Basic Static Malware Analysis using open source tools

B. Jaya Prasad, Haritha Annangi & Krishna Sastry Pendyala

Malware Analysis Unit, Digital Forensics CoE, Enterprise Security & Risk Management, TCS

***

The recent Malware attacks on banks, financial institutions, and payment processors are a validation of the increasing technical expertise of cybercriminals and their ability to cause significant damage while orchestrating remotely. From mobile malware to banking Trojans, and point-of-sale (POS) and retail breaches, the threat landscape continues to evolve. According to anti-malware product vendors the average time to resolve a malware attack ranges from 18-26 days, resulting huge business down-time. In addition, the average cost of cleanup, cost of investigation, increased manifold. The two reasons for this pathetic situation are:

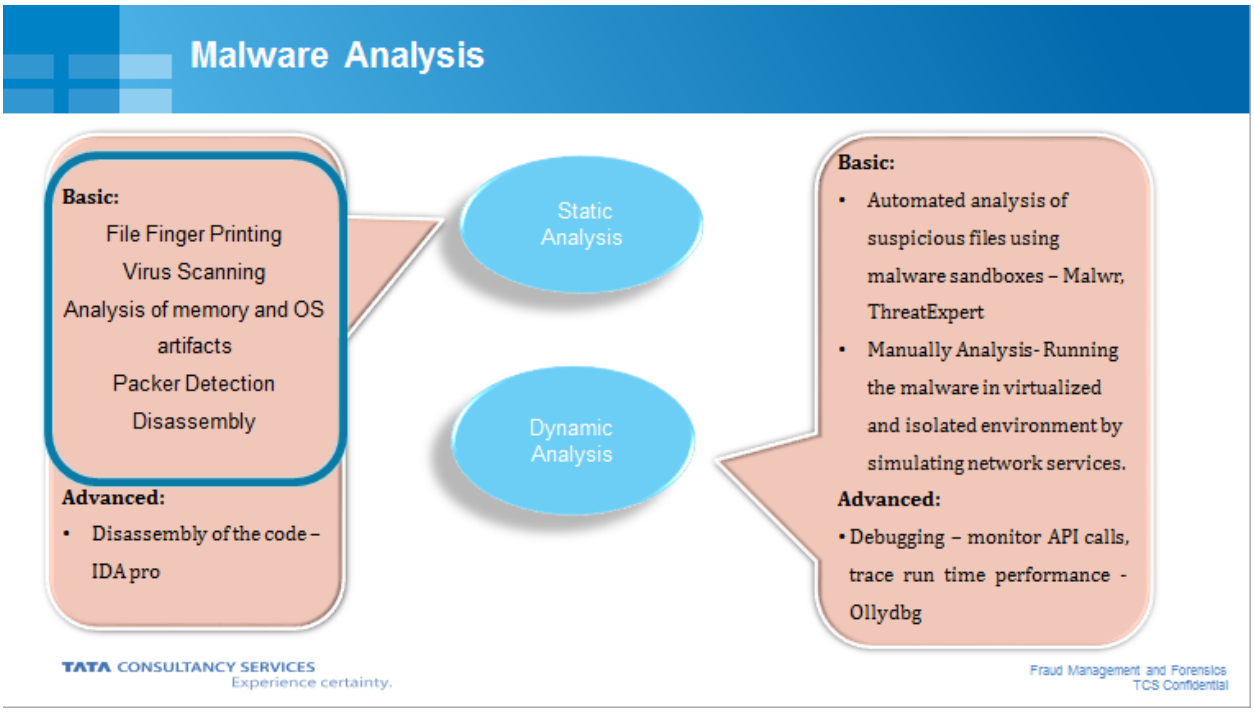
1. Lack of coordination between Security Operation Centre (SoC), Incident Response and Digital forensics teams, managed by different vendors and working in silo's ;
2. Lack of minimum knowledge to the members of SoC team in identification, collection and preliminary analysis of the malware or indicators of Compromise, to mitigate the impact.

Currently, the SOC team thinks "raising ticket" and informing the same to IR team as only their task/activity. The need of the hour is every member of Security operation center (SOC) team should have a minimum knowledge of malware incident handling and digital Forensics. This knowledge will help in stopping the spread of malware, reduces the duration & severity of the incident.

In this article, the authors described the collection of RAM dump and system files by using FTK Imager, and how to perform basic static malware analysis such as File Fingerprinting, Virus Scanning, analyzing memory artefacts (Pagefile.sys, hiberfile.sys), Packer Detection and Disassembly using open source and free tools.

Introduction:

Malware analysis is the art of dissecting malware to understand its behaviour such as, what changes it makes in the system files, how to identify it, and how to defeat/eliminate it etc., There are various industry accepted techniques available for malware analysis. Following diagram helps you to understand available malware analysis techniques.



Static analysis is done without executing the malware whereas dynamic analysis was carried-out by executing the malware file in a "**controlled environment**". When performing dynamic malware analysis malware analyst actually run the malware in an isolated operating environment to observe its behavior. Static analysis is generally safer than dynamic analysis. But it's largely ineffective against sophisticated malware, and it can miss important behaviors. This article provides guidelines specific to "**basic static malware analysis techniques" using free & open source tools** and also helpful for an incident responder to collect and preserve relevant information in a forensically sound manner during a malware investigation. An important and initial step of any malware analysis activity is collection of volatile memory and other important artifacts like **pagefile.sys, hiberfile.sys, windows event logs, registry files**.

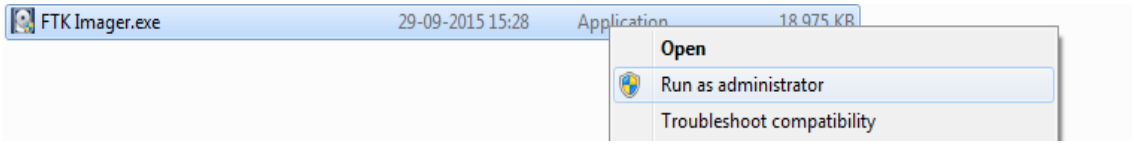
In this article we discussed on collecting volatile memory and other important artifacts using **"FTK Imager",** a free forensic utility to collect disk and process memory and some static malware analysis techniques. Following step-wise procedure can be used to collect and preserve relevant artifacts for a malware investigation.

Step 1: Pre-Requisites of RAM Dump Collection:

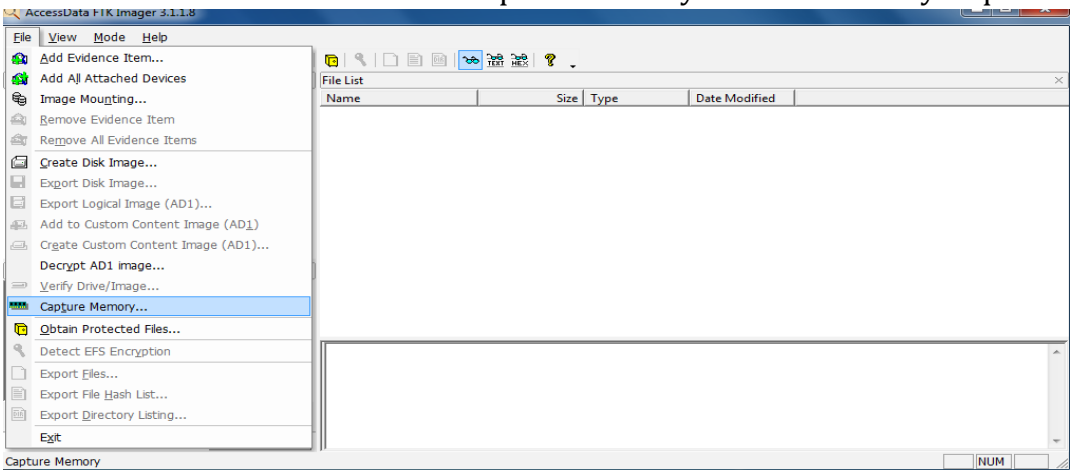
a. Incident responder should carry an external hard disk to the security incident location to collect the RAM dump. Make note of the Serial number of external device that is being connected to suspect's machine.
b. Make Sure the Hard disk is completely wiped. (NOTE: Make sure that the disk has more space than the size of all the collections)
c. If the size of RAM and the pagefile.sys is more than 4GB, Format the hard disk with NTFS file system as, the maximum file size that FAT32 supports is 4GB.
d. Create a directory in the external disk with the case name and create necessary sub directories in it.

