CySA+ Lab Series

Lab 10: Memory Forensics Analysis

Document Version: 2024-02-05

Material in this Lab Aligns to the Following	
CompTIA CySA+ (CS0-002) Exam Objectives	1.7 - Given a scenario, implement controls to mitigate attacks and software vulnerabilities 4.3 - Given an incident, analyze potential indicators of compromise 4.4 - Given a scenario, utilize basic digital forensics techniques
All-In-One CompTIA CySA+ Second Edition ISBN-13: 978-1260464306 Chapters	7: Mitigating Controls for Attacks and Software Vulnerabilities 17: Analyze Potential Indicators of Compromise 18: Utilize Basic Digital Forensics Techniques

Contents

Introduction

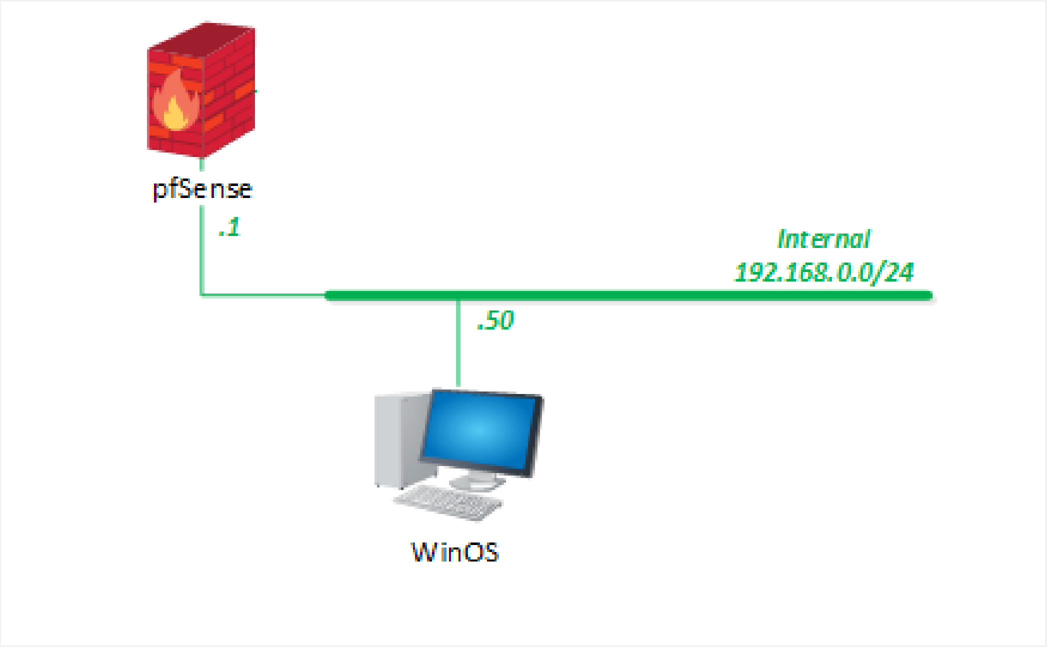
Memory Forensics is an important component of digital forensics. If a system has been compromised, a security analyst needs to be able to take a memory snapshot of the host, which can then be analyzed for evidence of malicious activities.

In this lab, you will use tools to create a memory image. Once the memory image is created, you will use additional tools to analyze the file.

Objective

- Capturing the current RAM contents
- Analysis of a captured memory image

Lab Topology



Lab Settings

The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	IP Address	Account	Password
WinOS (Server 2019)	`192.168.0.50`	`Administrator`	`NDGlabpass123!`
MintOS (Linux Mint)	`192.168.0.60`	`sysadmin`	`NDGlabpass123!`
OSSIM (Alien Vault)	`172.16.1.2`	`root`	`NDGlabpass123!`
UbuntuSRV (Ubuntu Server)	`172.16.1.10`	`sysadmin`	`NDGlabpass123!`
Kali	`203.0.113.2`	`sysadmin`	`NDGlabpass123!`

pfSense	`203.0.113.1`
	`172.16.1.1`
	`192.168.0.1`
		`admin`	`NDGlabpass123!`

1 Creating Memory Image Files for Analysis

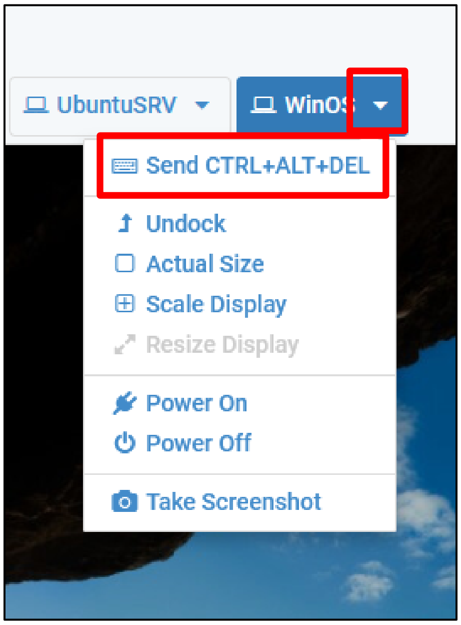
Fileless malware is loaded into and run from memory. When this happens, a security analyst needs to be able to investigate and analyze the memory of a compromised system. Since the information stored in RAM is lost when a computer is powered down, being able to capture the information immediately after a security incident is invaluable.

Memory dumps may contain the passwords to volumes that have been encrypted by tools such as *TrueCrypt* and *BitLocker*. There also may be account login credentials for webmail and social networks.

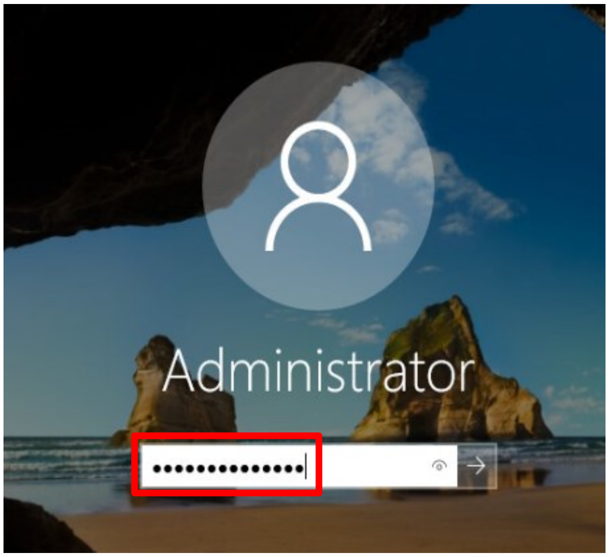
1.1 Using DumpIt for Windows

DumpIt is a great Windows tool from Comae Technologies to capture a dump of memory from a compromised system. In this task, you will capture the current contents of the system's memory using *DumpIt* and create a dump file for analysis.

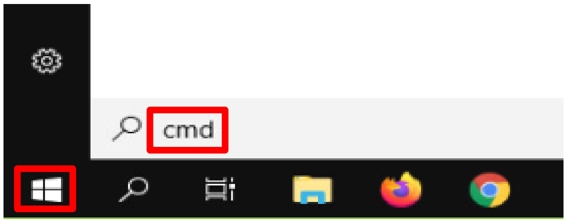
1. Set the focus on the **WinOS** computer.
2. Bring up the login window by sending a Ctrl + Alt + Delete. To do this, click the **WinOS** dropdown menu and click **Send CTRL+ALT+DEL**.



3. Log in as *Administrator* using the password: `NDGlabpass123!`

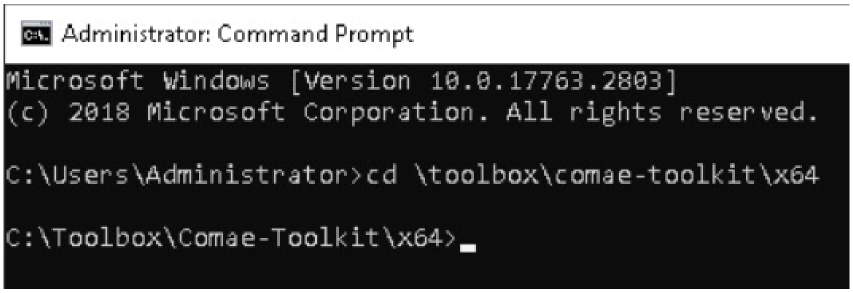


4. Click on the **Windows Start** button in the bottom-left corner and type `CMD` and press the **ENTER** key to bring up the command prompt window.



5. In the command prompt window, type the following command to open the **Comae-Toolbox** folder.

```
cd \toolbox\comae-toolkit\x64
```

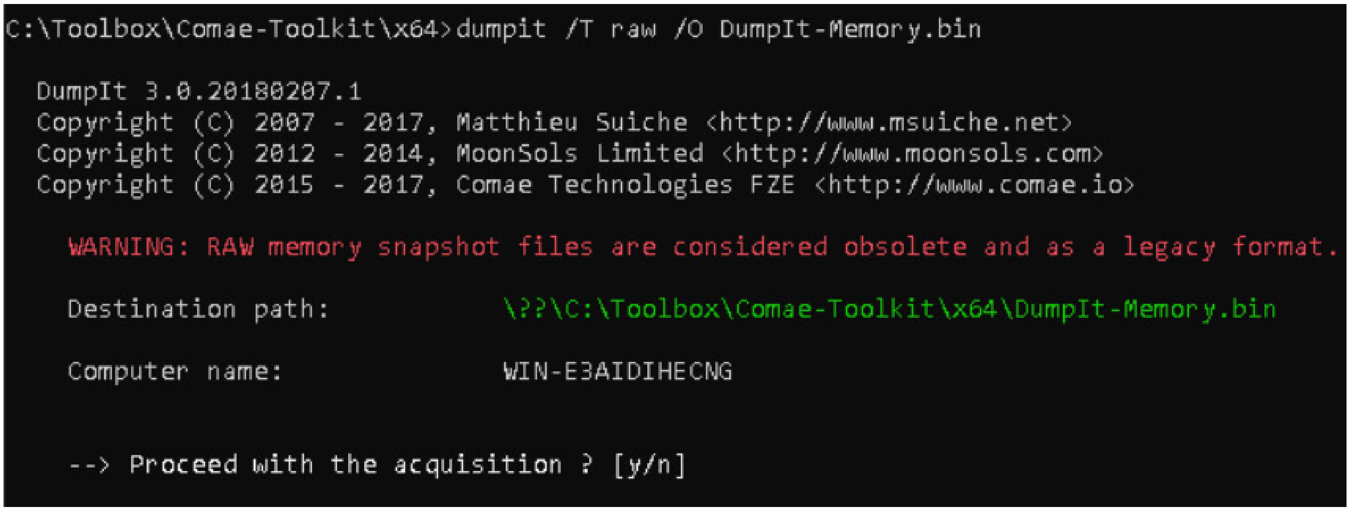


Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.17763.2803]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd \toolbox\comae-toolkit\x64

C:\Toolbox\Comae-Toolkit\x64>_
```

6. Run **DumpIt** by typing the following command:

```
dumpit /T raw /O DumpIt-Memory.bin
```



```
C:\Toolbox\Comae-Toolkit\x64>dumpit /T raw /O DumpIt-Memory.bin

  DumpIt 3.0.20180207.1
  Copyright (C) 2007 - 2017, Matthieu Suiche <http://www.msuiche.net>
  Copyright (C) 2012 - 2014, MoonSols Limited <http://www.moonsols.com>
  Copyright (C) 2015 - 2017, Comae Technologies FZE <http://www.comae.io>

    WARNING: RAW memory snapshot files are considered obsolete and as a legacy format.

