Introduction

Memory forensics is the analysis of data within a computer's volatile memory. Volatile memory refers to forms of temporary memory on a computer that are power-dependent – meaning that when the computer is turned off, the contents of the volatile memory are lost. While there are forms of non-volatile memory, such as Read Only Memory (ROM), they have typically been limited to storing firmware.

For forensic investigators, volatile memory can provide valuable and unique insights into a system's activity. This can include data such as open network connections, recently executed commands, encryption keys, account credentials, and running processes. Often, critical data relating to an attack or threat will only exist in the system's memory. In fact, many network-based security solutions, such as firewalls and antivirus tools, are unable to detect forms of malware that are written directly into a computer's volatile memory. As attack methods become increasingly sophisticated, memory forensics tools and skills are in high demand for today's forensic investigators.

Forensic investigators capture volatile memory in a memory dump, which you may also see referred to as a core dump or system dump. A memory dump is a snapshot of the computer's memory at a specific moment. Because all programs, whether legitimate or malicious, must be loaded into memory to execute, memory dumps can contain valuable forensic data about the state of the system before, during, or after an incident.

In this lab, you will learn how to use different tools for creating a memory dump. You will also learn how to analyze memory dumps and identify evidence of malware.

Lab Overview

SECTION 1 of this lab has two parts, which should be completed in the order specified.

1. In the first part of the lab, you will capture a system memory dump using DumpIt.

2. In the second part of the lab, you will analyze a memory dump using Paraben's E3.

SECTION 2 of this lab allows you to apply what you learned in **SECTION 1** with less guidance and different deliverables, as well as some expanded tasks and alternative methods. You will use two alternative tools to capture and analyze system memory: FTK Imager and Volatility. You will also use the Density Scout utility to assess suspicious files and determine whether they contain viruses.

Finally, you will explore the virtual environment on your own in **SECTION 3** of this lab to answer a set of questions and challenges that allow you to use the skills you learned in the lab to conduct independent, unguided work - similar to what you will encounter in a real-world situation.

Learning Objectives

Upon completing this lab, you will be able to:

1. Create a memory dump using DumpIt.
2. Create a memory dump using FTK Imager.
3. Analyze a memory dump using E3.
4. Analyze a memory dump using Volatility.
5. Identify signs of malware in memory dumps using Density Scout.

Topology

This lab contains the following virtual machines. Please refer to the network topology diagram below.

- vWorkstation (Windows: Server 2019)

Tools and Software

The following software and/or utilities are required to complete this lab. Students are encouraged to explore the Internet to learn more about the products and tools used in this lab.

- DumpIt
- Paraben's E3
- FTK Imager
- Volatility
- DensityScout

Deliverables

Upon completion of this lab, you are required to provide the following deliverables to your instructor:

SECTION 1

1. Lab Report file, including screen captures of the following:

- DumpIt success notification
- List of processes in the memory dump

- Registry keys opened by the Hooker.exe process
- Files opened by the hooker.exe process

2. Any additional information as directed by the lab:

- Record the start times for the oldest process and the newest process.
- Document your findings for the conhost.exe process. What is it and what is it used for?
- Document your findings for the hooker.exe process. What is it and what is it used for?

SECTION 2

1. Lab Report file, including screen captures of the following:

- *Memory capture finished successfully* confirmation
- DensityScout results.

2. Any additional information as directed by the lab:

- Document your findings for the rvlkl.exe process. What is it and what is it used for?
- Document whether any processes are flagged as hidden.
- Document whether the netscan module displays network usage associated with the Hooker.exe or rvlkl.exe processes.
- Document any information you were able to gather about port 56610.

SECTION 3

1. Lab Report file, including screen captures of the following:

- FixtureComputer.exe process, and all those below it, in the pslist output
- Output of the yarascan
- Output of your privilege comparison

2. Any additional information as directed by the lab:

- Document the three processes that connected to 205.134.253.10:4444.
- Document the name and purpose of the software you discovered.

Section 1: Hands-On Demonstration

Part 1: Capture Memory using DumpIt

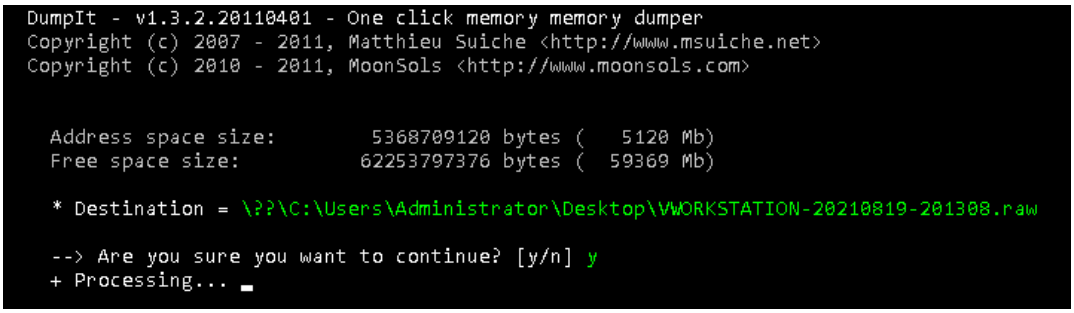
Note: In this part of the lab, you will use the free DumpIt utility to capture and save a memory dump from the vWorkstation. DumpIt is a contemporary combination of two older tools, win32dd and win64dd, and is used to capture data in a computer's volatile memory. When launched, either directly on the host or via a USB drive attached to the host, DumpIt takes a snapshot of the host's physical memory and saves it to the folder where the DumpIt executable is located. Although it does not provide analysis functions, DumpIt serves as a quick and effective tool for capturing volatile memory for later analysis.