Step 2: Collection of RAM Dump & other important artifacts

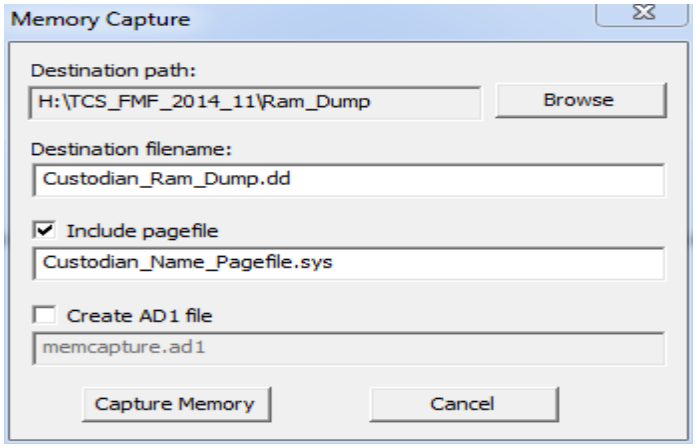
a. The RAM dump should be collected in .dd, .mem, or .vmem format. It should be collected as a single file.
b. Connect the external hard disk to the suspected machine and run FTK Imager with administrator privileges as shown below.



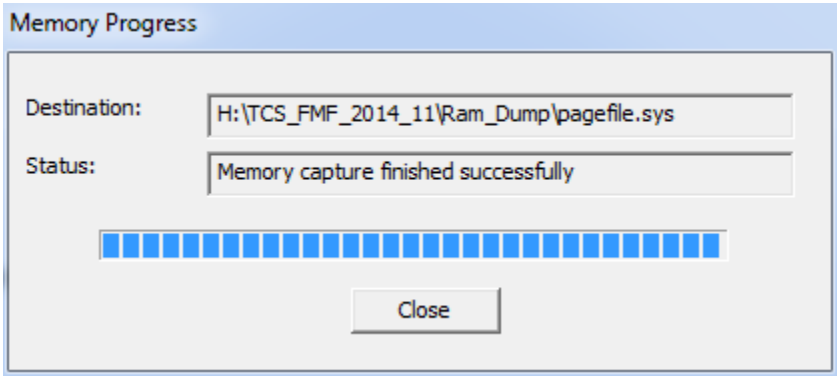
c. Select the File menu and click on "Capture memory" to start memory acquisition.



d. Relevant details of path of destination and image file can be provided in the popup.



e. In the **Destination Path**, browse to the location of external hard drive to save the RAM dump.
f. Fill the **Destination filename** field in the following format: **name of the custodian_Ram_Dump with extension dd.**
g. It is always preferred to include pagefile.sys. This can also be collected by manually selecting the "pagefile.sys" from the physical drive.
h. You don't need to select "Create AD1 file" as the output of image would be "Filename .dd ".
i. Progress bar is displayed showing the status of memory capture. Once the capture is completed, the status changes to "Memory capture finished successfully". Once it is done, press Close button.



j. Verify the successful completion of the imaging process by comparing the size of both RAM dump and pagefile.sys to system's RAM and pagefile.sys respectively.

k. Following are the few important artifacts in windows environment, useful for a malware analyst/ First responder.

   i. **Pagefile.sys:** Microsoft Windows uses a paging file, called pagefile.sys, to store frames of memory that do not current fit into physical memory. This file, stored in %SystemDrive%\pagefile.sys is a hidden system file. Because the

operating system keeps this file open during normal operation, it can never be read or accessed by a user. It is possible to read this file by parsing the raw file system.

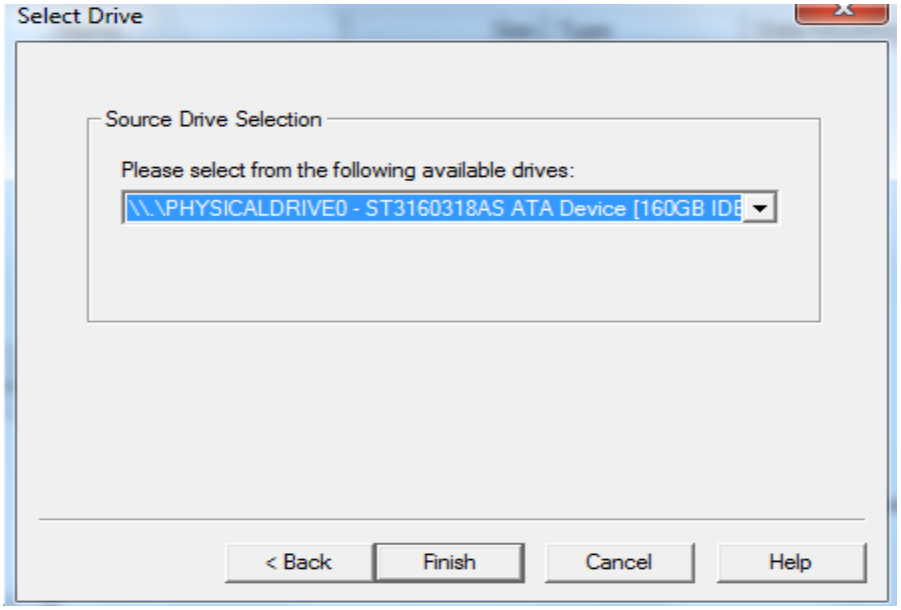
ii. **Hiberfile.sys:** hiberfil.sys is the file used by default by Microsoft Windows to save the machine's state as part of the hibernation process. The operating system also keeps an open file handle to this file, so no user, including the Administrator, can read the file while the system is running. This file, stored in %SystemDrive%\hiberfile.sys is a hidden system file.

iii. **Event Logs:** Windows systems are capable of recording a number of different events in the Event Log, depending upon the audit configuration. Windows event logs can be available at "C:\Windows\System32\winevt -Event Logs"