    Destination path:           \??\C:\Toolbox\Comae-Toolkit\x64\DumpIt-Memory.bin

    Computer name:              WIN-E3AIDIHECNG


    --> Proceed with the acquisition ? [y/n]
```

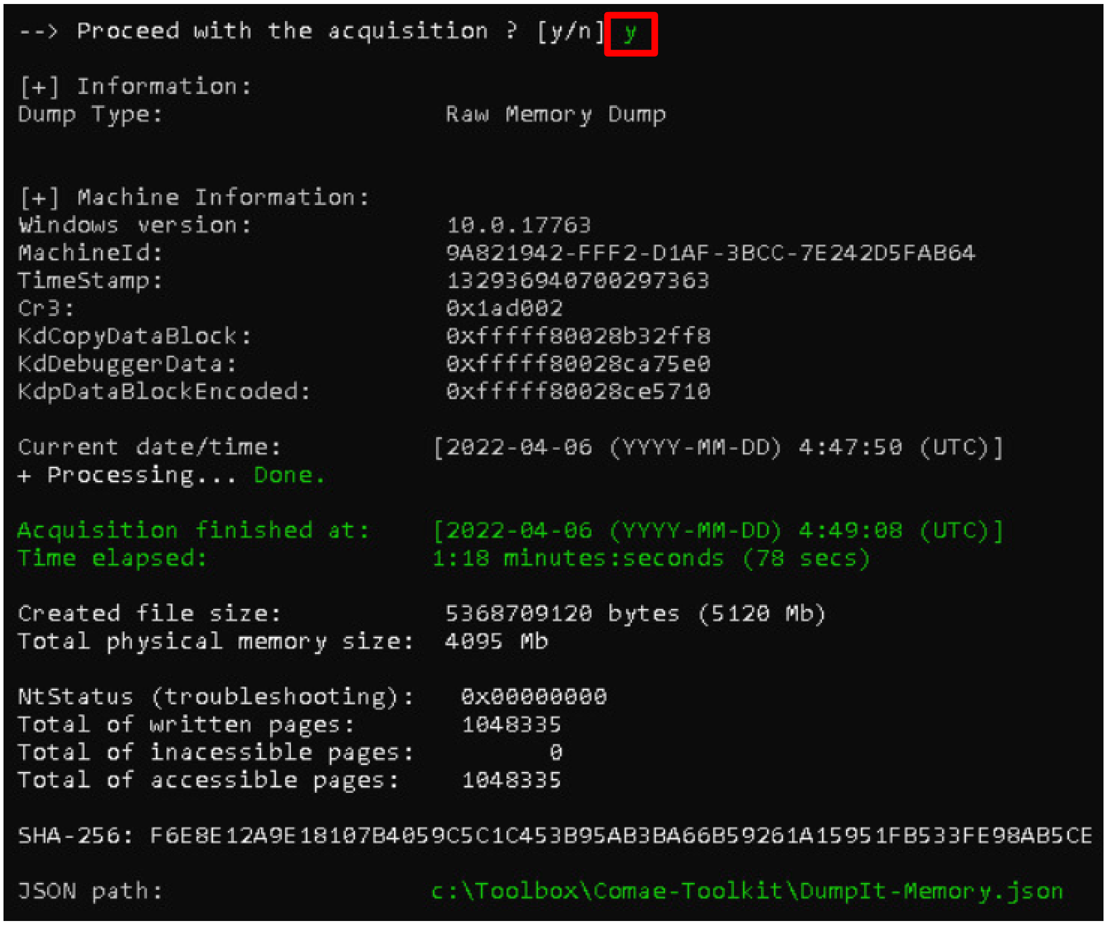
The **/T** raw option tells *DumpIt* to create the dump file in RAW format

Other formats are **BIN** and **MEM**. These formats are not compatible with the Windows *Volatility* tool.

The **/O** DumpIt-Memory.bin option tells *DumpIt* to name the dump file **DumpIt-Memory.bin**.

If you do not put in a file path, *DumpIt* will store the file in the DumpIt directory as indicated in the destination path.

7. At the *Proceed with the acquisition* prompt, type Y**.**



```
--> Proceed with the acquisition ? [y/n] y

[+] Information:
Dump Type:                     Raw Memory Dump


[+] Machine Information:
Windows version:               10.0.17763
MachineId:                     9A821942-FFF2-D1AF-3BCC-7E242D5FAB64
TimeStamp:                     132936940700297363
Cr3:                           0x1ad002
KdCopyDataBlock:               0xfffff80028b32ff8
KdDebuggerData:                0xfffff80028ca75e0
KdpDataBlockEncoded:           0xfffff80028ce5710

Current date/time:            [2022-04-06 (YYYY-MM-DD) 4:47:50 (UTC)]
+ Processing... Done.

Acquisition finished at:      [2022-04-06 (YYYY-MM-DD) 4:49:08 (UTC)]
Time elapsed:                 1:18 minutes:seconds (78 secs)

Created file size:             5368709120 bytes (5120 Mb)
Total physical memory size:    4095 Mb

NtStatus (troubleshooting):     0x00000000
Total of written pages:         1048335
Total of inaccessible pages:          0
Total of accessible pages:      1048335

SHA-256: F6E8E12A9E18107B4059C5C1C453B95AB3BA66B59261A15951FB533FE98AB5CE

JSON path:                    c:\Toolbox\Comae-Toolkit\DumpIt-Memory.json
```

8. After about a minute, the dump should be completed. Minimize the **Command Prompt** window.

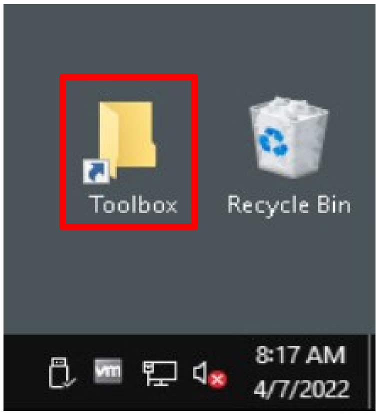
> After the command finishes, you will see output similar to what is shown above. (The dates and times may vary.) Notice that a *JSON* file was also created, containing metadata in *Javascript* format.

9. Remain on the *WinOS* computer and proceed to the next task.

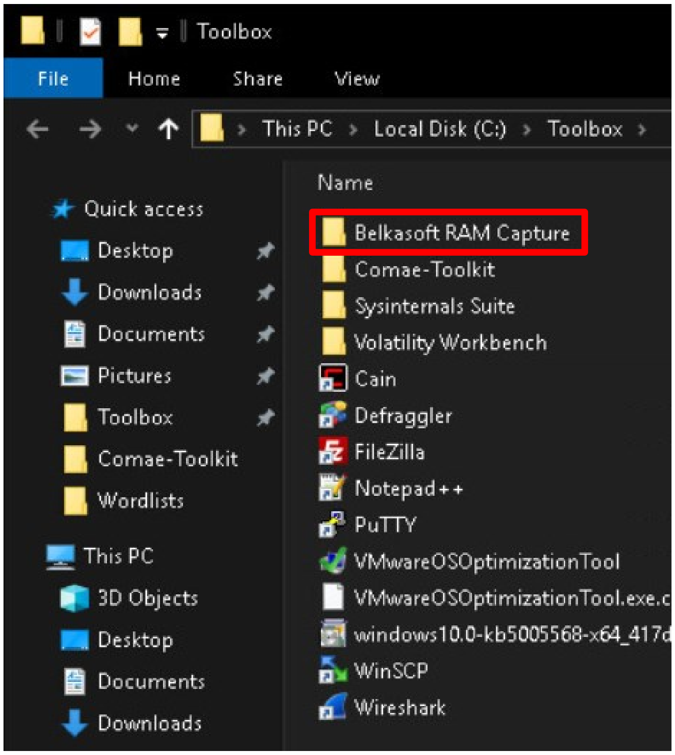
1.2 Using Belkasoft RAM Capture for Windows

Belkasoft Forensics' *Live RAM Capture* is another good forensics and analysis tool for capturing the contents of a RAM. It has a small footprint and does not require installation, which could possibly affect the results of the scan. It operates in "kernel mode" as opposed to "user mode" which will bypass anti-debugging and anti-memory dumping protections by operating on the same level as these protection systems and can acquire application address space correctly.

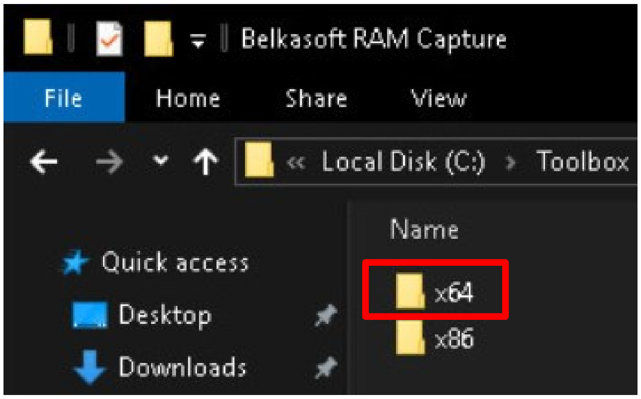
1. Double-click on the **Toolbox** folder.



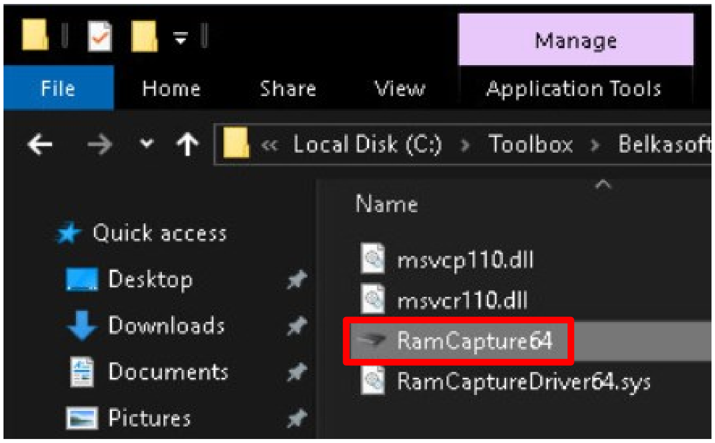
2. In the *File Explorer* window, double-click on the **Belkasoft RAM Capture** folder.



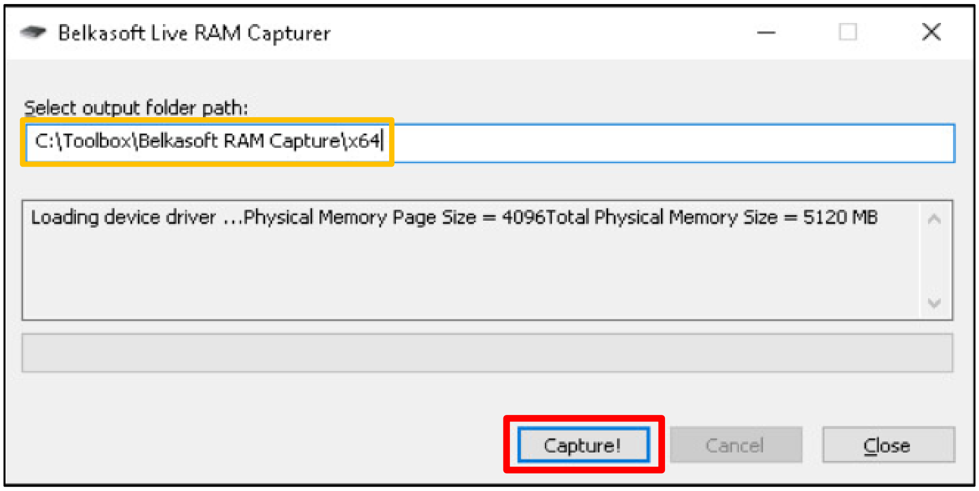
3. Double-click on the **x64** folder.



4. Start the program by double-clicking on **RamCapture64**.



5. In the *Select output folder path* box, leave the **C:\Toolbox\Belkasoft RAM Capture\x64** directory and click the **Capture!** button.

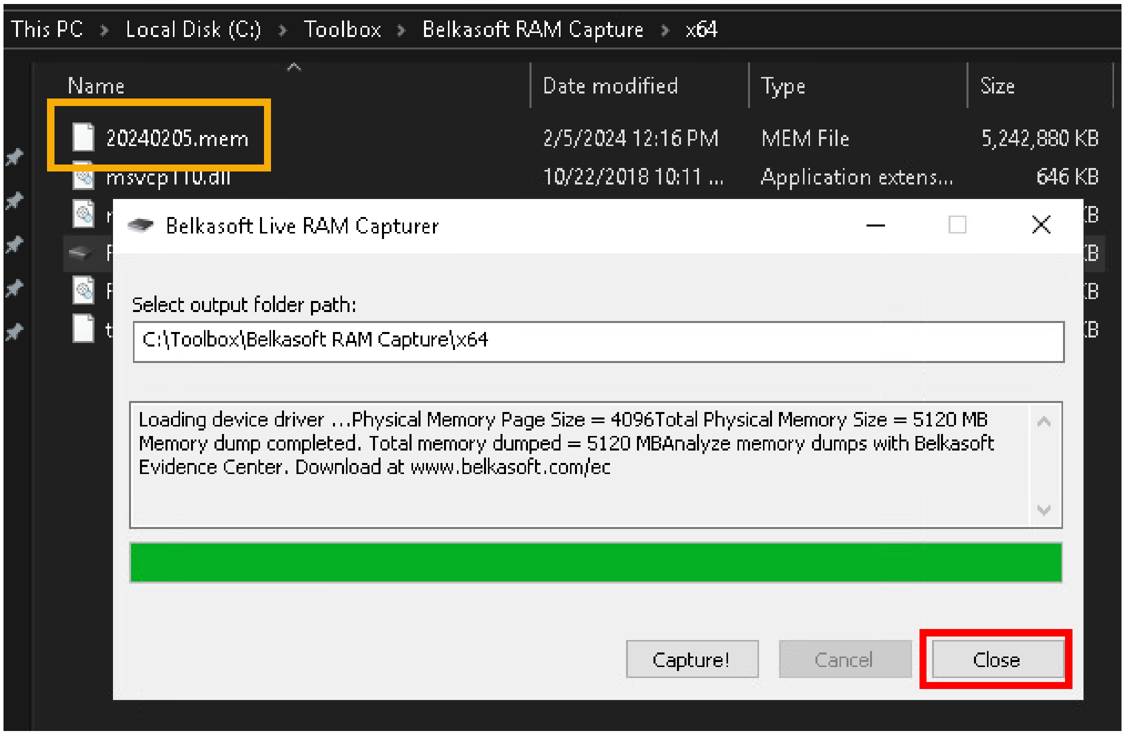


Select output folder path:

C:\Toolbox\Belkasoft RAM Capture\x64