1. On the vWorkstation desktop, **double-click** the **DumpIt icon** to open the DumpIt application.

DumpIt icon

Note: The DumpIt application will open in a new window. DumpIt will display the current system memory size as "Address space size," which refers to the amount of RAM to be acquired. The resulting image file generated will have approximately the same size. The path where the file is saved is shown in "Destination."

2. At the prompt, **type y** to start the memory capture process.



DumpIt interface

Note: The memory capture process will take up to 5 minutes to run. When the process is complete, a success notification will appear.

3. **Make a screen capture** showing the **DumpIt success notification**.

Note: Raw memory dumps typically contain a list of loaded drivers, processor information, process information, kernel mode, the stop message, and its parameters and much more. All this information in its raw form is not easily consumed by humans and is easier to understand when parsed by specialized tools.

4. **Press Enter** to close the DumpIt window.

Part 2: Analyze Memory using E3

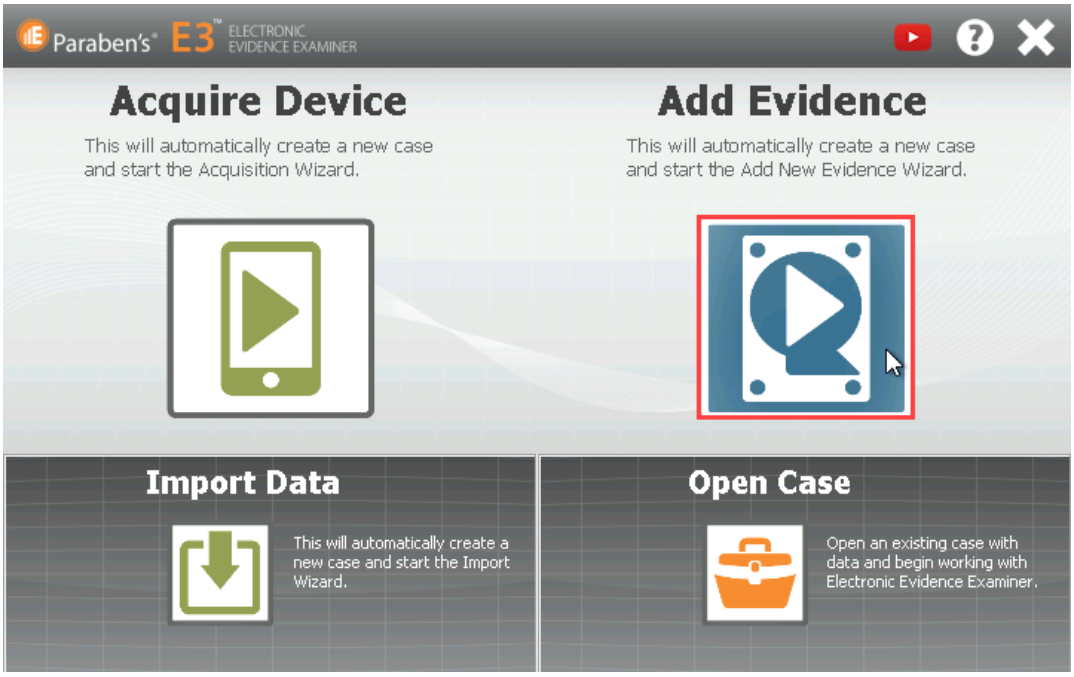
Note: In this part of the lab, you will use Paraben's E3 to analyze a simple memory dump. E3 contains integrated memory dump parsers that enable investigators to analyze system memory dumps from the same interface used for analyzing other forms of digital evidence. For the purposes of this lab, you will use an existing memory dump from a Windows machine, rather than the one you created in Part 1.

1. On the vWorkstation desktop, **double-click** the **Electronic Evidence Examiner icon** to open the E3 application.

E3 icon

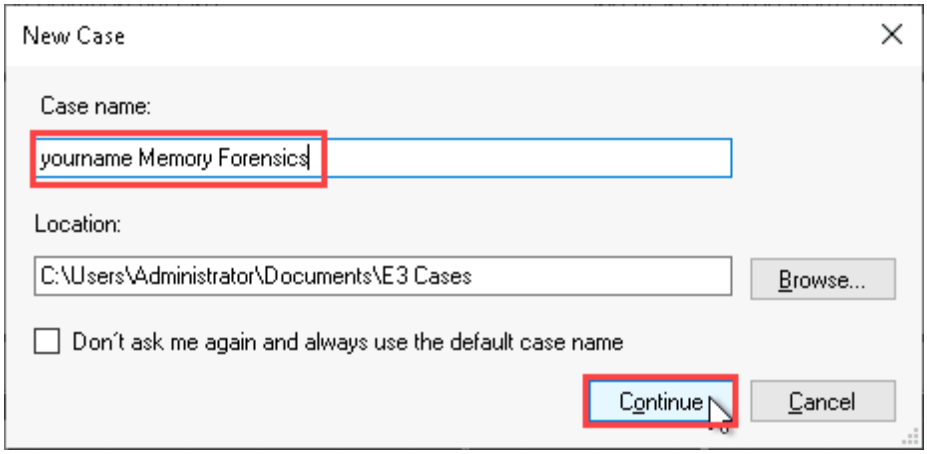
Note: E3 may take several minutes to load. The E3 welcome screen opens with shortcuts to the most common activities that forensic investigators will perform within the tool.