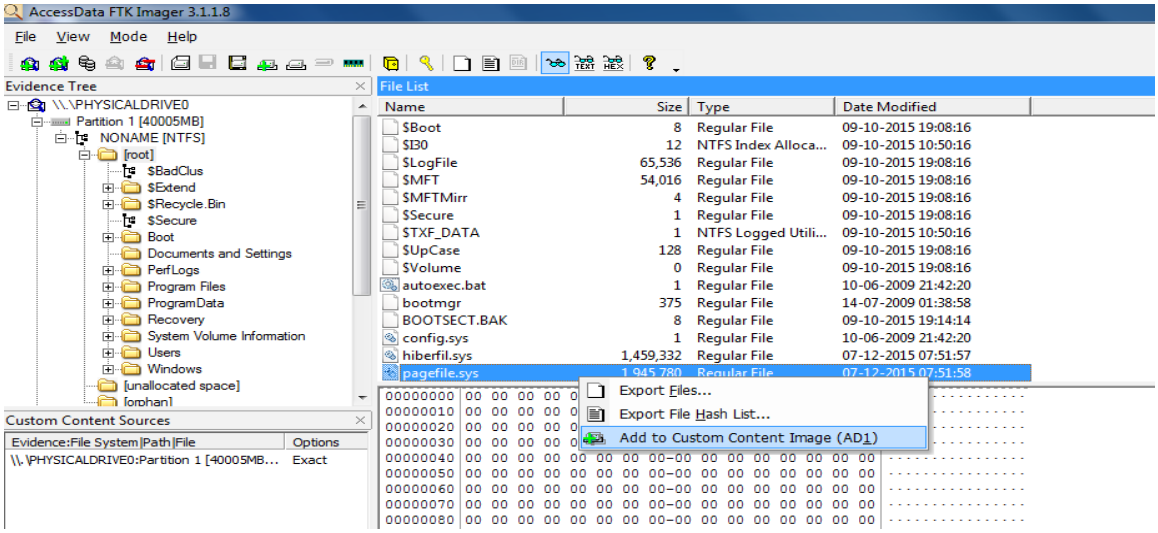
iv. **Registry Files:** The Windows Registry is a hierarchical database that stores low-level settings for the Microsoft Windows operating system and for applications that opt to use the Registry. The kernel, device drivers, services, Security Accounts Manager (SAM), and user interface can all use the Registry. The Registry also allows access to counters for profiling system performance. Registry files can be available at "C:\Windows\System32\config –Registry files".

The section below gives the procedure for collecting the windows operating system artifacts, which plays vital role in examination of malware traces, using FTK Imager.

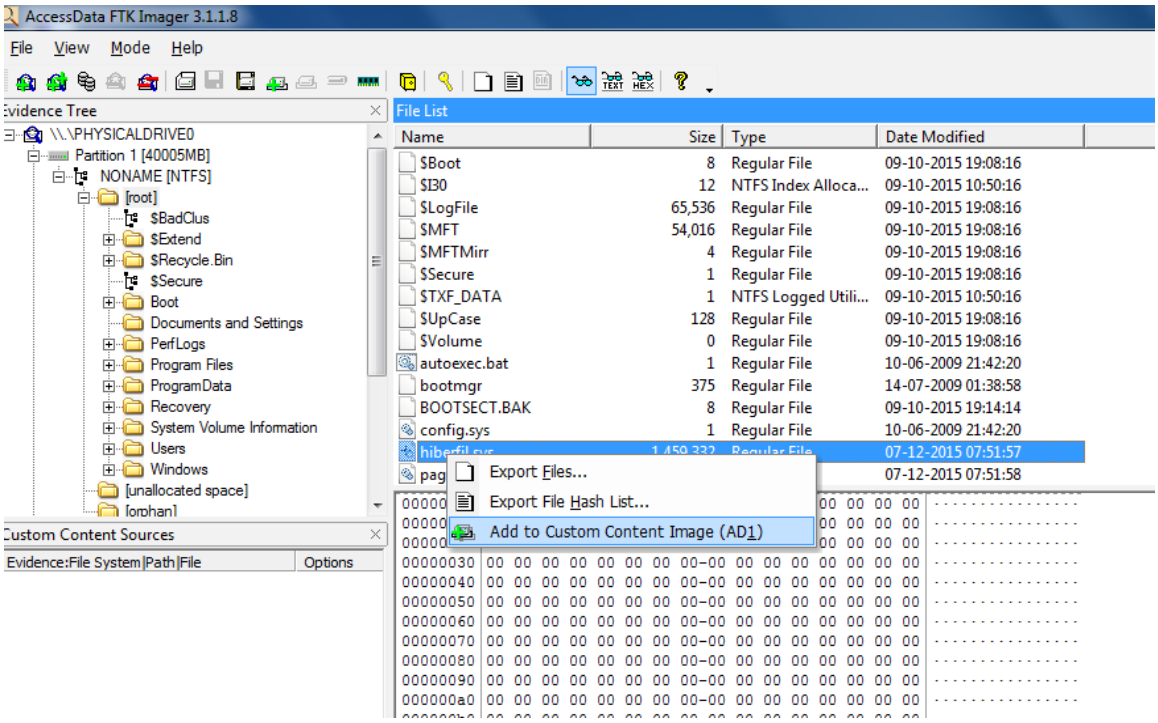
Source Drive Selection: Examiner shall be careful while selecting the Source Drive.



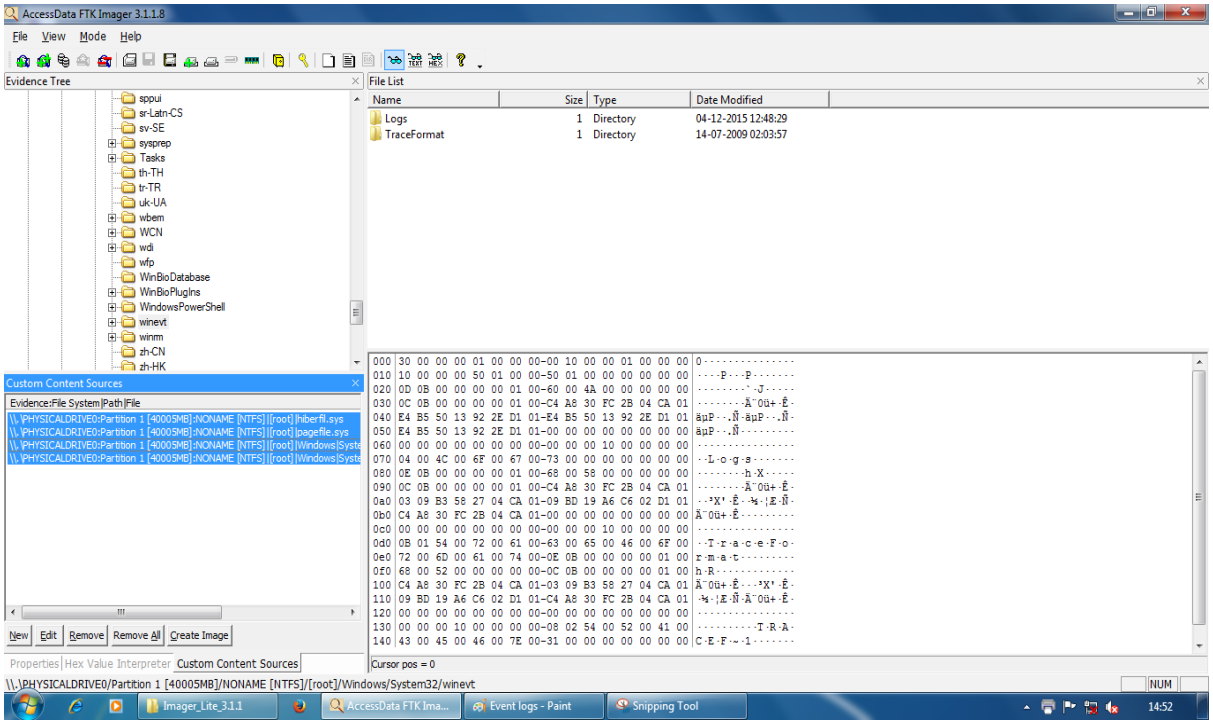
Selection of pagefile.sys and adding it for creating a custom content image in FTK.