```
Loading device driver ...Physical Memory Page Size = 4096Total Physical Memory Size = 5120 MB
```

6. When the capture has completed, you will see the dump file in the folder. It will have the date of the dump with the **.mem** extension. Make a note of the name of the file; you will be using it in the next task. Click the **Close** button to close the *Live RAM Capturer* window.



Select output folder path:
C:\Toolbox\Belkasoft RAM Capture\x64

```
Loading device driver ...Physical Memory Page Size = 4096Total Physical Memory Size = 5120 MB
Memory dump completed. Total memory dumped = 5120 MBAnalyze memory dumps with Belkasoft
Evidence Center. Download at www.belkasoft.com/ec
```

7. Minimize the **File Explorer** window.
8. Remain on the *WinOS* machine and proceed to the next task.

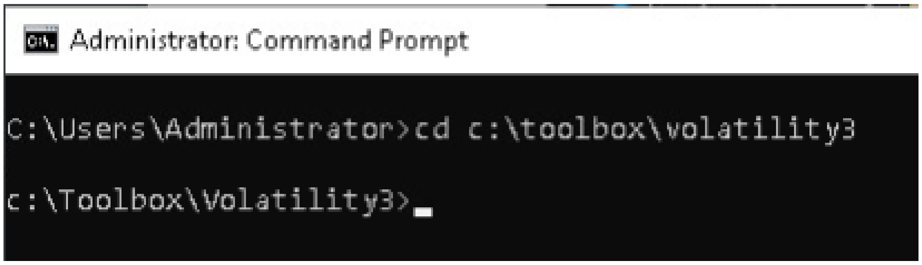
2 Analyzing the Memory Image

The memory dump is only the first part of memory forensics. Once the dump is created, a security analyst must then use memory analysis tools to detect malicious activities.

One of the "go to" tools used by security analysts is *Volatility*, a command-line memory analysis and forensics tool that is useful in discovering artifacts, such as running processes that might have malicious code injected, into the network connections and services. There are versions of *Volatility* for Windows, Mac, and Linux.

1. Restore the command prompt window. Type the following command to open the *Volatility3* folder:

```
cd c:\toolbox\volatility3
```



```
C:\Users\Administrator>cd c:\toolbox\volatility3

c:\Toolbox\Volatility3>_
```

The output lines from many of the *Volatility* scans are quite long. You should increase the width of the Command Prompt window to the width of the Windows Desktop.

Volatility parses and extracts interesting information from memory dumps by using **plugins** that have been developed. The **Volatility Workbench** will only show plugins that are generally useful, but additional plugins are available and can be downloaded and installed.

> *"Volatility has two main approaches to plugins, which are sometimes reflected in their names. "list" plugins will try to navigate through Windows Kernel structures to retrieve information like processes (locate and walk the linked list of _EPROCESS structures in memory), OS handles (locating and listing the handle table, dereferencing any pointers found, etc). They more or less behave like the Windows API would if requested to, for example, list processes.*
>
> *That makes "list" plugins pretty fast, but just as vulnerable as the Windows API to manipulation by malware. For instance, if malware uses DKOM to unlink a process from the _EPROCESS linked list, it won't show up in the Task Manager and neither will it in the pslist.*
>
> *"scan" plugins, on the other hand, will take an approach similar to carving the memory for things that might make sense when dereferenced as specific structures. psscan for instance will read the memory and try to make out _EPROCESS objects out of it (it uses pool-tag scanning, which is basically searching for 4-byte strings that indicate the presence of a structure of interest). The advantage is that it can dig up processes that have exited, and even if malware tampers with the _EPROCESS linked list, the plugin will still find the structure lying around in memory (since it still needs to exist for the process to run). The downfall is that "scan" plugins are a bit slower than "list" plugins, and can sometimes yield false-positives (a process that exited too long ago and had parts of its structure overwritten by other operations)."*
>
> http://tomchop.me/2016/11/21/tutorial-volatility-plugins-malware-analysis/

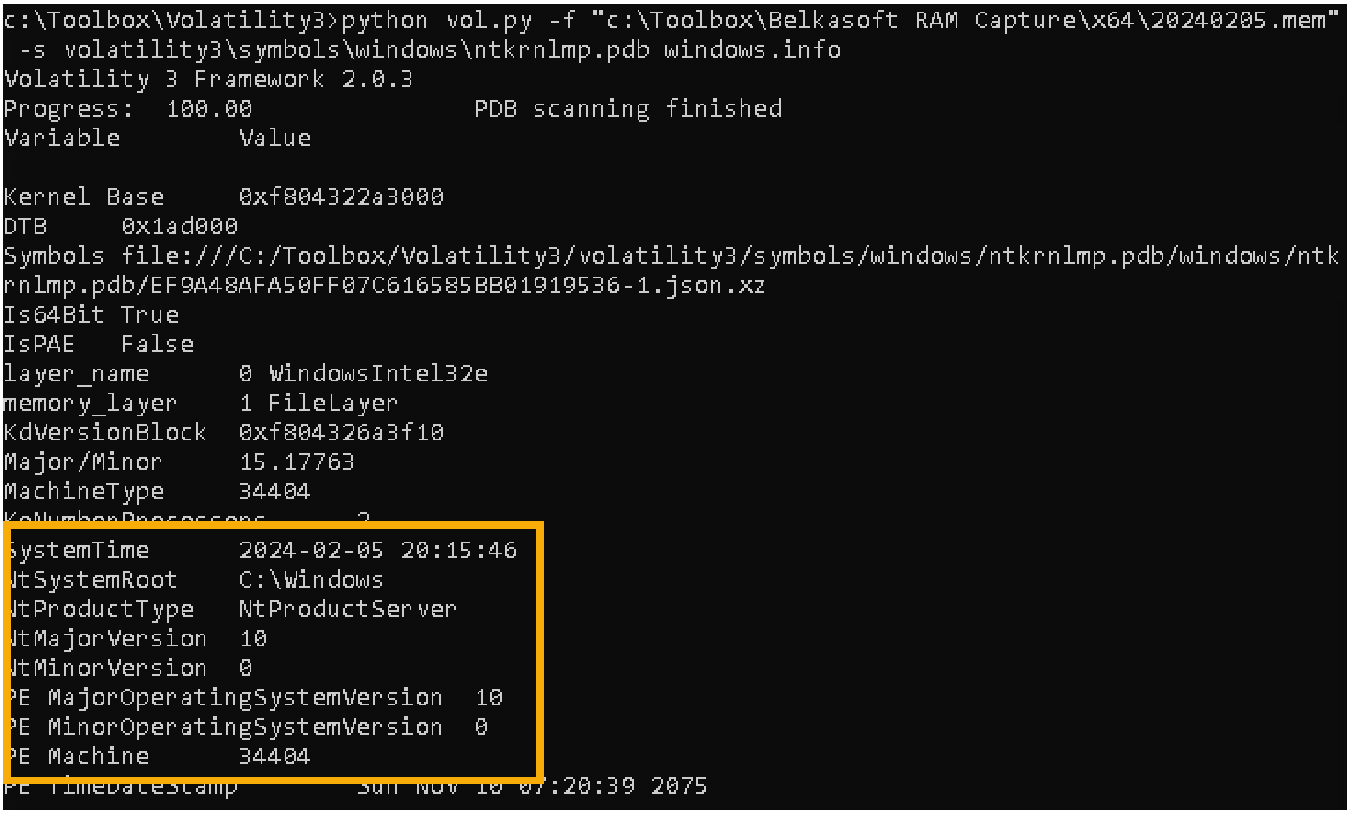
The output from *Volatility* scans can be saved to a text file which can then be read more easily than trying to read it through the command prompt window.

2. The first plugin module that should be executed is **windows.info** (there are also “info” plugins for Mac and Linux). This module shows a summary of the memory image. From the **C:\ToolBox\Volatility3** directory, type the following command (the module name is case-sensitive):

```
python vol.py –f “c:\Toolbox\Belkasoft RAM Capture\x64\<dumpdate>.mem”
–s volatility3\symbols\windows\ntkrnlmp.pdb windows.info
```

The ***<dumpdate>*** will be the date the dump file was made. Yours will be different than the above example.

Your output should look similar to the picture below. Some interesting information to look for is the date and time the image was created and the operating system version.