2. At the Welcome screen, **click** the **Add Evidence button** to open the New Case dialog box.



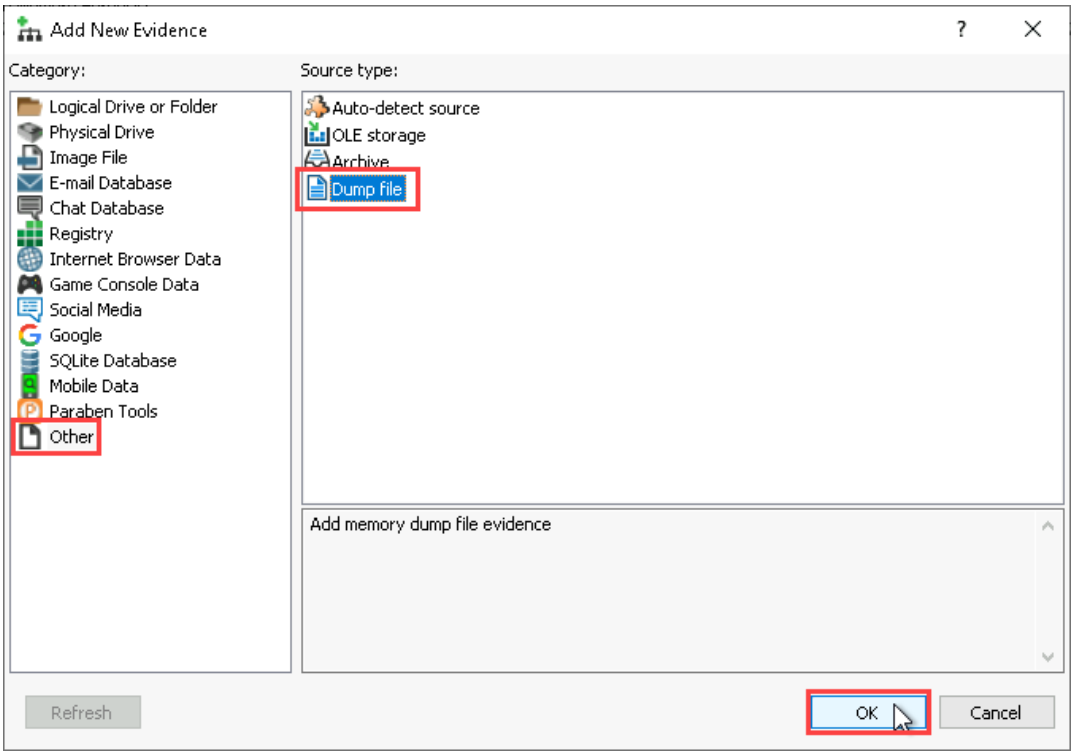
Welcome screen - Add Evidence

3. In the New Case window, **type** ***yourname*** **`Memory Forensics`** in the Case name field, replacing *yourname* with your own name, then **click Continue** to create your new case file and open the Add New Evidence window.



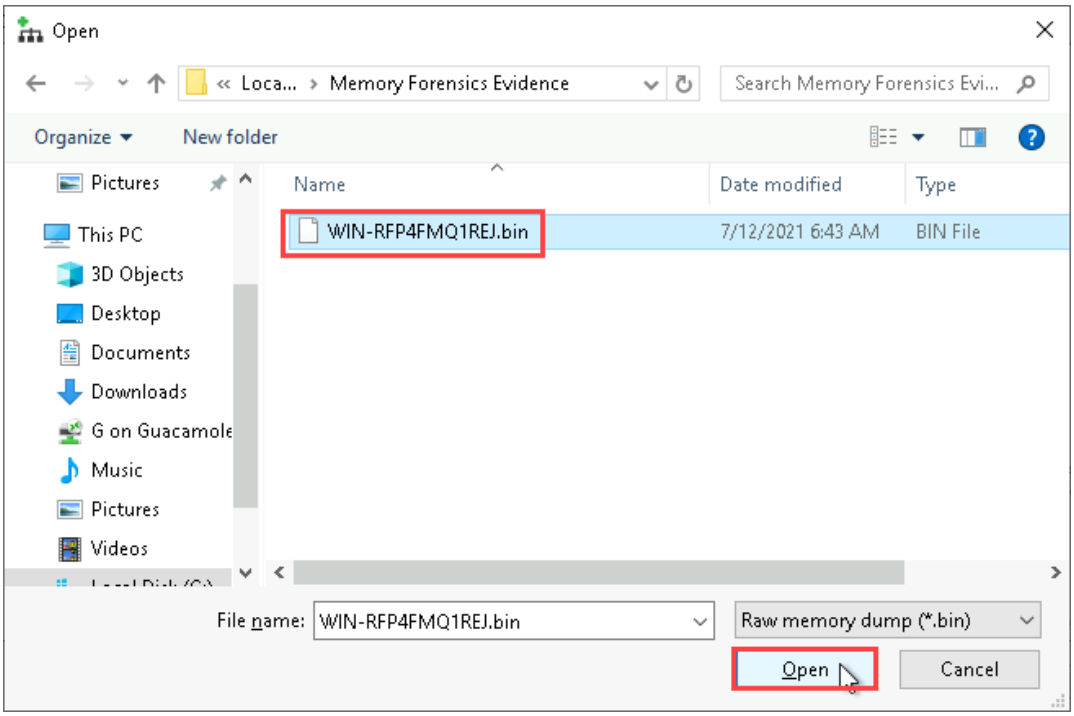
New Case dialog box

4. In the Add New Evidence window, **click** the **Other category**, then **select** the **Dump file source type** and **click OK** to continue.