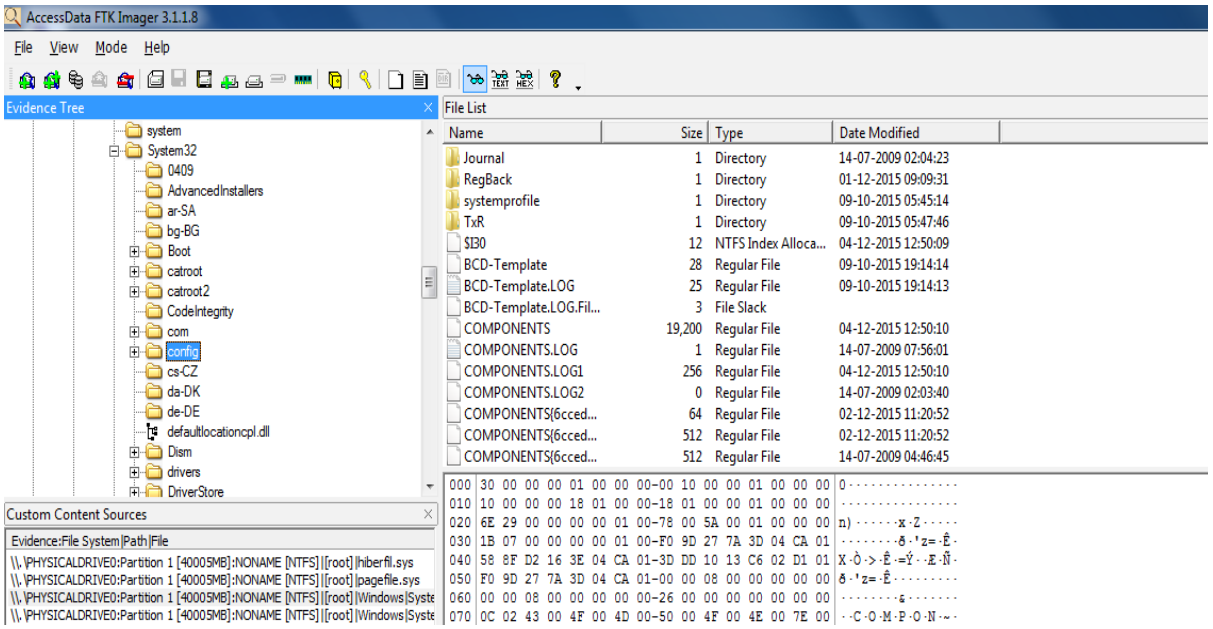
Selecting hiberfile.sys and adding it for creating a custom content image in FTK



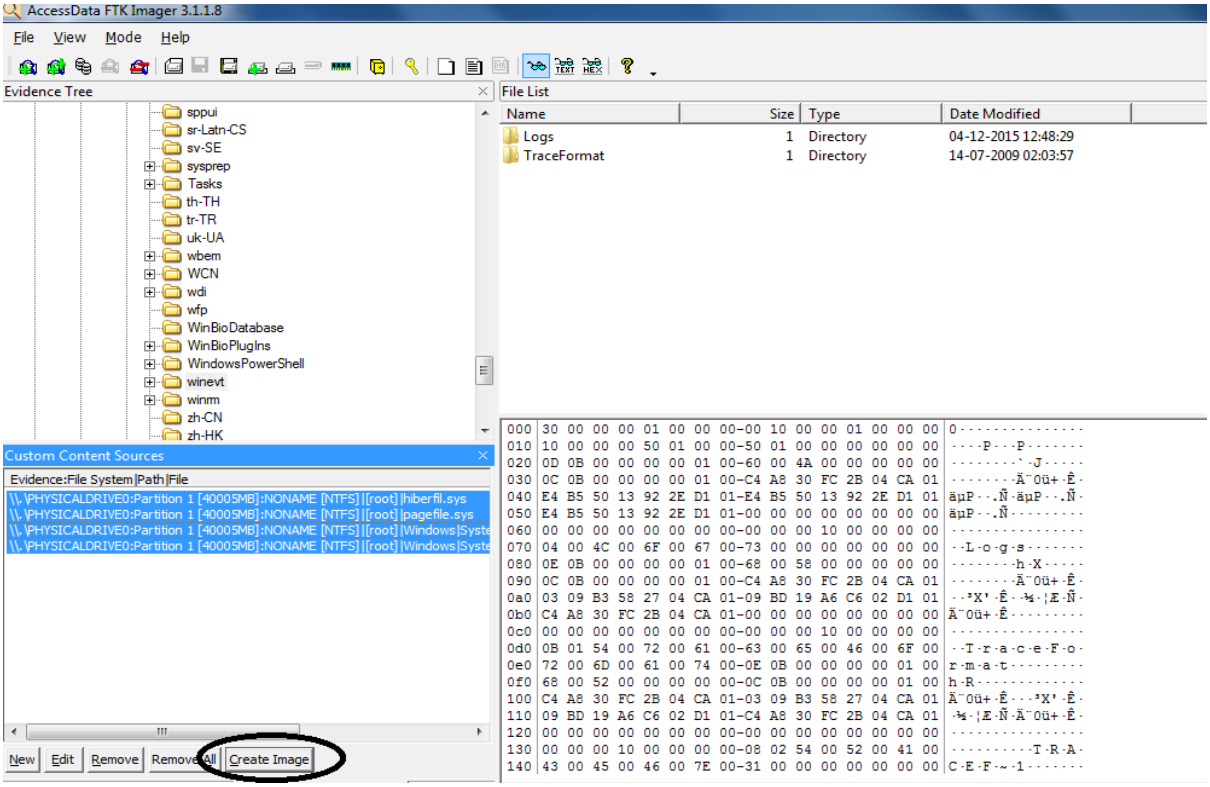
Selecting windows event logs and adding them for creating custom content image.



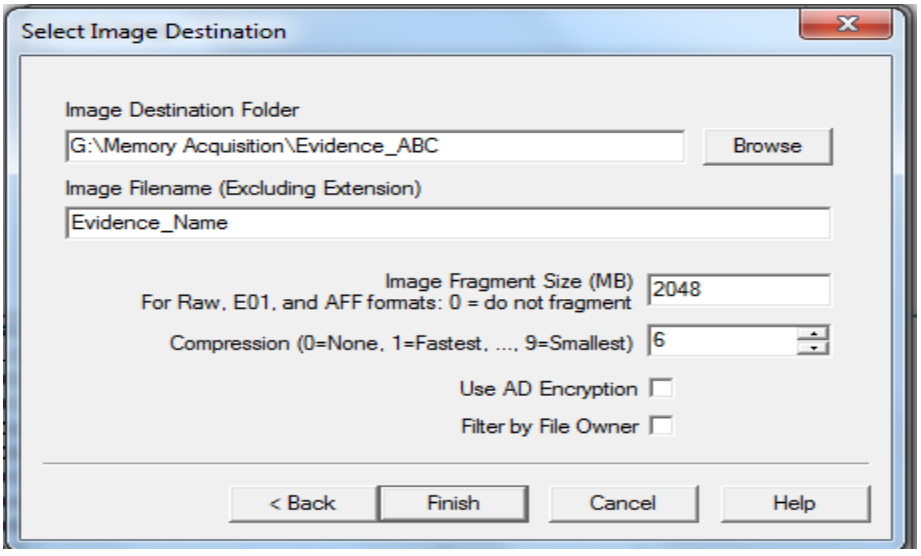
Selecting registry files and adding them for creating custom content image.