```
c:\Toolbox\Volatility3>python vol.py -f "c:\Toolbox\Belkasoft RAM Capture\x64\20240205.mem"
 -s volatility3\symbols\windows\ntkrnlmp.pdb windows.info
Volatility 3 Framework 2.0.3
Progress:  100.00               PDB scanning finished
Variable        Value

Kernel Base     0xf804322a3000
DTB     0x1ad000
Symbols file:///C:/Toolbox/Volatility3/volatility3/symbols/windows/ntkrnlmp.pdb/windows/ntk
rnlmp.pdb/EF9A48AFA50FF07C616585BB01919536-1.json.xz
Is64Bit True
IsPAE   False
layer_name      0 WindowsIntel32e
memory_layer    1 FileLayer
KdVersionBlock  0xf804326a3f10
Major/Minor     15.17763
MachineType     34404
KeNumberProcessors      2
SystemTime      2024-02-05 20:15:46
NtSystemRoot    C:\Windows
NtProductType   NtProductServer
NtMajorVersion  10
NtMinorVersion  0
PE MajorOperatingSystemVersion  10
PE MinorOperatingSystemVersion  0
PE Machine      34404
PE TimeDateStamp        Sun Nov 10 07:20:39 2075
```

There are three core plugins that are used to gather basic information:

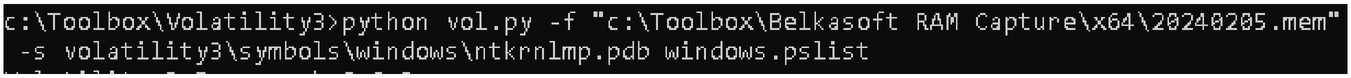
pslist	Displays a list of running processes
pstree	Displays the same output as pslist, but displays it using a Process Tree to see child and parent process relationships
netscan	Displays a list of open network connections (much like nmap)

You can save some typing by using the up-arrow key to repeat the previous command and just change the name of the plugin.

3. From the command prompt, type the following command to scan the dump file and display the list of running processes:

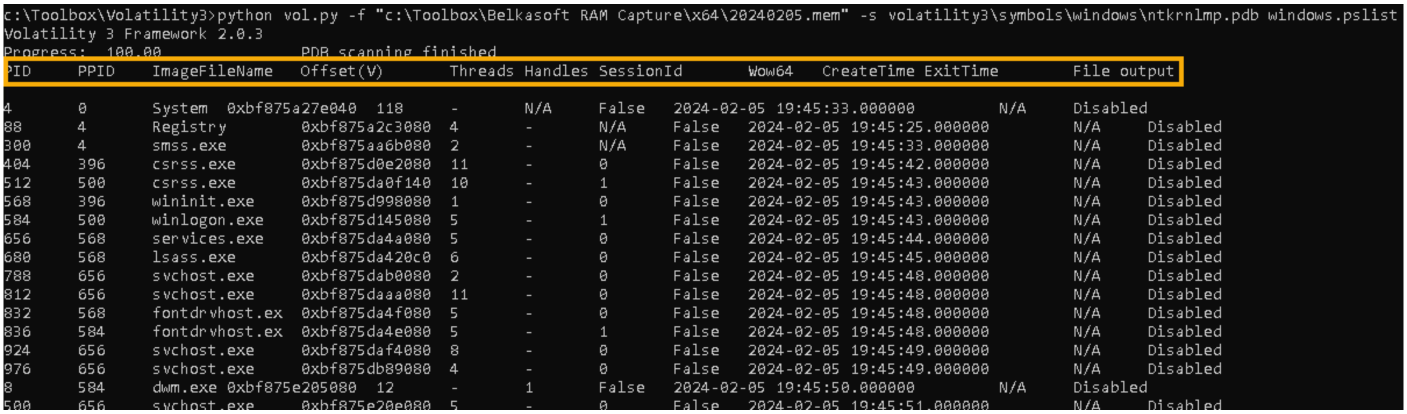
```
python vol.py –f “c:\Toolbox\Belkasoft RAM Capture\x64\<dumpdate>.mem”
–s volatility3\symbols\windows\ntkrnlmp.pdb windows.pslist
```

The module name, in this case, is **windows.pslist**, which is case-sensitive.



```
c:\Toolbox\Volatility3>python vol.py -f "c:\Toolbox\Belkasoft RAM Capture\x64\20240205.mem"
 -s volatility3\symbols\windows\ntkrnlmp.pdb windows.pslist
```

4. Scroll up to the start of the command; your output should look something like this:



```
c:\Toolbox\Volatility3>python vol.py -f "c:\Toolbox\Belkasoft RAM Capture\x64\20240205.mem" -s volatility3\symbols\windows\ntkrnlmp.pdb windows.pslist
Volatility 3 Framework 2.0.3
Progress:  100.00               PDB scanning finished
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime ExitTime         File output

4       0       System  0xbf875a27e040  118     -       N/A     False   2024-02-05 19:45:33.000000      N/A     Disabled
88      4       Registry        0xbf875a2c3080  4       -       N/A     False   2024-02-05 19:45:25.000000      N/A     Disabled
300     4       smss.exe        0xbf875aa6b080  2       -       N/A     False   2024-02-05 19:45:33.000000      N/A     Disabled
404     396     csrss.exe       0xbf875d0e2080  11      -       0       False   2024-02-05 19:45:42.000000      N/A     Disabled
512     500     csrss.exe       0xbf875da0f140  10      -       1       False   2024-02-05 19:45:43.000000      N/A     Disabled
568     396     wininit.exe     0xbf875d998080  1       -       0       False   2024-02-05 19:45:43.000000      N/A     Disabled
584     500     winlogon.exe    0xbf875d145080  5       -       1       False   2024-02-05 19:45:43.000000      N/A     Disabled
656     568     services.exe    0xbf875da4a080  5       -       0       False   2024-02-05 19:45:44.000000      N/A     Disabled
680     568     lsass.exe       0xbf875da420c0  6       -       0       False   2024-02-05 19:45:45.000000      N/A     Disabled
788     656     svchost.exe     0xbf875dab0080  2       -       0       False   2024-02-05 19:45:48.000000      N/A     Disabled
812     656     svchost.exe     0xbf875daaa080  11      -       0       False   2024-02-05 19:45:48.000000      N/A     Disabled
832     568     fontdrvhost.ex  0xbf875da4f080  5       -       0       False   2024-02-05 19:45:48.000000      N/A     Disabled
836     584     fontdrvhost.ex  0xbf875da4e080  5       -       1       False   2024-02-05 19:45:48.000000      N/A     Disabled
924     656     svchost.exe     0xbf875daf4080  8       -       0       False   2024-02-05 19:45:49.000000      N/A     Disabled
976     656     svchost.exe     0xbf875db89080  4       -       0       False   2024-02-05 19:45:49.000000      N/A     Disabled
8       584     dwm.exe 0xbf875e205080  12      -       1       False   2024-02-05 19:45:50.000000      N/A     Disabled
500     656     svchost.exe     0xbf875e20e080  5       -       0       False   2024-02-05 19:45:51.000000      N/A     Disabled
```

There are several columns in the output that are important to observe:

- **ImageFileName**: The name of the service or program that was activated. In the example above, the first process is System, and the second is Registry.

- **PID** (Process ID): A unique number used by the kernel of the operating system to identify an active process. In the above example, the System process has the PID of 4, and the Registry has the PID of 88.

- **PPID** (Parent Process ID): When a process is activated, the PID of the parent is listed here. In the above example, the System process does not have a PPID since it is the first process that was loaded. The Registry process was activated by the System process, so its PPID is 4.

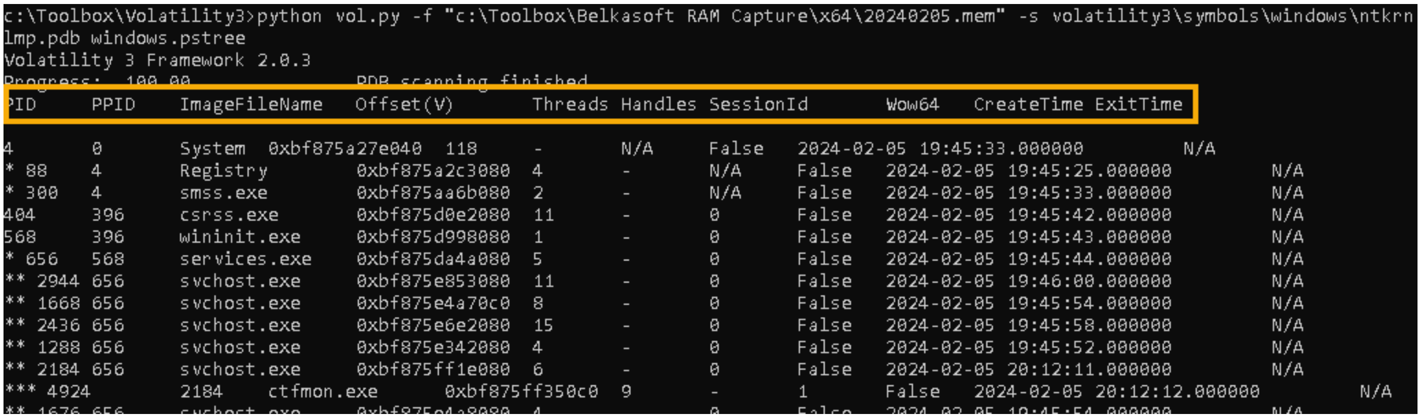
- **Offset(V):** The logical address within a memory segment showing where the process is loaded in memory.

- **Threads**: Units of execution within the process and processes can have more than one thread. Each thread can execute parts of the program (almost) simultaneously, resulting in better utilization of processing resources.

- **Handles**: Objects in memory that point, or link, to another object when that object (a file, for example) cannot be represented.

5. This time, display the process list in *Process Tree* output using the following command:

```
python vol.py –f “c:\Toolbox\Belkasoft RAM Capture\x64\<dumpdate>.mem”
-s volatility3\symbols\windows\ntkrnlmp.pdb windows.pstree
```



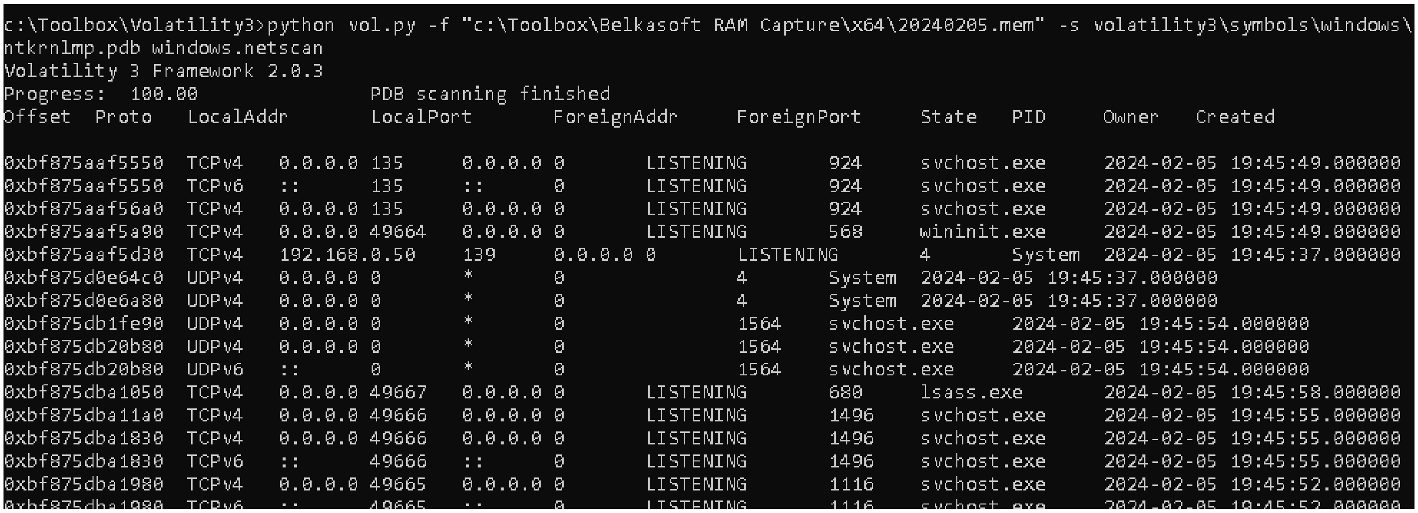
```
c:\Toolbox\Volatility3>python vol.py -f "c:\Toolbox\Belkasoft RAM Capture\x64\20240205.mem" -s volatility3\symbols\windows\ntkrn
lmp.pdb windows.pstree
Volatility 3 Framework 2.0.3
Progress:  100.00               PDB scanning finished
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime ExitTime

4       0       System  0xbf875a27e040  118     -       N/A     False   2024-02-05 19:45:33.000000        N/A
* 88    4       Registry        0xbf875a2c3080  4       -       N/A     False   2024-02-05 19:45:25.000000        N/A
* 300   4       smss.exe        0xbf875aa6b080  2       -       N/A     False   2024-02-05 19:45:33.000000        N/A
404     396     csrss.exe       0xbf875d0e2080  11      -       0       False   2024-02-05 19:45:42.000000        N/A
568     396     wininit.exe     0xbf875d998080  1       -       0       False   2024-02-05 19:45:43.000000        N/A
* 656   568     services.exe    0xbf875da4a080  5       -       0       False   2024-02-05 19:45:44.000000        N/A
** 2944 656     svchost.exe     0xbf875e853080  11      -       0       False   2024-02-05 19:46:00.000000        N/A
** 1668 656     svchost.exe     0xbf875e4a70c0  8       -       0       False   2024-02-05 19:45:54.000000        N/A
** 2436 656     svchost.exe     0xbf875e6e2080  15      -       0       False   2024-02-05 19:45:58.000000        N/A
** 1288 656     svchost.exe     0xbf875e342080  4       -       0       False   2024-02-05 19:45:52.000000        N/A
** 2184 656     svchost.exe     0xbf875ff1e080  6       -       0       False   2024-02-05 20:12:11.000000        N/A
*** 4924        2184    ctfmon.exe      0xbf875ff350c0  9       -       1       False   2024-02-05 20:12:12.000000        N/A
** 1676 656     svchost.exe     0xbf875e4a8080  4       -       0       False   2024-02-05 19:45:54.000000        N/A
```

In the above example, scrolling up to the top of the command, you should see that the *System* process, **PID 4**, is a “root” process, and has no parent, and the *Registry* process, **PID 88,** and the *SMSS.EXE* process, **PID 300** were both launched by the *System* process. Similarly, you can track the relationships between parent processes and child-launched processes.

6. Now, let’s display the list of open network connections using the *windows.netscan.NetScan* module:

```
python vol.py –f “c:\Toolbox\Belkasoft RAM Capture\x64\<dumpdate>.mem”
–s volatility3\symbols\windows\ntkrnlmp.pdb windows.netscan
```

The output from this module is similar to the output from *nmap*.



```
c:\Toolbox\Volatility3>python vol.py -f "c:\Toolbox\Belkasoft RAM Capture\x64\20240205.mem" -s volatility3\symbols\windows\
ntkrnlmp.pdb windows.netscan
Volatility 3 Framework 2.0.3
Progress:  100.00               PDB scanning finished
Offset  Proto   LocalAddr       LocalPort       ForeignAddr     ForeignPort     State   PID     Owner   Created

0xbf875aaf5550  TCPv4   0.0.0.0 135     0.0.0.0 0       LISTENING       924     svchost.exe     2024-02-05 19:45:49.000000
0xbf875aaf5550  TCPv6   ::      135     ::      0       LISTENING       924     svchost.exe     2024-02-05 19:45:49.000000
0xbf875aaf56a0  TCPv4   0.0.0.0 135     0.0.0.0 0       LISTENING       924     svchost.exe     2024-02-05 19:45:49.000000
0xbf875aaf5a90  TCPv4   0.0.0.0 49664   0.0.0.0 0       LISTENING       568     wininit.exe     2024-02-05 19:45:49.000000
0xbf875aaf5d30  TCPv4   192.168.0.50    139     0.0.0.0 0       LISTENING       4       System  2024-02-05 19:45:37.000000
0xbf875d0e64c0  UDPv4   0.0.0.0 0       *       0               4       System  2024-02-05 19:45:37.000000
0xbf875d0e6a80  UDPv4   0.0.0.0 0       *       0               4       System  2024-02-05 19:45:37.000000
0xbf875db1fe90  UDPv4   0.0.0.0 0       *       0               1564    svchost.exe     2024-02-05 19:45:54.000000
0xbf875db20b80  UDPv4   0.0.0.0 0       *       0               1564    svchost.exe     2024-02-05 19:45:54.000000
0xbf875db20b80  UDPv6   ::      0       *       0               1564    svchost.exe     2024-02-05 19:45:54.000000
0xbf875dba1050  TCPv4   0.0.0.0 49667   0.0.0.0 0       LISTENING       680     lsass.exe       2024-02-05 19:45:58.000000
0xbf875dba11a0  TCPv4   0.0.0.0 49666   0.0.0.0 0       LISTENING       1496    svchost.exe     2024-02-05 19:45:55.000000
0xbf875dba1830  TCPv4   0.0.0.0 49666   0.0.0.0 0       LISTENING       1496    svchost.exe     2024-02-05 19:45:55.000000
0xbf875dba1830  TCPv6   ::      49666   ::      0       LISTENING       1496    svchost.exe     2024-02-05 19:45:55.000000
0xbf875dba1980  TCPv4   0.0.0.0 49665   0.0.0.0 0       LISTENING       1116    svchost.exe     2024-02-05 19:45:52.000000
0xbf875dba1980  TCPv6   ::      49665   ::      0       LISTENING       1116    svchost.exe     2024-02-05 19:45:52.000000
```

It is beyond the scope of this lab to go through all of the *Volatility* plugin modules. There are many good books, papers, and websites that can be consulted for specific plugin usage.

7. Remain on the *WinOS* computer, with the command prompt open, and continue to the next task.

3 Analyzing a Memory Image Containing Malware

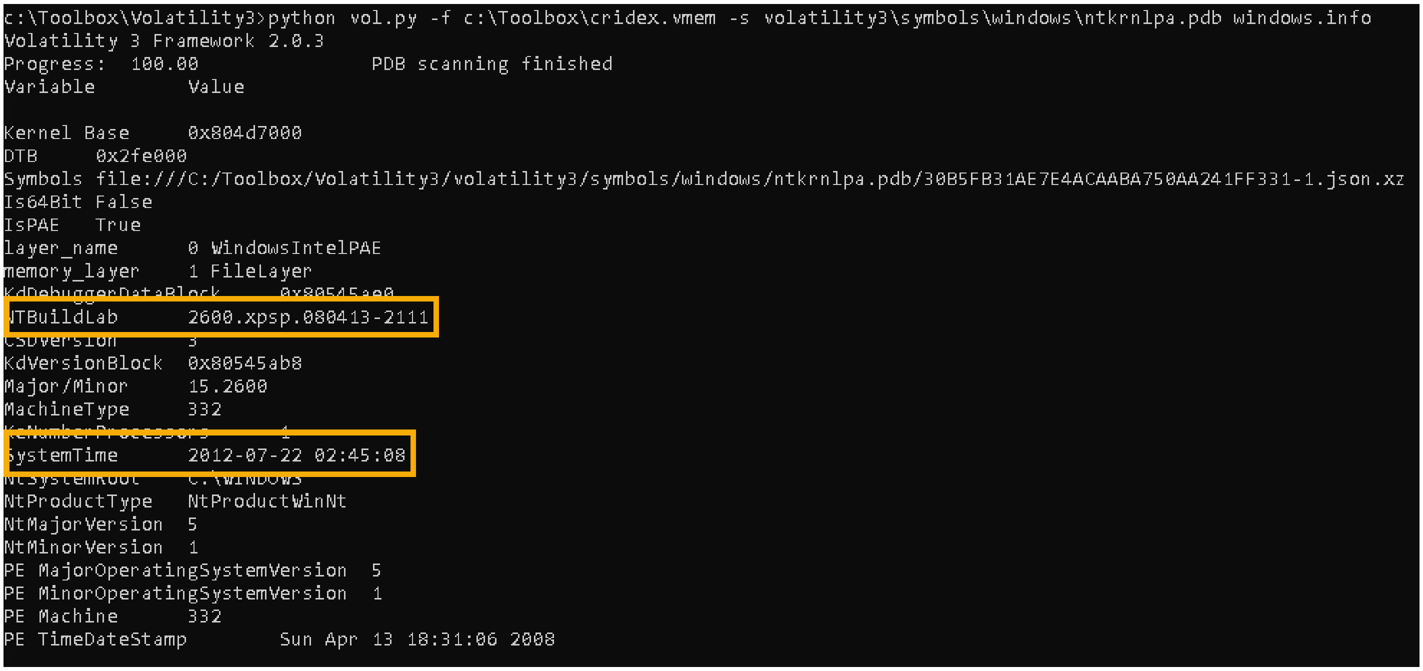
Now that you have gone through the memory analysis process using *Volatility,* let's go through a memory analysis where the memory image file contains malware, in this case, Cridex.

> *"Cridex malware, also known as Cridex or W32.Cridex, is a malicious computer worm that spread to computers by copying itself to removable disks. On each computer it infects, it opens a backdoor and downloads malicious software to the hard disk. The malicious software gathers personal information on the compromised machine, including web session and banking data, and transmits it to a third-party.*
>
> *Cridex-infected machines can also become botnet slaves, participating in behavior such as DDoS attacks."*
>
> https://www.computerhope.com/jargon/c/cridex-malware.htm/

1. On the command prompt window, in the **C:\Toolbox\Volatility3** folder, type the following command to get information about the memory dump.