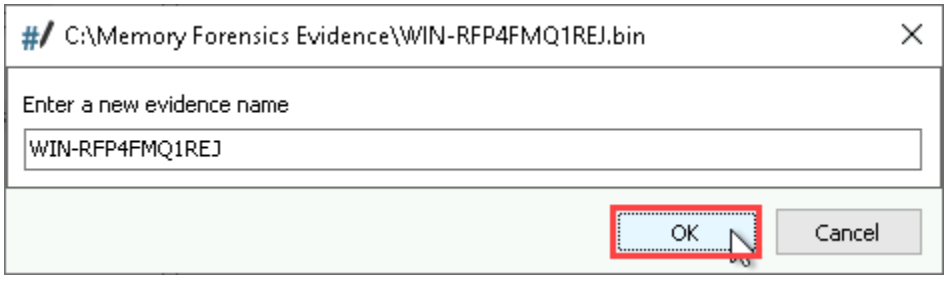
Add New Evidence - Dump file

5. In the Open dialog box, **navigate** to **This PC > Local Disk (C:) > Memory Forensics Evidence**, then **select** the **WIN-RFP4FMQ1REJ.bin file** and **click Open** to import the raw memory dump for this lab.



Open dialog box

6. When prompted, **click OK** to accept the default name for the memory dump and add the data from the memory dump to your case file.



Evidence name

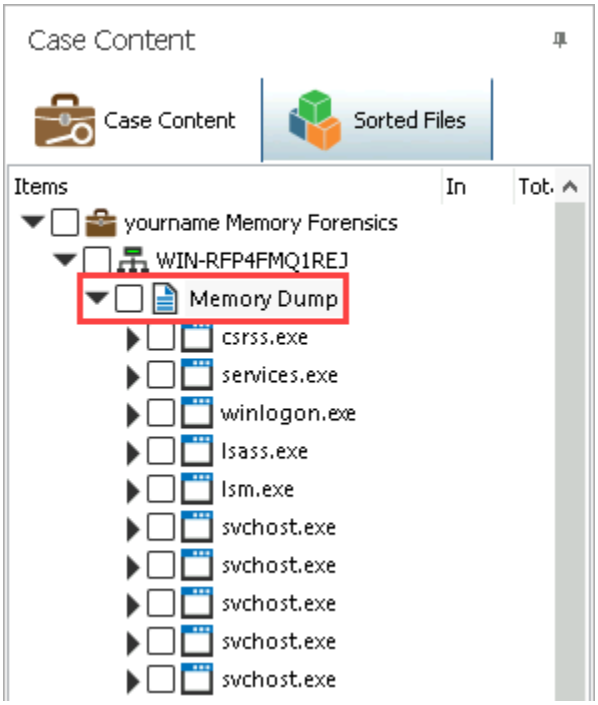
Note: The *yourname* Memory Forensics case will appear in the Case Content pane. The Case Content pane is the primary display and navigation area for all evidence in the open case file. Within

the Case Content pane's tree-view structure, you can access case nodes, evidence nodes, evidence type nodes, folder nodes, and data grids/streams. Technically, the Case Content pane does not store the actual evidence, but provides a series of links to elements within the physical evidence.

When you select a specific node or grid within the Case Content pane, the contents of the enclosing folder or database will be displayed in the Data Viewer in the center pane. Within the Data Viewer, you can select individual files, folders, and records. Additional information about the currently selected node, grid, file, folder, or record will be displayed in the Viewers pane on the right, which provides multiple ways to view your current selection.

7. In the Case Content pane, **navigate** to ***yourname* Memory Forenics / WIN-RFP4FMQ1REJ**, then **expand** the **Memory Dump node** to display the contents in the Data Viewer pane.

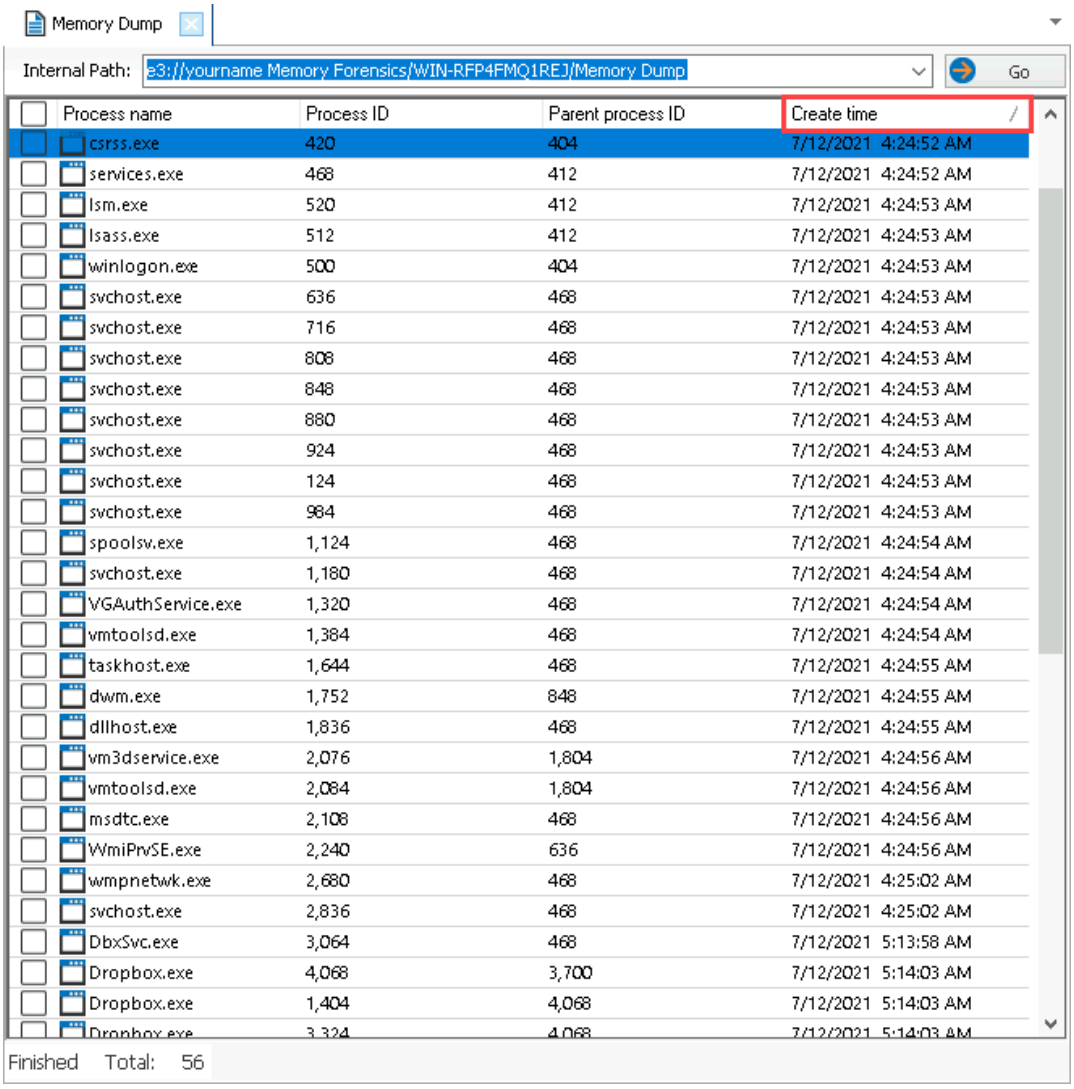
   You may need to re-select the Memory Dump node once E3 has finished processing the evidence.