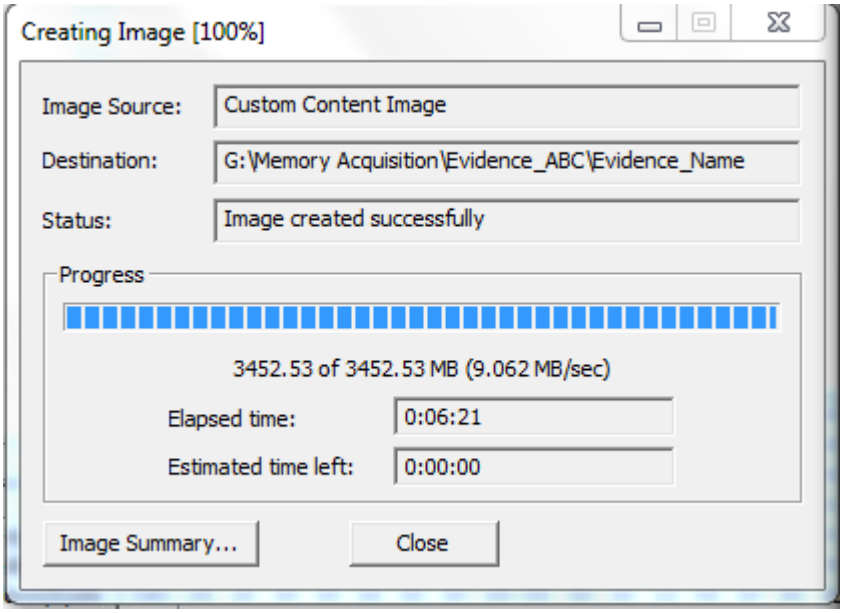
Post selection of relevant artefacts the list can be verified at the tab “Custom Content Sources” and forensic image of above stated artifacts can be created with the option “Create image”.



Providing recommended settings for the image attributes.



Successful Completion of Image



Once the above artifacts are collected examiner can start analyzing with some of the available basic static malware analysis techniques discussed below.

Step 3: Basic Static Malware Analysis:

The following techniques were used for performing basic static malware analysis.

i. File Fingerprinting
ii. Virus Scanning
iii. Analyzing memory artefacts (Pagefile.sys,hiberfile.sys)
iv. Packer Detection
v. Disassembly

i. File Fingerprinting

In this phase a cryptographic hash value for each file under investigation shall be computed. One of the most flexible is the open source command-line program md5deep3. Examiner can also rely on any forensic software like Encase and FTK.

ii. Virus Scanning

a. The first good step is to run it through multiple antivirus programs, which may already have identified the malware.
b. Websites like http://www.virustotal.com and http://virusscan.jotti.org allow you to upload files and have them scanned by a wide-variety of different scan engines and generates a report that provides the total number of engines that

marked the file as malicious, the malware name, and, if available, additional information about the malware.

iii. Analyzing memory artefacts

a. In the process of analyzing memory artefacts like [RAM dump, page file.sys, hiberfile.sys] examiner can start Identification of Rogue Process. An open source memory forensics tool named volatility can be used for malware analysis.

b. Open Command Prompt and go to the directory, where volatility frame work is present. Run the .exe file in command prompt with respective options.

C:\Users\Username\ volatility-2.3.1.standalone.exe –h (For help)

As we currently have memory dump with us, we now try to get the information about the image. The command to get the information about the image is **imageinfo.**

```
C:\Users\ \Desktop>volatility-2.3.1.standalone.exe -f image.dd imageinfo
Volatility Foundation Volatility Framework 2.3.1
Determining profile based on KDBG search...

          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with Win
XPSP2x86)
                     AS Layer1 : IA32PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (C:\Users\ICS\Desktop\image.dd
)
                      PAE type : No PAE
                           DTB : 0x39000L
                          KDBG : 0x8054cde0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2014-08-04 06:05:40 UTC+0000
     Image local date and time : 2014-08-04 11:35:40 +0530
```

To know the processes that were running on system at the time when the RAM was captured, the command is **pslist**.

```
C:\Users\  \Desktop>volatility-2.3.1.standalone.exe -f image.dd pslist
Volatility Foundation Volatility Framework 2.3.1
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x85fc6a00 System                    4      0     67     1070 ------      0
0x85e5f020 smss.exe                864      4      3       19 ------      0 2014-08-04 04:59:59 UTC+0000
0x85eb96e8 csrss.exe               968    864     12      520      0      0 2014-08-04 05:00:00 UTC+0000
0x85cd3da0 winlogon.exe            992    864     18      517      0      0 2014-08-04 05:00:00 UTC+0000
0x85ca8c08 services.exe           1036    992     17      367      0      0 2014-08-04 05:00:01 UTC+0000
0x85c71da0 lsass.exe              1048    992     20      374      0      0 2014-08-04 05:00:01 UTC+0000
0x85c6f958 svchost.exe            1212   1036     18      203      0      0 2014-08-04 05:00:01 UTC+0000
0x85bcc470 svchost.exe            1288   1036     10      249      0      0 2014-08-04 05:00:01 UTC+0000
0x858d8da0 svchost.exe            1400   1036     76     1374      0      0 2014-08-04 05:00:01 UTC+0000
0x85c30da0 svchost.exe            1656   1036      6       72      0      0 2014-08-04 05:00:01 UTC+0000
0x85dd1020 svchost.exe            1820   1036     14      200      0      0 2014-08-04 05:00:02 UTC+0000
0x85d53798 spoolsv.exe             120   1036     15      173      0      0 2014-08-04 05:00:03 UTC+0000
0x85d95798 econser.exe             444   1036      4       46      0      0 2014-08-04 05:00:09 UTC+0000
0x85b87798 econceal.exe            468    444     14      209      0      0 2014-08-04 05:00:09 UTC+0000
0x85b85798 avpmapp.exe             472   1036     35      353      0      0 2014-08-04 05:00:09 UTC+0000
0x85d0d798 traysser.exe            492   1036     12       97      0      0 2014-08-04 05:00:09 UTC+0000
0x85cf7798 jqs.exe                 612   1036      5      125      0      0 2014-08-04 05:00:09 UTC+0000
0x85ce6b48 consctl.exe             664    492     12      141      0      0 2014-08-04 05:00:09 UTC+0000
0x85c80518 explorer.exe           1240   1116     22     1442      0      0 2014-08-04 05:00:14 UTC+0000
0x85890da0 igfxpers.exe            380   1240      3      136      0      0 2014-08-04 05:00:19 UTC+0000
0x85573020 RTHDCPL.EXE             548   1240      4      171      0      0 2014-08-04 05:00:19 UTC+0000
0x85d07da0 TRAYICOS.EXE            648   1240      6      121      0      0 2014-08-04 05:00:19 UTC+0000
0x856eb510 jusched.exe             260   1240      1      125      0      0 2014-08-04 05:00:19 UTC+0000
0x85682510 ctfmon.exe              948   1240      3      152      0      0 2014-08-04 05:00:20 UTC+0000
0x85677da0 ewvije.exe             1688   1240      0 -------      0      0 2014-08-04 05:00:20 UTC+0000   2014-08-04 05:03:42 UTC
0x8556d360 alg.exe                2236   1036      5      106      0      0 2014-08-04 05:00:21 UTC+0000
0x8556d020 CyberoamClient.        2244   1240      2      174      0      0 2014-08-04 05:00:21 UTC+0000
0x85c20da0 escannon.exe           3288   1460     24      291      0      0 2014-08-04 05:00:28 UTC+0000
0x85661788 jucheck.exe            3696    260      5      267      0      0 2014-08-04 05:05:21 UTC+0000
0x85c2ca50 MWASER.EXE             1012   1036      4       52      0      0 2014-08-04 05:50:57 UTC+0000
0x85e86940 MWAGENT.EXE            1092   1012     14      187      0      0 2014-08-04 05:50:57 UTC+0000
0x856db9e8 helix.exe              1684   1240      9      321      0      0 2014-08-04 06:03:19 UTC+0000
0x854177e0 wmic.exe               3444   1684      0 -------      0      0 2014-08-04 06:03:51 UTC+0000   2014-08-04 06:04:02 UTC
0x85ac4020 cmd.exe                1676   1684      1       73      0      0 2014-08-04 06:05:39 UTC+0000
0x8563d1b0 dd.exe                 2180   1676      1       69      0      0 2014-08-04 06:05:40 UTC+0000
```

From **pslist** command you can get to know the process id for a particular process and also the parent process which triggered it. From this information we can know if any process is triggered by a suspicious parent process or if any suspicious process has created any child process.

