f563a0 lsass.exe           504  388   7    659   0        0 2015-10-12 18:34:07 UTC+0000
0xffffffff8002f57410 lsm.exe             512  388  10    156   0        0 2015-10-12 18:34:07 UTC+0000
0xffffffff8003038060 svchost.exe         604  488  10    372   0        0 2015-10-12 18:34:07 UTC+0000
0xffffffff800305b060 vmacthlp.exe        664  488   3    54   0        0 2015-10-12 18:34:07 UTC+0000
0xffffffff800307eb30 svchost.exe         708  488   7    315   0        0 2015-10-12 18:34:07 UTC+0000
0xffffffff80030c07a0 svchost.exe         792  488  19    470   0        0 2015-10-12 18:34:07 UTC+0000
0xffffffff80030dd370 svchost.exe         832  488  23    543   0        0 2015-10-12 18:34:08 UTC+0000
0xffffffff80030eab30 svchost.exe         860  488  37    982   0        0 2015-10-12 18:34:08 UTC+0000
0xffffffff800311b730 audiodg.exe         944  792   5    133   0        0 2015-10-12 18:34:08 UTC+0000
0xffffffff800314db30 svchost.exe        1016 488  11    546   0        0 2015-10-12 18:34:08 UTC+0000
0xffffffff8003168b30 svchost.exe         560  488  19    408   0        0 2015-10-12 18:34:08 UTC+0000
0xffffffff80032559a0 dwm.exe            1148 832   5    127   1        0 2015-10-12 18:34:08 UTC+0000
0xffffffff8003278920 explorer.exe       1200 1140  30   797   1        0 2015-10-12 18:34:08 UTC+0000
0xffffffff80032e1b30 spoolsv.exe        1260 488  14    354   0        0 2015-10-12 18:34:09 UTC+0000
0xffffffff8003312b30 taskhost.exe       1308 488   7    150   1        0 2015-10-12 18:34:09 UTC+0000
0xffffffff80033405f0 svchost.exe         1332 488  19    317   0        0 2015-10-12 18:34:09 UTC+0000
0xffffffff80033e2060 vmtoolsd.exe       1472 1200   6    217   1        0 2015-10-12 18:34:09 UTC+0000
0xffffffff80033ebb30 RaUI.exe           1484 1200   2    139   1        1 2015-10-12 18:34:09 UTC+0000
0xffffffff8003422900 RaRegistry.exe     1540 488   4    86   0        1 2015-10-12 18:34:09 UTC+0000
0xffffffff80034a5b30 RaRegistry64.exe  1660 488   4    60   0        0 2015-10-12 18:34:09 UTC+0000
0xffffffff80034ee270 RtlService.exe     1712 488   4    67   0        1 2015-10-12 18:34:10 UTC+0000
0xffffffff8003524b30 RtlService.exe    1760 488   4    67   0        1 2015-10-12 18:34:10 UTC+0000
0xffffffff8003536550 RtlWlan.exe        1796 1712   6    199   1        1 2015-10-12 18:34:10 UTC+0000
0xffffffff800353c060 RtlService.exe    1804 488   4    67   0        1 2015-10-12 18:34:10 UTC+0000
0xffffffff8002f429d0 RtlWlan.exe        1836 1760   6    166   1        1 2015-10-12 18:34:10 UTC+0000
0xffffffff8003572b30 RtlWlan.exe        1864 1804   4    203   1        1 2015-10-12 18:34:10 UTC+0000
0xffffffff800357e680 runSM.exe           1876 488   3    66   0        1 2015-10-12 18:34:10 UTC+0000
0xffffffff8003580b30 svchost.exe        1908 488   7    98   0        0 2015-10-12 18:34:10 UTC+0000
0xffffffff80035bab30 SwUSB.exe          1916 1876   6    210   1        1 2015-10-12 18:34:10 UTC+0000
```

Figure 9: Pslist output for memory4.img

```
brian@ubuntu:~/Desktop/CFTT$ volatility -f memory4.img --profile=Win7SP1x64 pslist > memory4_output1.txt
Volatility Foundation Volatility Framework 2.6
brian@ubuntu:~/Desktop/CFTT$ volatility -f WC_memory4.img --profile=Win7SP1x64 pslist > memory4_output2.txt
Volatility Foundation Volatility Framework 2.6
brian@ubuntu:~/Desktop/CFTT$ volatility -f PRES_memory4.img --profile=Win7SP1x64 pslist > memory4_output3.txt
Volatility Foundation Volatility Framework 2.6
brian@ubuntu:~/Desktop/CFTT$ md5sum memory4_out*
4901984294b995397e1e1cef79f60e41 memory4_output1.txt
4901984294b995397e1e1cef79f60e41 memory4_output2.txt
4901984294b995397e1e1cef79f60e41 memory4_output3.txt
brian@ubuntu:~/Desktop/CFTT$
```

Figure 10: Creation and comparison of validation output files related to ca713e8ba0d5e1b1387c6389cedab5bctt

The Analyst compared the hash values of all the output files in order to determine if the output would vary if the hash value of the input file varied. The hash values of the output files in *Figure 11*, show that there are three different hash values among the output files. Additionally, each hash value is associated with three of the output files. The associated output files hash grouping corresponded with the associated input files hash grouping. Since the hash values varied across the output files, the determination was able to be made that when provided with

input files that varied in hash value, the output would vary. Additionally, the plist plugin of VF successfully processed each of the nine memory image capture files.