Case Content

8. **Make a screen capture** showing the **list of processes in the memory dump**.

9. In the Data Viewer pane, **click** the **Create time column header** to sort the processes by creation time.



Data Viewer

Note: When analyzing a memory dump in response to a specific incident, it is often advisable for investigators to begin by looking for processes that started around the time that the incident was first identified, then working backwards. Similarly, if a system has been running for an extended period of time, a best practice is to begin by reviewing the most recently started processes, which might indicate a deviation from the system's baseline behavior. Of course, these practices should all be assessed within the context of the known facts of the investigation. Depending on the investigation, it is also possible that a malicious process was started at boot-up, which could indicate that the system had

previously been compromised at an earlier date.

10. **Record** the start times for the oldest process and the newest process.

Note: While a real investigation would likely require in-depth analysis of many of the processes captured in a memory dump, for the purposes of this lab, you will only examine the five most recently started processes.

One of the most recent processes is the DumpIt.exe process, which makes sense, given that it was likely the last application that the user had started when the memory capture began. You should also see a cmd.exe process, which is a known process on Windows machines associated with the Command Prompt application. You should also see two instances of a process titled conhost.exe and one instance of a process titled hooker.exe. In the next steps, you will research the conhost.exe and hooker.exe processes to determine their function and whether they could be considered suspicious and possibly malicious.

11. From the vWorkstation taskbar, **click** the **Chrome icon** to launch the Chrome application.

    Chrome will automatically open to the Google search engine.

Chrome icon

Note: As an investigator, you can typically search for the name of an executable in Google to determine whether it is a suspicious process. If you cannot find any information about the process at all, you can probably safely conclude that it is suspicious.

13. In the Google search field, **type** `conhost.exe` and **press Enter** to look up information about the first process.

14. **Review** the search results to determine if the conhost.exe process is suspicious.

Note: You should find documentation indicating that this process is harmless. Among the results, you should see a page from file.net, which provides comprehensive lists of common files and processes and information about them.

15. **Document** your findings for the conhost.exe process. What is it and what is it used for?

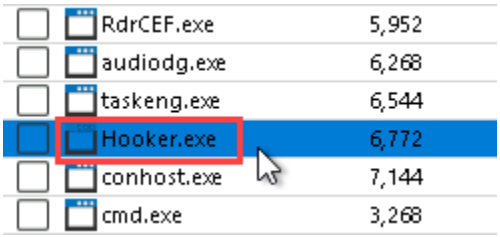
16. **Repeat steps 12-14** for the hooker.exe process.

Note: In this case, you should find documentation indicating that this process warrants further investigation. In the next steps, you will document your findings and conduct further analysis on the hooker.exe process in E3.

17. **Document** your findings for the hooker.exe process. What is it and what is it used for?

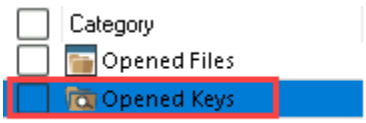
18. **Close** the **Chrome window**.

19. In the Data Viewer pane, **double-click** the **Hooker.exe process** to display additional information about the Hooker.exe process.



Hooker.exe

20. In the Data Viewer pane, **double-click** the **Opened Keys category** to display the contents,

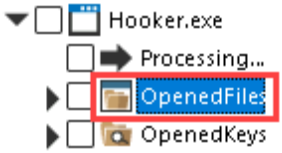


Opened Keys

Note: The Opened Keys view displays the Windows Registry keys associated with a specific process in the memory dump. Keys can be thought of as folders for values, which store configuration data. Each record in the Opened Keys view provides the key's file path within the Registry.

21. **Make a screen capture** showing the **registry keys opened by the Hooker.exe process**.