```
python vol.py -f c:\Toolbox\cridex.vmem
-s volatility3\symbols\windows\ntkrnlpa.pdb windows.info
```

   Looking at the output, you can see the image was created back in 2012, and it was extracted from a Windows XP computer.

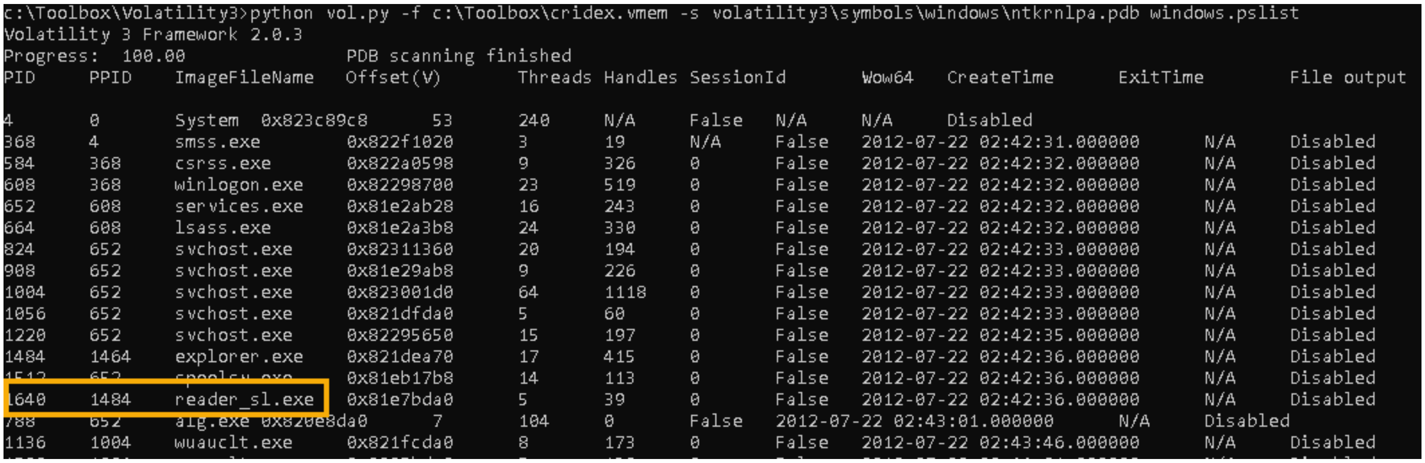


```
c:\Toolbox\Volatility3>python vol.py -f c:\Toolbox\cridex.vmem -s volatility3\symbols\windows\ntkrnlpa.pdb windows.info
Volatility 3 Framework 2.0.3
Progress:  100.00               PDB scanning finished
Variable        Value

Kernel Base     0x804d7000
DTB     0x2fe000
Symbols file:///C:/Toolbox/Volatility3/volatility3/symbols/windows/ntkrnlpa.pdb/30B5FB31AE7E4ACAABA750AA241FF331-1.json.xz
Is64Bit False
IsPAE   True
layer_name      0 WindowsIntelPAE
memory_layer    1 FileLayer
KdDebuggerDataBlock     0x80545ae0
NTBuildLab      2600.xpsp.080413-2111
CSDVersion      3
KdVersionBlock  0x80545ab8
Major/Minor     15.2600
MachineType     332
KeNumberProcessors      1
SystemTime      2012-07-22 02:45:08
NtSystemRoot    C:\WINDOWS
NtProductType   NtProductWinNt
NtMajorVersion  5
NtMinorVersion  1
PE MajorOperatingSystemVersion  5
PE MinorOperatingSystemVersion  1
PE Machine      332
PE TimeDateStamp        Sun Apr 13 18:31:06 2008
```

2. Type the following command to see the running processes:

```
python vol.py –f \Toolbox\cridex.vmem
-s volatility3\symbols\windows\ntkrnlpa.pdb windows.pslist
```

Notice the process, *reader_sl.exe* in the list. It has the PID of 1640, and the parent process (PPID) is 1484, which is explorer.exe. This file is a legitimate file from Adobe Acrobat, also known as Acrobat Speed Launcher. So, at first glance, you might not suspect that it is malware, but, you know that Acrobat has NOT been installed on the computer where this memory image was captured. A bit of research informs you that hackers create files with malicious content, in this case, *Cridex*, and name the file *reader_sl.exe* to spread the virus.

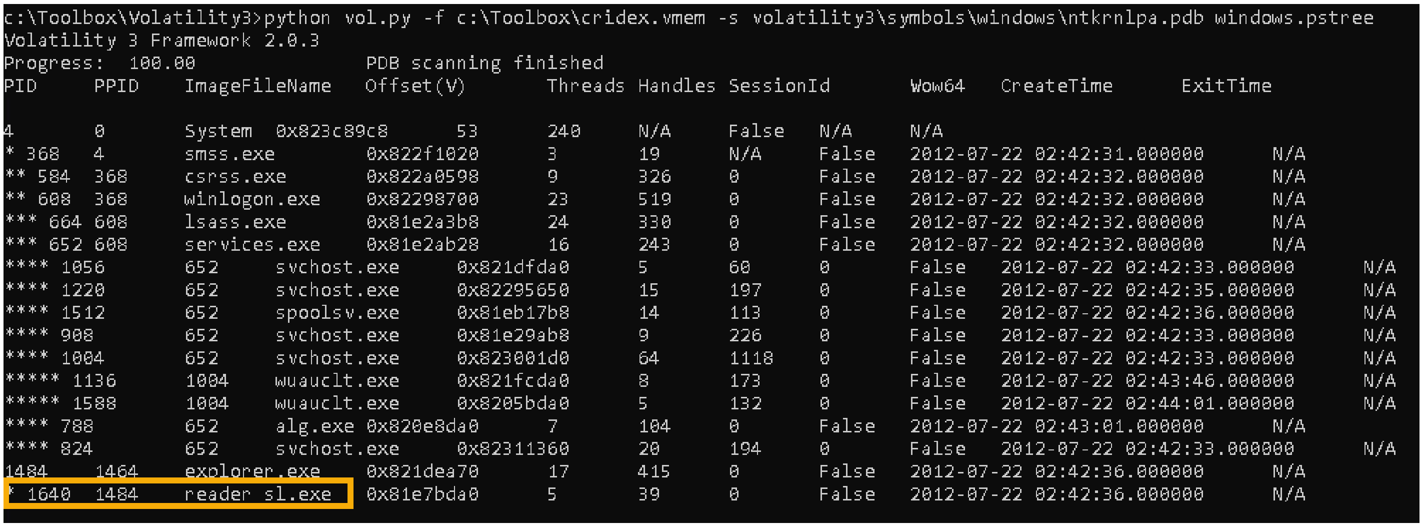


```
c:\Toolbox\Volatility3>python vol.py -f c:\Toolbox\cridex.vmem -s volatility3\symbols\windows\ntkrnlpa.pdb windows.pslist
Volatility 3 Framework 2.0.3
Progress:  100.00               PDB scanning finished
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime      ExitTime        File output

4       0       System  0x823c89c8      53      240     N/A     False   N/A     N/A     Disabled
368     4       smss.exe        0x822f1020      3       19      N/A     False   2012-07-22 02:42:31.000000      N/A     Disabled
584     368     csrss.exe       0x822a0598      9       326     0       False   2012-07-22 02:42:32.000000      N/A     Disabled
608     368     winlogon.exe    0x82298700      23      519     0       False   2012-07-22 02:42:32.000000      N/A     Disabled
652     608     services.exe    0x81e2ab28      16      243     0       False   2012-07-22 02:42:32.000000      N/A     Disabled
664     608     lsass.exe       0x81e2a3b8      24      330     0       False   2012-07-22 02:42:32.000000      N/A     Disabled
824     652     svchost.exe     0x82311360      20      194     0       False   2012-07-22 02:42:33.000000      N/A     Disabled
908     652     svchost.exe     0x81e29ab8      9       226     0       False   2012-07-22 02:42:33.000000      N/A     Disabled
1004    652     svchost.exe     0x823001d0      64      1118    0       False   2012-07-22 02:42:33.000000      N/A     Disabled
1056    652     svchost.exe     0x821dfda0      5       60      0       False   2012-07-22 02:42:33.000000      N/A     Disabled
1220    652     svchost.exe     0x82295650      15      197     0       False   2012-07-22 02:42:35.000000      N/A     Disabled
1484    1464    explorer.exe    0x821dea70      17      415     0       False   2012-07-22 02:42:36.000000      N/A     Disabled
1512    652     spoolsv.exe     0x81eb17b8      14      113     0       False   2012-07-22 02:42:36.000000      N/A     Disabled
1640    1484    reader_sl.exe   0x81e7bda0      5       39      0       False   2012-07-22 02:42:36.000000      N/A     Disabled
788     652     alg.exe 0x820e8da0      7       104     0       False   2012-07-22 02:43:01.000000      N/A     Disabled
1136    1004    wuauclt.exe     0x821fcda0      8       173     0       False   2012-07-22 02:43:46.000000      N/A     Disabled
```

3. Type the following command to see the process list in tree format.

```
python vol.py –f \Toolbox\cridex.vmem
-s volatility3\symbols\windows\ntkrnlpa.pdb windows.pstree
```

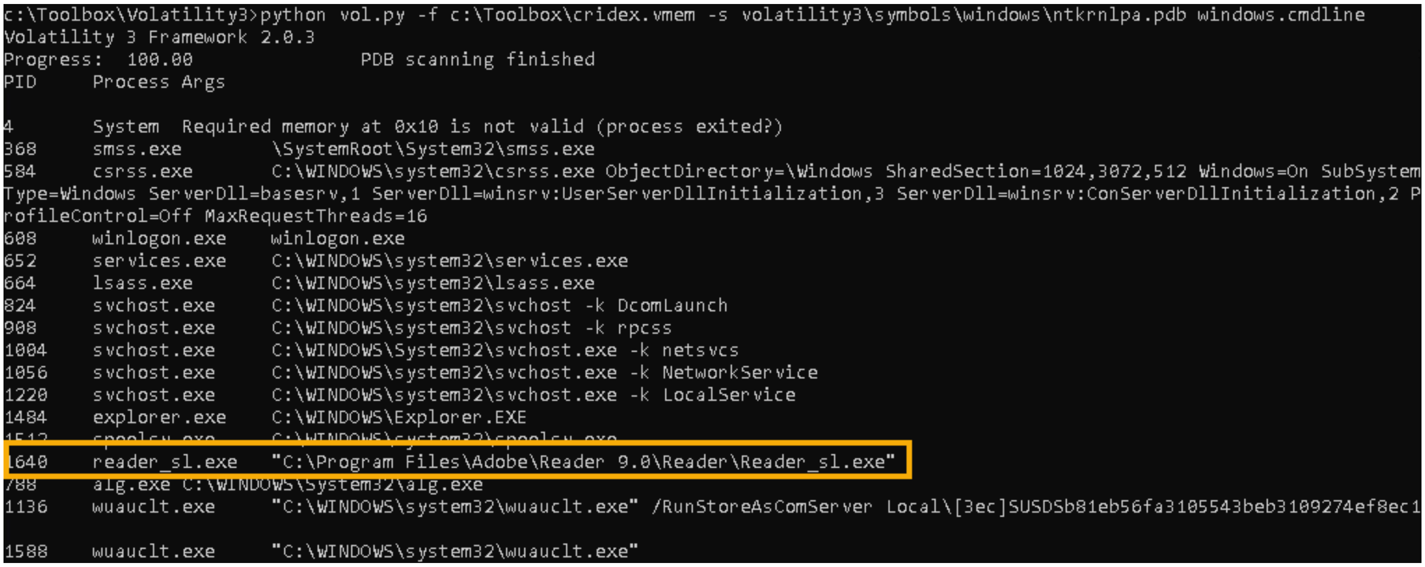
By examining the output, it can be seen that the *reader_sl.exe* process was one of the last processes to be started.