To identify the hidden running process, we **psscan** command. This command gives same information as pslist, but provides additional info about hidden process.

```
C:\Users\  \Desktop>volatility-2.3.1.standalone.exe -f image.dd psscan
Volatility Foundation Volatility Framework 2.3.1
Offset(P)  Name                    PID   PPID PDB        Time created                   Time exited
---------- -------------------- ------ ------ ---------- ------------------------------ ------------------------------
0x054177e0 wmic.exe               3444   1684 0x1de32000 2014-08-04 06:03:51 UTC+0000   2014-08-04 06:04:02 UTC+0000
0x0556d020 CyberoamClient.        2244   1240 0x33080000 2014-08-04 05:00:21 UTC+0000
0x0556d360 alg.exe                2236   1036 0x32d6c000 2014-08-04 05:00:21 UTC+0000
0x05573020 RTHDCPL.EXE             548   1240 0x30759000 2014-08-04 05:00:19 UTC+0000
0x0563d1b0 dd.exe                 2180   1676 0x3ddd2000 2014-08-04 06:05:40 UTC+0000
0x05661788 jucheck.exe            3696    260 0x1174f000 2014-08-04 05:05:21 UTC+0000
0x05677da0 ewvije.exe             1688   1240 0x316ce000 2014-08-04 05:00:20 UTC+0000   2014-08-04 05:03:42 UTC+0000
0x05682510 ctfmon.exe              948   1240 0x30fe1000 2014-08-04 05:00:20 UTC+0000
0x056db9e8 helix.exe              1684   1240 0x266e2000 2014-08-04 06:03:19 UTC+0000
0x056eb510 jusched.exe             260   1240 0x30ac8000 2014-08-04 05:00:19 UTC+0000
0x05890da0 igfxpers.exe            380   1240 0x30858000 2014-08-04 05:00:19 UTC+0000
0x058d8da0 svchost.exe            1400   1036 0x1cd20000 2014-08-04 05:00:01 UTC+0000
0x05ac4020 cmd.exe                1676   1684 0x0daa8000 2014-08-04 06:05:39 UTC+0000
0x05b5c798 consctl.exe             688    504 0x1f82a000 2014-08-04 04:08:08 UTC+0000
0x05b85798 avpmapp.exe             472   1036 0x1f422000 2014-08-04 05:00:09 UTC+0000
0x05b87798 econceal.exe            468    444 0x1f7c2000 2014-08-04 05:00:09 UTC+0000
0x05bcc470 svchost.exe            1288   1036 0x1c89a000 2014-08-04 05:00:01 UTC+0000
0x05c20da0 escannon.exe           3288   1460 0x3783a000 2014-08-04 05:00:28 UTC+0000
0x05c2ca50 MWASER.EXE             1012   1036 0x18350000 2014-08-04 05:50:57 UTC+0000
0x05c30da0 svchost.exe            1656   1036 0x1cd36000 2014-08-04 05:00:01 UTC+0000
0x05c6f958 svchost.exe            1212   1036 0x1c4a5000 2014-08-04 05:00:01 UTC+0000
0x05c71da0 lsass.exe              1048    992 0x1bd75000 2014-08-04 05:00:01 UTC+0000
0x05c7f9d8 MWAGENT.EXE            1668    784 0x2f5b6000 2014-08-04 05:00:18 UTC+0000   2014-08-04 05:50:57 UTC+0000
0x05c88518 explorer.exe           1240   1116 0x2b0d1000 2014-08-04 05:00:14 UTC+0000
0x05ca8c08 services.exe           1036    992 0x1bd0b000 2014-08-04 05:00:01 UTC+0000
0x05cd3da0 winlogon.exe            992    864 0x1b8d3000 2014-08-04 05:00:00 UTC+0000
0x05ce6b48 consctl.exe             664    492 0x1fcaa000 2014-08-04 05:00:09 UTC+0000
0x05cf7798 jqs.exe                 612   1036 0x1fc15000 2014-08-04 05:00:09 UTC+0000
0x05d07da0 TRAYICOS.EXE            648   1240 0x306b6000 2014-08-04 05:00:19 UTC+0000
0x05d0d798 traysser.exe            492   1036 0x1f54f000 2014-08-04 05:00:09 UTC+0000
0x05d53798 spoolsv.exe             120   1036 0x1ed8a000 2014-08-04 05:00:03 UTC+0000
0x05d95798 econser.exe             444   1036 0x1f1bf000 2014-08-04 05:00:09 UTC+0000
0x05dd1020 svchost.exe            1820   1036 0x1ddff000 2014-08-04 05:00:02 UTC+0000
0x05e5f020 smss.exe                864      4 0x18f2d000 2014-08-04 04:57:59 UTC+0000
0x05e86940 MWAGENT.EXE            1092   1012 0x1d983000 2014-08-04 05:50:57 UTC+0000
0x05eb96e8 csrss.exe               968    864 0x1a60e000 2014-08-04 05:00:00 UTC+0000
0x05fc6a00 System                    4      0 0x00039000
0x06888a00 System                    4      0 0x00039000
0x06e29da0 escannon.exe           3288   1460 0x3783a000 2014-08-04 05:00:28 UTC+0000
0x0a259da0 svchost.exe            1656   1036 0x1cd36000 2014-08-04 05:00:01 UTC+0000
0x0d6c4798 econser.exe             444   1036 0x1f1bf000 2014-08-04 05:00:09 UTC+0000
0x0da48da0 winlogon.exe            992    864 0x1b8d3000 2014-08-04 05:00:00 UTC+0000
0x0dcc0020 svchost.exe            1820   1036 0x1ddff000 2014-08-04 05:00:02 UTC+0000
0x0fe929d0 MWAGENT.EXE            1668    784 0x2f5b6000 2014-08-04 05:00:18 UTC+0000   2014-08-04 05:50:57 UTC+0000
0x105256e8 csrss.exe               968    864 0x1a60e000 2014-08-04 05:00:00 UTC+0000
0x12475020 cmd.exe                1676   1684 0x0daa8000 2014-08-04 06:05:39 UTC+0000
0x125fcb48 consctl.exe             664    492 0x1fcaa000 2014-08-04 05:00:09 UTC+0000
0x12853518 explorer.exe           1240   1116 0x2b0d1000 2014-08-04 05:00:14 UTC+0000
0x13a57020 svchost.exe            1820   1036 0x1ddff000 2014-08-04 05:00:02 UTC+0000
0x16c151b0 dd.exe                 2180   1676 0x3ddd2000 2014-08-04 06:05:40 UTC+0000
0x172b2da0 svchost.exe            1400   1036 0x1cd20000 2014-08-04 05:00:01 UTC+0000
0x19039d0 MWAGENT.EXE             1668    784 0x2f5b6000 2014-08-04 05:00:18 UTC+0000   2014-08-04 05:50:57 UTC+0000
0x1c1b1510 ctfmon.exe              948   1240 0x30fe1000 2014-08-04 05:00:20 UTC+0000
0x1e19dda0 TRAYICOS.EXE            648   1240 0x306b6000 2014-08-04 05:00:19 UTC+0000
0x202f5a50 MWASER.EXE             1012   1036 0x18350000 2014-08-04 05:50:57 UTC+0000
0x21d53798 traysser.exe            492   1036 0x1f54f000 2014-08-04 05:00:09 UTC+0000
0x22f23940 MWAGENT.EXE            1092   1012 0x1d983000 2014-08-04 05:50:57 UTC+0000
0x23bceda0 escannon.exe           3288   1460 0x3783a000 2014-08-04 05:00:28 UTC+0000
0x27edb798 spoolsv.exe             120   1036 0x1ed8a000 2014-08-04 05:00:03 UTC+0000
0x2851c510 jusched.exe             260   1240 0x30ac8000 2014-08-04 05:00:19 UTC+0000
0x285aa798 avpmapp.exe             472   1036 0x1f422000 2014-08-04 05:00:09 UTC+0000
0x285c2958 svchost.exe            1212   1036 0x1c4a5000 2014-08-04 05:00:01 UTC+0000
0x2ebe1940 MWAGENT.EXE            1092   1012 0x1d983000 2014-08-04 05:50:57 UTC+0000
0x39629798 traysser.exe            492   1036 0x1f54f000 2014-08-04 05:00:09 UTC+0000
```