22. In the Case Content pane, **select** the **Opened Files node** to display the contents in the Data Viewer pane.



Opened Files

Note: The Opened Files view shows which files were accessed by a specific process. In this case, you should see that the Hooker.exe process accessed the \Users\Henry Smith\Downloads\hooker-3.4 file. The fact that this file is located in the Downloads folder could be an indicator that the process might be malicious or at least unwanted, as malware is often downloaded accidentally by the user and saved to the default Downloads location.

23. **Make a screen capture** showing the **files opened by the hooker.exe process**.

23. **Close** the **E3 window**.

Note: This concludes Section 1 of the lab. If you have been assigned Section 2, please reset your lab before continuing.

Section 2: Applied Learning

Note: SECTION 2 of this lab allows you to apply what you learned in **SECTION 1** with less guidance and different deliverables, as well as some expanded tasks and alternative methods. You will also use the Density Scout utility to assess suspicious files and determine whether they contain viruses.

If you have already completed Section 1, please reset your lab before continuing to the Section 2 activities.

Part 1: Capture Memory using FTK Imager

Note: In this part of the lab, you will learn to take a memory capture using a different tool – FTK Imager. FTK Imager is a free disk imaging and data preview tool developed by AccessData, now owned by Exterro. FTK Imager is a standalone utility associated with the Forensic Toolkit (FTK), a professional-grade digital forensics tool suite that offers enhanced search and analysis functionality for a variety of digital evidence formats. FTK Imager allows you to create forensic images, preview files and folders, mount an image for read-only viewing, recover deleted files, create hashes of files, and generate hash reports.

In addition to its disk imaging and data preview capabilities, FTK Imager also provides a built-in memory capture function. Compared to DumpIt, FTK Imager's memory capture function can create a more robust memory dump with more artifacts, but also requires considerably more RAM to run. Although FTK Imager provides data preview functionality for drive images and other forms of evidence, the same functionality does not extend to memory dumps. To analyze a memory dump created using FTK Imager, you will either need the full version of the Forensic Toolkit or a separate tool.

In the next steps, you will open FTK Imager and create a memory dump.

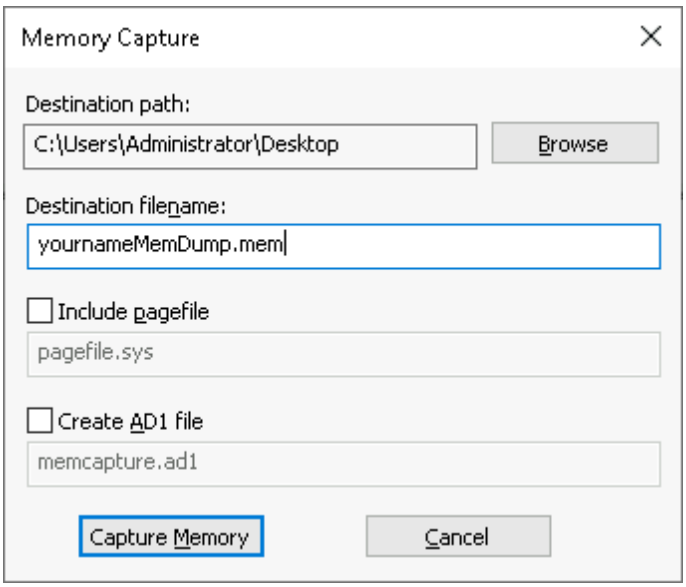
1. From the vWorkstation desktop, **open** the **FTK Imager application**.

2. From the FTK Imager menu bar, **click File** and **select Capture Memory** to open the Memory Capture window.

3. In the Memory Capture window, **click Browse** and **select** the **Desktop** as the target location for saving the output of the memory capture.

4. **Type** ***yourname*****MemDump.mem** in the Destination file name field, replacing *yourname* with

your own name.



Memory Capture

5. **Click** the **Capture Memory button** to begin the capture process.

6. **Make a screen capture** showing the ***Memory capture finished successfully*** **confirmation.**

7. **Close** the **FTK Imager window**.

Part 2: Analyze Memory using Volatility

Note: In this part of the lab, you will perform a more advanced analysis of the same memory dump used in Section 1 with the memory forensics tool Volatility. Volatility is a Python-based open-source memory forensics tool that is commonly used for incident response and malware analysis. Unlike E3, Volatility runs in the command line interface. The command line, while powerful, may require additional effort on the part of the investigator to distill information from the memory dump, depending on their level of familiarity.

1. From the vWorkstation taskbar, **open** the **Command Prompt**.

   For the best results, **maximize** the Command Prompt window.