```
c:\Toolbox\Volatility3>python vol.py -f c:\Toolbox\cridex.vmem -s volatility3\symbols\windows\ntkrnlpa.pdb windows.pstree
Volatility 3 Framework 2.0.3
Progress:  100.00               PDB scanning finished
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime      ExitTime

4       0       System  0x823c89c8      53      240     N/A     False   N/A     N/A
* 368   4       smss.exe        0x822f1020      3       19      N/A     False   2012-07-22 02:42:31.000000      N/A
** 584  368     csrss.exe       0x822a0598      9       326     0       False   2012-07-22 02:42:32.000000      N/A
** 608  368     winlogon.exe    0x82298700      23      519     0       False   2012-07-22 02:42:32.000000      N/A
*** 664 608     lsass.exe       0x81e2a3b8      24      330     0       False   2012-07-22 02:42:32.000000      N/A
*** 652 608     services.exe    0x81e2ab28      16      243     0       False   2012-07-22 02:42:32.000000      N/A
**** 1056       652     svchost.exe     0x821dfda0      5       60      0       False   2012-07-22 02:42:33.000000      N/A
**** 1220       652     svchost.exe     0x82295650      15      197     0       False   2012-07-22 02:42:35.000000      N/A
**** 1512       652     spoolsv.exe     0x81eb17b8      14      113     0       False   2012-07-22 02:42:36.000000      N/A
**** 908        652     svchost.exe     0x81e29ab8      9       226     0       False   2012-07-22 02:42:33.000000      N/A
**** 1004       652     svchost.exe     0x823001d0      64      1118    0       False   2012-07-22 02:42:33.000000      N/A
***** 1136      1004    wuauclt.exe     0x821fcda0      8       173     0       False   2012-07-22 02:43:46.000000      N/A
***** 1588      1004    wuauclt.exe     0x8205bda0      5       132     0       False   2012-07-22 02:44:01.000000      N/A
**** 788        652     alg.exe 0x820e8da0      7       104     0       False   2012-07-22 02:43:01.000000      N/A
**** 824        652     svchost.exe     0x82311360      20      194     0       False   2012-07-22 02:42:33.000000      N/A
1484    1464    explorer.exe    0x821dea70      17      415     0       False   2012-07-22 02:42:36.000000      N/A
* 1640  1484    reader_sl.exe   0x81e7bda0      5       39      0       False   2012-07-22 02:42:36.000000      N/A
```

4. Type the following command to list the full path of the running processes:

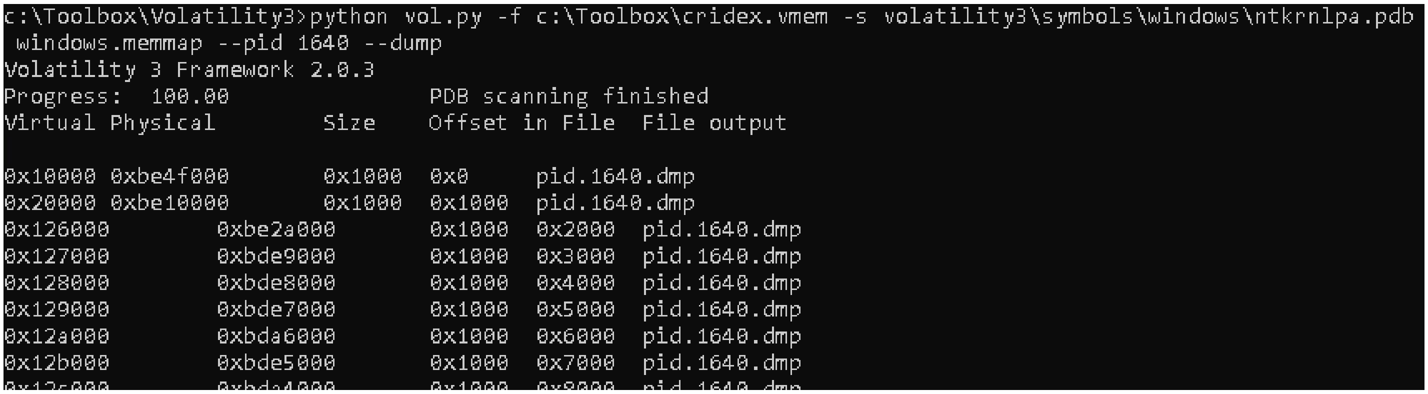
```
python vol.py –f \Toolbox\cridex.vmem
-s volatility3\symbols\windows\ntkrnlpa.pdb windows.cmdline
```



```
c:\Toolbox\Volatility3>python vol.py -f c:\Toolbox\cridex.vmem -s volatility3\symbols\windows\ntkrnlpa.pdb windows.cmdline
Volatility 3 Framework 2.0.3
Progress:  100.00               PDB scanning finished
PID     Process Args

4       System  Required memory at 0x10 is not valid (process exited?)
368     smss.exe        \SystemRoot\System32\smss.exe
584     csrss.exe       C:\WINDOWS\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,3072,512 Windows=On SubSystem
Type=Windows ServerDll=basesrv,1 ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:ConServerDllInitialization,2 P
rofileControl=Off MaxRequestThreads=16
608     winlogon.exe    winlogon.exe
652     services.exe    C:\WINDOWS\system32\services.exe
664     lsass.exe       C:\WINDOWS\system32\lsass.exe
824     svchost.exe     C:\WINDOWS\system32\svchost -k DcomLaunch
908     svchost.exe     C:\WINDOWS\system32\svchost -k rpcss
1004    svchost.exe     C:\WINDOWS\System32\svchost.exe -k netsvcs
1056    svchost.exe     C:\WINDOWS\system32\svchost.exe -k NetworkService
1220    svchost.exe     C:\WINDOWS\system32\svchost.exe -k LocalService
1484    explorer.exe    C:\WINDOWS\Explorer.EXE
1512    spoolsv.exe     C:\WINDOWS\system32\spoolsv.exe
1640    reader_sl.exe   "C:\Program Files\Adobe\Reader 9.0\Reader\Reader_sl.exe"
788     alg.exe C:\WINDOWS\System32\alg.exe
1136    wuauclt.exe     "C:\WINDOWS\system32\wuauclt.exe" /RunStoreAsComServer Local\[3ec]SUSDSb81eb56fa3105543beb3109274ef8ec1

1588    wuauclt.exe     "C:\WINDOWS\system32\wuauclt.exe"
```

5. You can use the *Volatility* plugin **windows.memmap** to create an addressable memory dump file for the **reader_sl.exe** file using the **PID** of **1640**, by typing the following command:

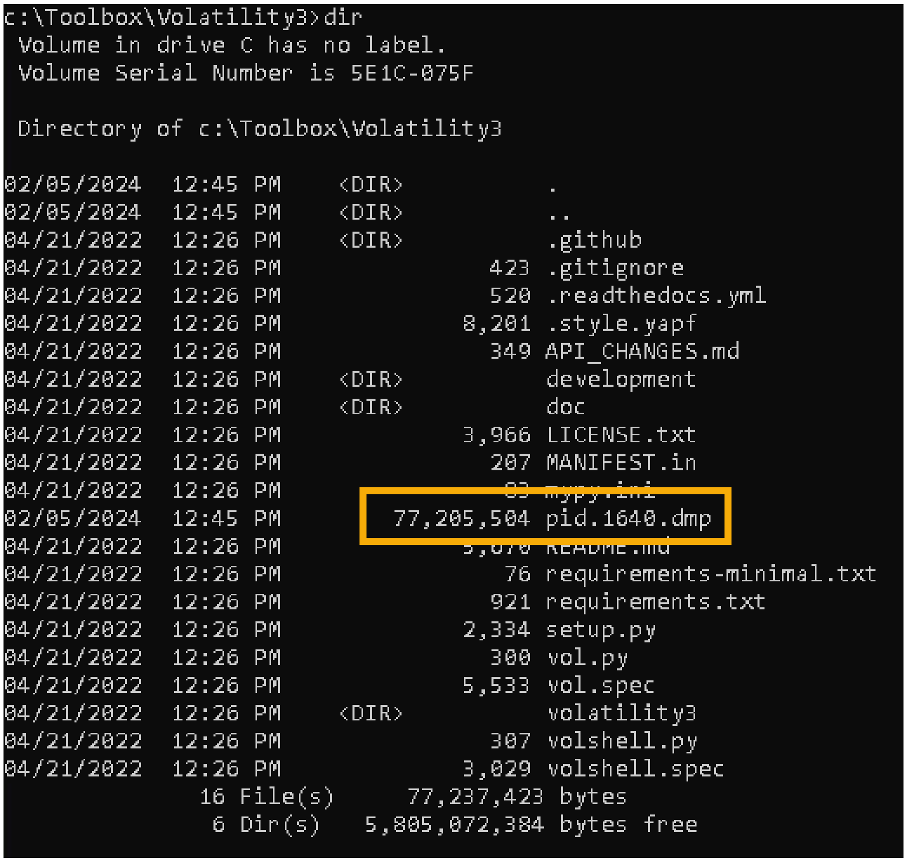
```
python vol.py –f \Toolbox\cridex.vmem
-s volatility3\symbols\windows\ntkrnlpa.pdb windows.memmap
--pid 1640 --dump
```



```
c:\Toolbox\Volatility3>python vol.py -f c:\Toolbox\cridex.vmem -s volatility3\symbols\windows\ntkrnlpa.pdb
 windows.memmap --pid 1640 --dump
Volatility 3 Framework 2.0.3
Progress:  100.00               PDB scanning finished
Virtual Physical        Size    Offset in File  File output

0x10000 0xbe4f000       0x1000  0x0     pid.1640.dmp
0x20000 0xbe10000       0x1000  0x1000  pid.1640.dmp
0x126000        0xbe2a000       0x1000  0x2000  pid.1640.dmp
0x127000        0xbde9000       0x1000  0x3000  pid.1640.dmp
0x128000        0xbde8000       0x1000  0x4000  pid.1640.dmp
0x129000        0xbde7000       0x1000  0x5000  pid.1640.dmp
0x12a000        0xbda6000       0x1000  0x6000  pid.1640.dmp
0x12b000        0xbde5000       0x1000  0x7000  pid.1640.dmp
0x12c000        0xbda4000       0x1000  0x8000  pid.1640.dmp
```

6. Type the **dir** command to list the contents of the C:\Toolbox\Volatility folder:

```
dir
```

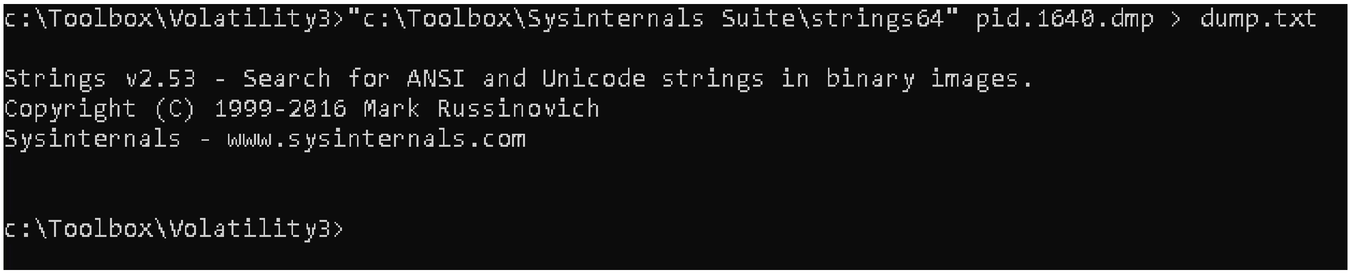


```
c:\Toolbox\Volatility3>dir
 Volume in drive C has no label.
 Volume Serial Number is 5E1C-075F