To check if any process has tried to open any TCP connection, we have **connscan** command. This will list all the process which tried to establish any TCP connections.

```
C:\Users\[redacted]\Desktop>volatility-2.3.1.standalone.exe -f image.dd connscan
Volatility Foundation Volatility Framework 2.3.1
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------ ------------------------- ---
0x05554008 127.0.0.1:1229           127.0.0.1:2226            472
0x055f5008 127.0.0.1:1416           127.0.0.1:2226            664
0x0571f008 127.0.0.1:2226           127.0.0.1:1229            1668
0x05721008 0.0.0.0:1498             24.177.33.91:2297         1240
0x05b77860 111.112.113.52:1075      216.163.188.45:80         3288
0x05c10e68 111.112.113.52:1025      111.112.113.199:445       4
0x05c1a008 111.112.113.52:1116      184.27.23.30:443          3696
0x05dbb860 111.112.113.52:1074      216.163.188.45:80         3288
0x071ca860 111.112.113.52:1074      216.163.188.45:80         3288
0x08c26008 127.0.0.1:1229           127.0.0.1:2226            472
0x094f0008 127.0.0.1:2226           127.0.0.1:1229            1668
0x12e83008 111.112.113.52:1116      184.27.23.30:443          3696
0x152d7e68 111.112.113.52:1025      111.112.113.199:445       4
0x1a589008 127.0.0.1:1416           127.0.0.1:2226            664
0x241b0008 127.0.0.1:1229           127.0.0.1:2226            472
```

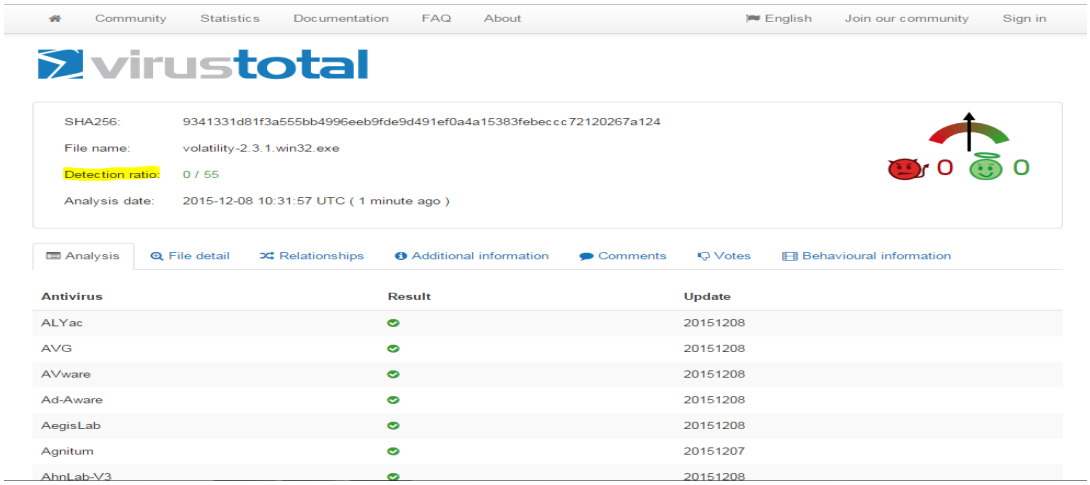
To identify any rogue process in the RAM dump, we have a plugin name **malfind**. It will identify the process using **psscan** command and it also search using YARA rules.

```
C:\Users\[redacted]\Desktop>volatility-2.3.1.standalone.exe -f image.dd malfind > C:\Users\[redacted]\Desktop\Mally.txt
Volatility Foundation Volatility Framework 2.3.1
```

Once you identify the rogue process, we need to export the rogue process for further analysis. To dump a process the command used is **procexedump** and **--dump-dir** directory to save the extracted information.

```
C:\Users\[redacted]\Desktop>volatility-2.3.1.standalone.exe procexedump -f image.dd --dump-dir "C:\Users\[redacted]\Desktop\EXE"
Volatility Foundation Volatility Framework 2.3.1
Process(V) ImageBase  Name                 Result
---------- ---------- -------------------- ------
0x85fc6a00 ---------- System               Error: PEB at 0x0 is paged
0x85e5f020 0x48580000 smss.exe             OK: executable.864.exe
0x85eb96e8 0x4a680000 csrss.exe            Error: ImageBaseAddress at 0x4a680000 is paged
0x85cd3da0 0x01000000 winlogon.exe         OK: executable.992.exe
0x85ca8c08 0x01000000 services.exe         OK: executable.1036.exe
0x85c71da0 0x01000000 lsass.exe            Error: ImageBaseAddress at 0x1000000 is paged
0x85c6f958 0x01000000 svchost.exe          OK: executable.1212.exe
0x85bcc470 0x01000000 svchost.exe          OK: executable.1288.exe
0x858d8da0 0x01000000 svchost.exe          OK: executable.1400.exe
0x85c30da0 0x01000000 svchost.exe          OK: executable.1656.exe
0x85dd1020 0x01000000 svchost.exe          OK: executable.1820.exe
0x85d53798 0x01000000 spoolsv.exe          OK: executable.120.exe
0x85d95798 0x00400000 econser.exe          Error: ImageBaseAddress at 0x400000 is paged
0x85b87798 0x00400000 econceal.exe         OK: executable.468.exe
0x85b85798 0x00400000 avpmapp.exe          Error: ImageBaseAddress at 0x400000 is paged
0x85d0d798 0x00400000 traysser.exe         Error: ImageBaseAddress at 0x400000 is paged
0x85cf7798 0x00400000 jqs.exe              OK: executable.612.exe
0x85ce6b48 0x00400000 consctl.exe          OK: executable.664.exe
0x85c80518 0x01000000 explorer.exe         OK: executable.1240.exe
0x85890da0 0x00400000 igfxpers.exe         OK: executable.380.exe
0x85573020 0x00400000 RTHDCPL.EXE          OK: executable.548.exe
0x85d07da0 0x00400000 TRAYICOS.EXE         Error: ImageBaseAddress at 0x400000 is paged
0x856eb510 0x00400000 jusched.exe          OK: executable.260.exe
0x85682510 0x00400000 ctfmon.exe           OK: executable.948.exe
0x85677da0 ---------- envije.exe            Error: PEB at 0x7ffd7000 is paged
0x8556d360 0x01000000 alg.exe              OK: executable.2236.exe
0x8556d020 0x00400000 CyberoamClient.      OK: executable.2244.exe
0x85c20da0 0x00400000 escanmon.exe         OK: executable.3288.exe
0x85661788 0x00400000 jucheck.exe          OK: executable.3696.exe
0x85c2ca50 0x00400000 MUASER.EXE           Error: ImageBaseAddress at 0x400000 is paged
0x85e86940 0x00400000 MUAGENT.EXE          Error: ImageBaseAddress at 0x400000 is paged
0x856db9e8 0x00400000 helix.exe            OK: executable.1684.exe
0x854177e0 ---------- wmic.exe             Error: PEB at 0x7ffda000 is paged
0x85ac4020 0x4ad00000 cmd.exe              Error: ImageBaseAddress at 0x4ad00000 is paged
0x8563d1b0 0x00400000 dd.exe               Error: ImageBaseAddress at 0x400000 is paged
```