2. At the command prompt, **execute** `C:\volatility.exe -f "C:\Memory Forensics Evidence\WIN-RFP4FMQ1REJ.bin" --profile=Win7SP1x64 pstree` to open the memory dump file in Volatility with the Win7SP1x64 profile and display the output using the pstree module.

```
C:\Users\Administrator>C:\volatility.exe -f "C:\Memory Forensics Evidence\WIN-RFP4FMQ1REJ.bin" --profile=Win7SP1x64 pstree
Volatility Foundation Volatility Framework 2.6
Name                                                 Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0xfffffa80268c5760:wininit.exe                       412    352      3     81 2021-07-12 10:24:52 UTC+0000
. 0xfffffa8026931b00:lsass.exe                        512    412      9    642 2021-07-12 10:24:53 UTC+0000
. 0xfffffa8026924b00:lsm.exe                          520    412     10    161 2021-07-12 10:24:53 UTC+0000
. 0xfffffa8026920b00:services.exe                     468    412      7    225 2021-07-12 10:24:52 UTC+0000
.. 0xfffffa802772c5e0:TrustedInstall                 4416    468      5    249 2021-07-12 11:14:26 UTC+0000
.. 0xfffffa80271e1060:svchost.exe                    2836    468      9    138 2021-07-12 10:25:02 UTC+0000
.. 0xfffffa8026a5b060:svchost.exe                     924    468     37   1158 2021-07-12 10:24:53 UTC+0000
... 0xfffffa8027bafb00:taskeng.exe                   6544    924      4     88 2021-07-12 12:34:14 UTC+0000
.. 0xfffffa802695d320:taskhost.exe                   4720    468      5    103 2021-07-12 11:19:58 UTC+0000
.. 0xfffffa8026a0ab00:svchost.exe                     808    468     20    478 2021-07-12 10:24:53 UTC+0000
... 0xfffffa80270ffb00:audiodg.exe                   6268    808      7    140 2021-07-12 12:33:37 UTC+0000
.. 0xfffffa8026b973b0:svchost.exe                    1180    468     17    322 2021-07-12 10:24:54 UTC+0000
.. 0xfffffa8026e85b00:dllhost.exe                    1836    468     13    204 2021-07-12 10:24:55 UTC+0000
.. 0xfffffa8026dcdb00:SearchIndexer.                 4336    468     13    649 2021-07-12 12:32:43 UTC+0000
.. 0xfffffa8026f8c430:msdtc.exe                      2108    468     12    153 2021-07-12 10:24:56 UTC+0000
.. 0xfffffa80269a04a0:svchost.exe                     716    468      8    302 2021-07-12 10:24:53 UTC+0000
.. 0xfffffa8026a2d060:svchost.exe                     848    468     17    423 2021-07-12 10:24:53 UTC+0000
```

Volatility - pstree

Note: Running Volatility using the pstree module will output a list of the processes in the target capture file displayed in a tree view, which can aid forensic investigators with identifying the relationships between different processes. When analyzing a memory capture, it is common practice to determine the relationships between different processes and identify which processes were spawned by which. This correlation exercise is especially important when determining if standard processes like cmd.exe were spawned by a suspicious process. In the next steps, you will trace a cmd.exe process back to an originating parent process.

3. In the Command Prompt window, **locate** the **0xfffffa8026afc930:cmd.exe process**.

Note: Within the Volatility tree view, the relationships between processes is represented using a combination of periods. When one process spawns another, the newly spawned process will be marked with an additional period. For example, if you are looking at a process with three periods, you can identify the parent process by working your way up the list until you find a process with two periods. In turn, the parent process for the two-period process will be the next process up the list with one period.

4. In the Command Prompt window, **identify** the **parent process** for the **0xfffffa8026afc930:cmd.exe process.**

Note: You should identify the parent process as explorer.exe, which is a normal parent process for cmd.exe, given that a Command Prompt is commonly launched from the Windows Explorer.

In the next steps, you identify other processes that were spawned by explorer.exe.

5. In the Command Prompt window, **review** the other processes that follow the explorer.exe process.

Note: You should several processes spawned by explorer.exe, including notepad.exe, rvlkl.exe, AcroRd32.exe, DumpIt.exe, cmd.exe, and Hooker.exe. Of these processes, you already know from Section 1 that Hooker.exe is suspicious. Notepad.exe, AcroRd32, DumpIt.exe, and cmd.exe should all be familiar processes, which leaves rvlkl.exe as the only candidate for further investigation.

6. Using the Internet, **research** the **rvlkl.exe process**.

7. **Document** your findings for the rvlkl.exe process. What is it and what is it used for?

Note: In the next steps, you will run Volatility using the psxview module. The psxview module compares active processes with any other possible sources within the memory dump, which can reveal hidden processes that might not otherwise be seen. Hidden processes will have a False value in the pslist column. Hidden processes, especially if they are normally not hidden (for example, dllhost.exe) might be a indicator of malware.

8. At the command prompt, **execute** `C:\volatility.exe -f "C:\Memory Forensics Evidence\WIN-RFP4FMQ1REJ.bin" --profile=Win7SP1x64 psxview` to open the memory dump file in Volatility with the Win7SP1x64 profile and display the output using the psxview module.

```
C:\Users\Administrator>C:\volatility.exe -f "C:\Memory Forensics Evidence\WIN-RFP4FMQ1REJ.bin" --profile=Win7SP1x64 psxview
Volatility Foundation Volatility Framework 2.6
Offset(P)          Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
------------------ -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x00000000be320b00 services.exe            468 True   False  False    True   True  True    True
0x00000000be0eab00 svchost.exe             984 True   False  False    True   True  True    True
0x00000000be00ab00 svchost.exe             808 True   False  False    True   True  True    True
0x00000000bdfcdb00 SearchIndexer.         4336 True   False  False    True   True  True    True
0x00000000bd50d790 Dropbox.exe            1404 True   False  False    True   True  True    True
0x00000000bd1afb00 taskeng.exe            6544 True   False  False    True   True  True    True
0x00000000be15ab00 spoolsv.exe            1124 True   False  False    True   True  True    True
0x00000000bd137060 AcroRd32.exe           5508 True   False  False    True   True  True    True
0x00000000be331b00 lsass.exe               512 True   False  False    True   True  True    False
0x00000000be3a04a0 svchost.exe             716 True   False  False    True   True  True    True
0x00000000bdd8c430 msdtc.exe              2108 True   False  False    True   True  True    True
0x00000000bd548790 Dropbox.exe            4068 True   False  False    True   True  True    True
0x00000000bd493b00 conhost.exe            6000 True   False  False    True   True  True    True
0x00000000bdfae060 dwm.exe                1752 True   False  False    True   True  True    True
0x00000000bd3fcb00 RdrCEF.exe             5952 True   False  False    True   True  True    True
0x00000000bd5e6b00 QtWebEnginePro         3240 True   False  False    True   True  True    True
0x00000000be1973b0 svchost.exe            1180 True   False  False    True   True  True    True
0x00000000bdf71060 taskhost.exe           1644 True   False  False    True   True  True    True
```

Volatility - psxview

9. **Document** whether any processes are flagged as hidden.