 Directory of c:\Toolbox\Volatility3

02/05/2024  12:45 PM    <DIR>          .
02/05/2024  12:45 PM    <DIR>          ..
04/21/2022  12:26 PM    <DIR>          .github
04/21/2022  12:26 PM               423 .gitignore
04/21/2022  12:26 PM               520 .readthedocs.yml
04/21/2022  12:26 PM             8,201 .style.yapf
04/21/2022  12:26 PM               349 API_CHANGES.md
04/21/2022  12:26 PM    <DIR>          development
04/21/2022  12:26 PM    <DIR>          doc
04/21/2022  12:26 PM             3,966 LICENSE.txt
04/21/2022  12:26 PM               207 MANIFEST.in
04/21/2022  12:26 PM                83 mypy.ini
02/05/2024  12:45 PM        77,205,504 pid.1640.dmp
04/21/2022  12:26 PM             5,070 README.md
04/21/2022  12:26 PM                76 requirements-minimal.txt
04/21/2022  12:26 PM               921 requirements.txt
04/21/2022  12:26 PM             2,334 setup.py
04/21/2022  12:26 PM               300 vol.py
04/21/2022  12:26 PM             5,533 vol.spec
04/21/2022  12:26 PM    <DIR>          volatility3
04/21/2022  12:26 PM               307 volshell.py
04/21/2022  12:26 PM             3,029 volshell.spec
              16 File(s)     77,237,423 bytes
               6 Dir(s)   5,805,072,384 bytes free
```

You should see the file **pid.1640.dmp** file.

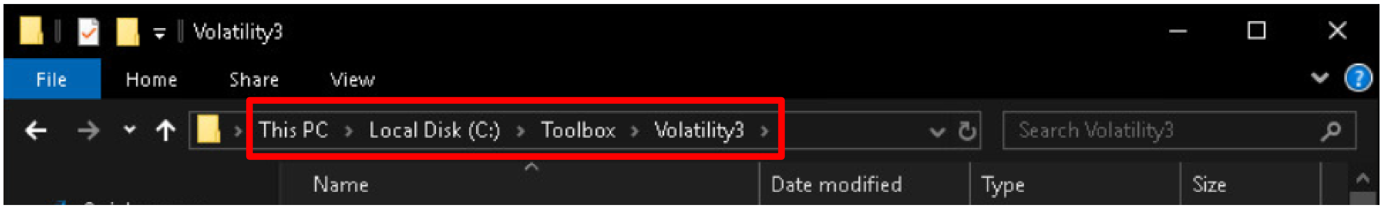
7. You can use the *Sysinternals* **strings64** command to analyze the file looking for a reference to a **Host**. This can indicate a reference to an external, public IP address. Type the following command:

```
“c:\Toolbox\Sysinternals Suite\stings64” pid.1640.dmp > dump.txt
```

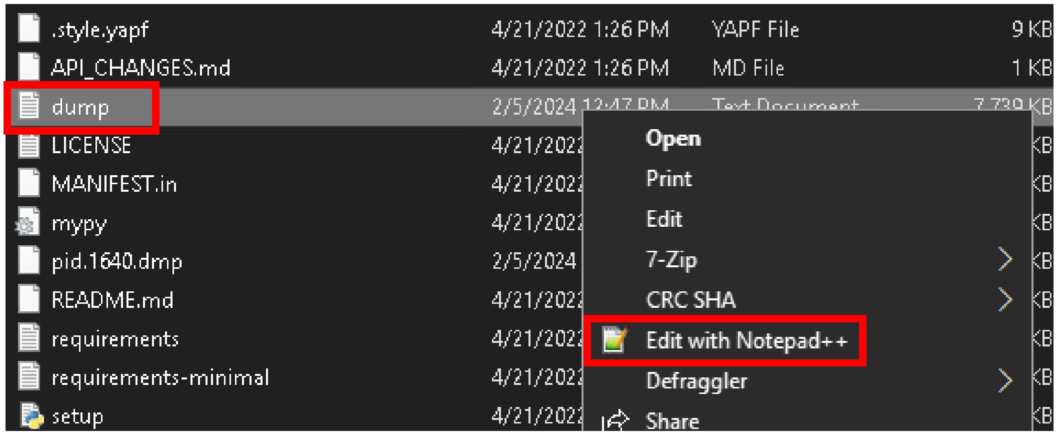


```
c:\Toolbox\Volatility3>"c:\Toolbox\Sysinternals Suite\strings64" pid.1640.dmp > dump.txt

Strings v2.53 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2016 Mark Russinovich
Sysinternals - www.sysinternals.com


c:\Toolbox\Volatility3>
```

8. Restore the **File Explorer** window, then change the directory to: **This PC > Local Disk (C:) > Toolbox > Volatility3**.

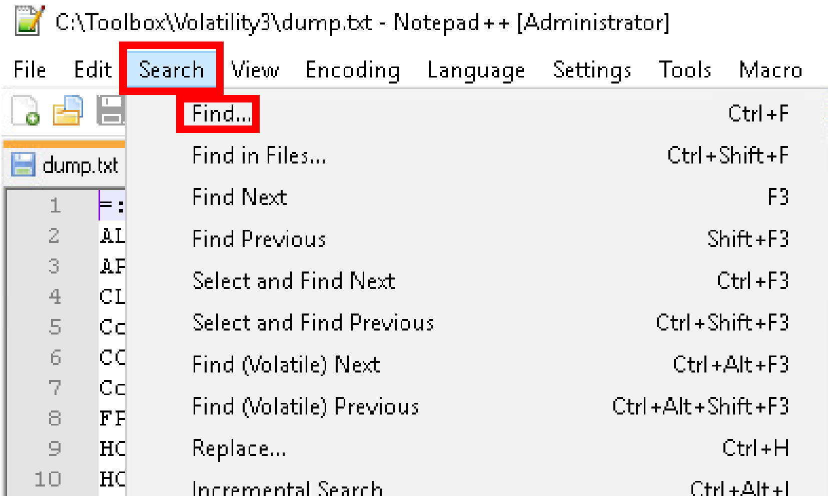


9. In the list, find the **dump** text file, right-click and select **Edit with Notepad++.**

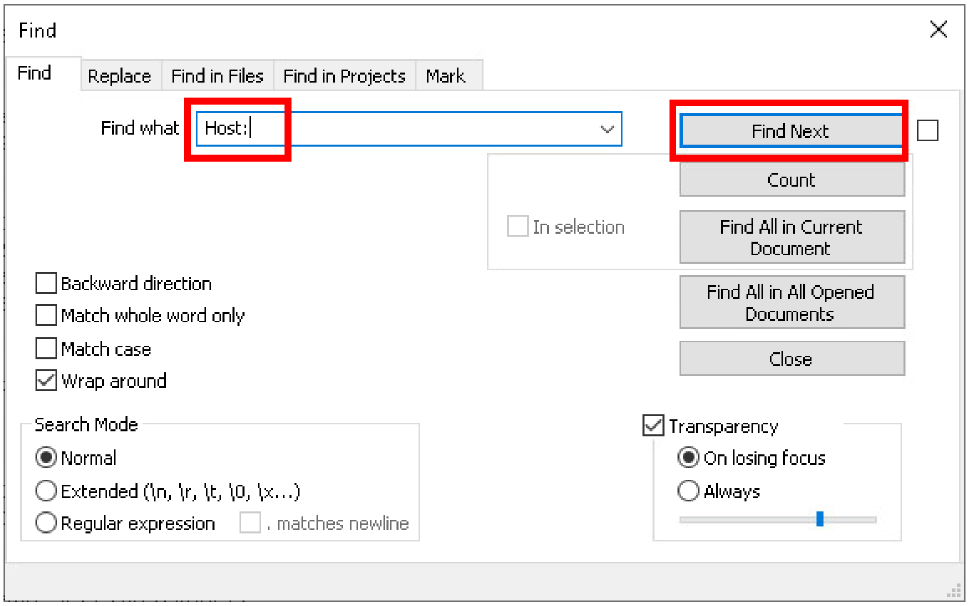


.style.yapf	4/21/2022 1:26 PM	YAPF File	9 KB
API_CHANGES.md	4/21/2022 1:26 PM	MD File	1 KB
dump	2/5/2024 12:47 PM	Text Document	7,739 KB
LICENSE	4/21/202		KB
MANIFEST.in	4/21/202		KB
mypy	4/21/202		KB
pid.1640.dmp	2/5/2024		KB
README.md	4/21/202		KB
requirements	4/21/202		KB
requirements-minimal	4/21/202		KB
setup	4/21/202		KB

10. In the *Notepad++* window, click on **Search** in the menu bar, and then click on **Find.**


C:\Toolbox\Volatility3\dump.txt - Notepad++ [Administrator]
File
Edit
Search
View
Encoding
Language
Settings
Tools
Macro
dump.txt
Find... Ctrl+F
Find in Files... Ctrl+Shift+F
Find Next F3
Find Previous Shift+F3
Select and Find Next Ctrl+F3
Select and Find Previous Ctrl+Shift+F3
Find (Volatile) Next Ctrl+Alt+F3
Find (Volatile) Previous Ctrl+Alt+Shift+F3
Replace... Ctrl+H
Incremental Search Ctrl+Alt+I

11. In the *Find* window, in the *Find what* box, type `Host:` and then click the **Find Next** button.



Find what: Host:

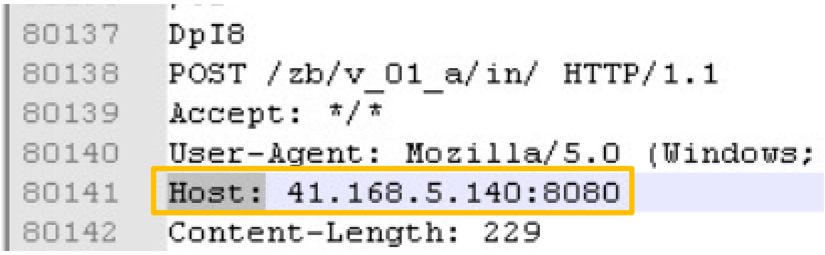
□ Backward direction
□ Match whole word only
□ Match case
☑ Wrap around

● Normal

☑ Transparency
● On losing focus

An IP address of *239.255.255.250:1900*, is a multicast address, which may be an indicator but not proof.

12. Continue looking for a **Host** with a public IP address that is not a multicast address by pressing the **Find Next** button; in this example, it is *41.168.5.140*.



```
DpI8
POST /zb/v_01_a/in/ HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (Windows;
Host: 41.168.5.140:8080
Content-Length: 229
```

In the output, you can see that the *reader_sl.exe* program is sending packets from this host computer to the destination IP address, port 8080, using a POST request.

At this point, you would use websites such as *VirusTotal* or *HybridAnalysis* to submit the file for further analysis.

13. This concludes the lab. You may now end the reservation.