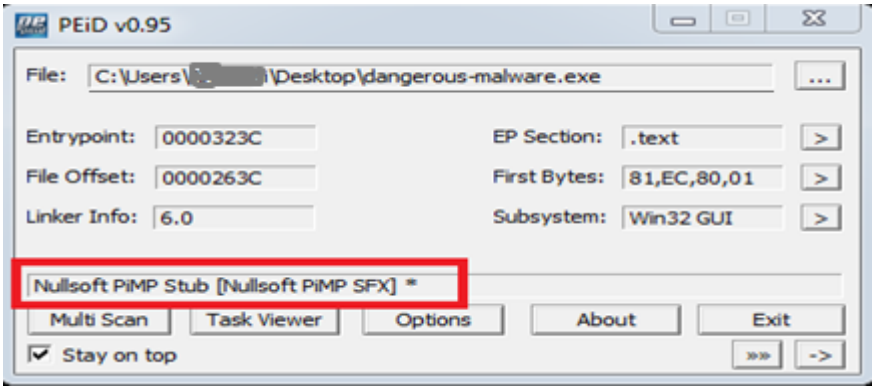
Once the process is exported, we need to calculate the hash of that particular process and submit it to www.virustotal.com. Virus total can check the list of hash values against 55 antivirus definitions and will give detection ratio. Incident responder should be familiar with legitimate processes.



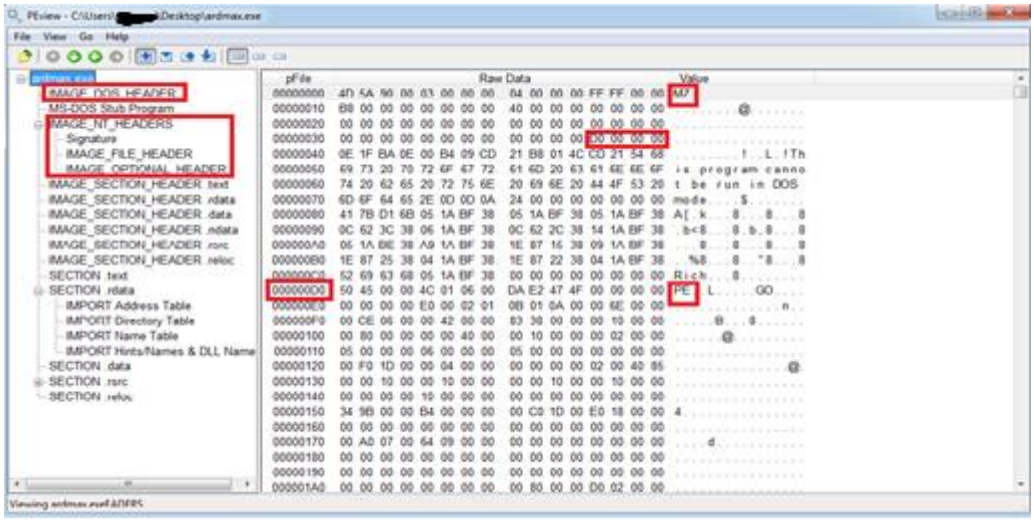
iv. Packer Detection

Obfuscated programs are ones whose execution was attempted to hide by malware author. Packed programs are programs in which the malicious program is compressed and cannot be analyzed. These packers can be detected by PEiD [https://www.aldeid.com/wiki/PEiD].

Detecting Packers with PEiD: When a program is packed, you must unpack it in order to perform basic static analysis. PEiD is used to detect the type of packer or compiler employed to build an application, which makes analyzing the packed file much easier.



PE Structure using PEView



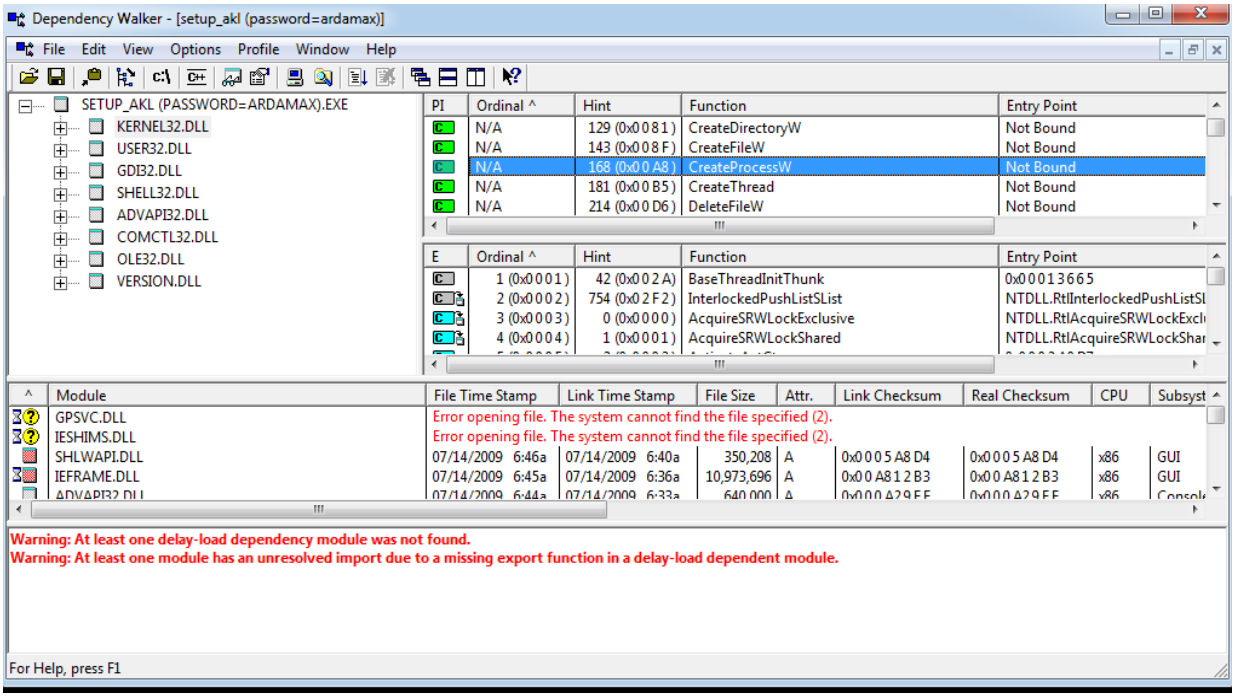
Analyzing PE Structure:

A PE file contains a header and some more important sections, analysis of these fields can help examiner to identify interesting fields of code.

- .text: This contains the executable code.
- .rdata: This sections holds read only globally accessible data.
- .data: Stores global data accessed through the program.
- .rsrc: This sections stores resources needed by the executable.

v. Debugging

Most often malware authors use dynamic linking in their code. DLL files can be analyzed with Dependency walker.



Conclusion

Identification, collection and preliminary analysis of malware is very much necessary to contain the spread of malware and to reduce the impact of malware security incident. This will also help in reducing the business downtime, cost of investigation and cleaning. The need of the hour is every SoC team member should have fundamental knowledge of malware analysis to contain the spread of malware.