```
brian@ubuntu:~/Desktop/CFTTS md5sum * output*
96db1918f0104262b34ca36494924ead memory2_output1.txt
96db1918f0104262b34ca36494924ead memory2_output2.txt
96db1918f0104262b34ca36494924ead memory2_output3.txt
4901984294b995397e1e1cef79f60e41 memory4_output1.txt
4901984294b995397e1e1cef79f60e41 memory4_output2.txt
4901984294b995397e1e1cef79f60e41 memory4_output3.txt
39c1228cfc359fe19d33fabadb12247b memory_output1.txt
39c1228cfc359fe19d33fabadb12247b memory_output2.txt
39c1228cfc359fe19d33fabadb12247b memory_output3.txt
brian@ubuntu:~/Desktop/CFTTS
```

Figure 11: MD5 hash values of all output files for comparison

The items in *Table2* show the relationships between the input and output files of the computer forensics tool testing project.

Table 2: Hash Comparison Table

Input File Name	Input File Hash Value	Output File Name	Output File Hash Value	Profile Used
memory.img	349f6a9bc1efbdc9ca024be701823604	memory_output1.txt	39c1228cfc359fe19d33fabadb12247b	Win7SP1x64
WC_memory.img	349f6a9bc1efbdc9ca024be701823604	memory_output2.txt	39c1228cfc359fe19d33fabadb12247b	Win7SP1x64
PRES_memory.img	349f6a9bc1efbdc9ca024be701823604	memory_output3.txt	39c1228cfc359fe19d33fabadb12247b	Win7SP1x64
memory2.img	f5a328a3d4bdba93a4116c3c57589fa1	memory2_output1.txt	96db1918f0104262b34ca36494924ead	WinXPSP2x86
WC_memory2.img	f5a328a3d4bdba93a4116c3c57589fa1	memory2_output2.txt	96db1918f0104262b34ca36494924ead	WinXPSP2x86
PRES_memory2.img	f5a328a3d4bdba93a4116c3c57589fa1	memory2_output3.txt	96db1918f0104262b34ca36494924ead	WinXPSP2x86
memory4.img	ca713e8ba0d5e1b1387c6389cedab5bc	memory4_output1.txt	4901984294b995397e1e1cef79f60e41	Win7SP1x64
WC_memory4.img	ca713e8ba0d5e1b1387c6389cedab5bc	memory4_output2.txt	4901984294b995397e1e1cef79f60e41	Win7SP1x64
PRES_memory4.img	ca713e8ba0d5e1b1387c6389cedab5bc	memory4_output3.txt	4901984294b995397e1e1cef79f60e41	Win7SP1x64

Discussion

VF is an extremely useful tool for performing analysis of volatile memory data. VF is a cross-platform, modular, memory analysis tool. In recent years memory forensics has become increasingly important, and many analysts and examiners are finding that the VF is a necessary tool to use. (The Volatility Foundation, 2018)

Memory forensic has become important for several reasons. One of those is that many of the new variants of malware infect the host system in a fileless manor. These fileless variants of malware are becoming increasingly common. The security firm TrendMicro reported that they had witnessed a 396% increase in fileless malware threats which they encountered from January of 2018 to June of 2019. (TrendMicro, 2019)

Another reason is that volatile data may provide insight with non-volatile storage data cannot. "Critical data often exists exclusively in memory, such as disk encryption keys, memory-resident injected code fragments, off-the-record chat messages, unencrypted e-mail messages, and non-cacheable Internet history records" (Ligh, Case, Levy, & Walters, 2014) Volatile data is separate from the non-volatile storage data. Additionally, those two locations store different types of data. Analyzing any memory captures along with the non-volatile storage media of a system, will help to ensure that an examiner is not overlooking anything.

Conclusion and Future Research Recommendations

The plist plugin of VF was able to successfully process nine memory image capture files. Additionally, VF was able to demonstrate consistency and integrity when the images were processed the by plist plugin. It was determined through hash value comparison that when the plist plugin of VF was provided with two input files that had the same MD5 hash value, the output would be consistent. Additionally, the test showed that when provided with two input files

with different MD5 hash values, the output would additionally be different for each use of the pslist plugin.

The Analyst recommends testing additional functionalities of VF. The Analyst additionally recommends testing additional tools used for memory analysis. Memory forensics has become more important as some threats have moved to live exclusively in volatile data.

In order to use these tools to help defeat cybercriminals, validating data for integrity is an essential function of any digital forensic professional as any evidence or findings should be ready to be prepared in a court of law. (Nelson, Phillips, & Steuart, 2016) It is necessary to validate additional functionalities of VF in order to ensure that the integrity of those functions output will also stand up to scrutiny.

References

- Ligh, M. H., Case, A., Levy, J., & Walters, A. (2014). *The Art of Memory Forensics Detecting Malware and Threats in Windows, Linux, and Mac, Memory*. Indianapolis: Wiley.
- National Institute for Standards and Technology. (2018, Feb 22). *Methodology Overview*. Retrieved from nist.gov: <https://www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-program-cftt/cftt-general-0>
- Nelson, B., Phillips, A., & Steuart, C. (2016). *Guide To Computer Forensics And Investigations*. Boston: Cengage Learning.
- The Volatility Foundation. (2018). *About The Volatility Foundation*. Retrieved from volatilityfoundation.org: <https://www.volatilityfoundation.org/about>
- TrendMicro. (2019, July 29). *Risks Under The Radar Understanding Fileless Threats*. Retrieved from trendmicro.com: <https://www.trendmicro.com/vinfo/ph/security/news/security-technology/risks-under-the-radar-understanding-fileless-threats>

Appendix

Appendix A: Examiner Workstation Specifications

- Computer Name: BrianCarrGS75
- Operating System (OS) Name: Windows 10 Home
- OS Version: 1809
- System Make/Model: MSI GS75 Stealth
- System Serial Number: K1910N0052372
- Time Zone of Examiner Machine: Eastern Standard Time (-5:00 GMT)
- System date/time is consistent with the time zone listed above, as verified by <http://nist.time.gov/>.