Note: In the next steps, you will run Volatility using the netscan module. The netscan module displays information about the network usage associated with each process, including protocol, IP addresses, and state.

10. At the command prompt, **execute** **`C:\volatility.exe -f "C:\Memory Forensics Evidence\WIN-RFP4FMQ1REJ.bin" --profile=Win7SP1x64 netscan`** to open the memory dump file in Volatility with the Win7SP1x64 profile and display the output using the netscan module.

```
C:\Users\Administrator>C:\volatility.exe -f "C:\Memory Forensics Evidence\WIN-RFP4FMQ1REJ.bin" --profile=Win7SP1x64 netscan
Volatility Foundation Volatility Framework 2.6
Offset(P)          Proto    Local Address                  Foreign Address      State            Pid      Owner          Created
0xbd0c1470         UDPv4    0.0.0.0:50525                  *:*                                   304      chrome.exe     2021-07-12 12:40:24 UTC+0000
0xbd580830         UDPv4    0.0.0.0:17500                  *:*                                   4068     Dropbox.exe    2021-07-12 11:15:04 UTC+0000
0xbd0adac0         TCPv4    0.0.0.0:17500                  0.0.0.0:0            LISTENING        4068     Dropbox.exe
0xbd0adac0         TCPv6    :::17500                       :::0                 LISTENING        4068     Dropbox.exe
0xbd12e9a0         TCPv4    127.0.0.1:17600                0.0.0.0:0            LISTENING        4068     Dropbox.exe
0xbd417ee0         TCPv4    0.0.0.0:17500                  0.0.0.0:0            LISTENING        4068     Dropbox.exe
0xbd0c3490         TCPv4    192.168.106.144:52312          162.125.19.130:443   CLOSED           -1
0xbd990e40         UDPv4    0.0.0.0:0                      *:*                                   924      svchost.exe    2021-07-12 12:42:53 UTC+0000
0xbd990e40         UDPv6    :::0                           *:*                                   924      svchost.exe    2021-07-12 12:42:53 UTC+0000
0xbdaab160         UDPv6    fe80::b09a:bf39:49b7:6f1e:63647 *:*                                  2836     svchost.exe    2021-07-12 10:25:03 UTC+0000
0xbdb70a80         UDPv4    0.0.0.0:5355                   *:*                                   984      svchost.exe    2021-07-12 12:39:55 UTC+0000
0xbdbdc520         UDPv4    192.168.106.144:63649          *:*                                   2836     svchost.exe    2021-07-12 10:25:03 UTC+0000
0xbdbdc7c0         UDPv6    ::1:63648                      *:*                                   2836     svchost.exe    2021-07-12 10:25:03 UTC+0000
0xbdbe9360         UDPv4    127.0.0.1:1900                 *:*                                   2836     svchost.exe    2021-07-12 10:25:03 UTC+0000
0xbdbed010         UDPv6    fe80::b09a:bf39:49b7:6f1e:1900 *:*                                   2836     svchost.exe    2021-07-12 10:25:03 UTC+0000
0xbdbedbb0         UDPv4    192.168.106.144:1900           *:*                                   2836     svchost.exe    2021-07-12 10:25:03 UTC+0000
0xbdbee9a0         UDPv6    ::1:1900                       *:*                                   2836     svchost.exe    2021-07-12 10:25:03 UTC+0000
0xbdea8500         UDPv6    fe80::b09a:bf39:49b7:6f1e:546  *:*                                   808      svchost.exe    2021-07-12 12:33:10 UTC+0000
0xbdedb340         UDPv4    0.0.0.0:5355                   *:*                                   984      svchost.exe    2021-07-12 12:39:55 UTC+0000
0xbdedb340         UDPv6    :::5355                        *:*                                   984      svchost.exe    2021-07-12 12:39:55 UTC+0000
0xbdeedd00         UDPv4    192.168.106.144:138            *:*                                   4        System         2021-07-12 10:24:55 UTC+0000
0xbdf10010         UDPv4    192.168.106.144:137            *:*                                   4        System         2021-07-12 10:24:55 UTC+0000
0xbdf28390         UDPv4    0.0.0.0:0                      *:*                                   984      svchost.exe    2021-07-12 10:24:55 UTC+0000
0xbdf28390         UDPv6    :::0                           *:*                                   984      svchost.exe    2021-07-12 10:24:55 UTC+0000
0xbd80f8a0         TCPv4    0.0.0.0:49157                  0.0.0.0:0            LISTENING        512      lsass.exe
0xbd80f8a0         TCPv6    :::49157                       :::0                 LISTENING        512      lsass.exe
0xbd9aeb70         TCPv4    127.0.0.1:843                  0.0.0.0:0            LISTENING        4068     Dropbox.exe
0xbdbff1d0         TCPv4    0.0.0.0:49157                  0.0.0.0:0            LISTENING        512      lsass.exe
0xbdc02780         TCPv4    0.0.0.0:49155                  0.0.0.0:0            LISTENING        468      services.exe
0xbdf12860         TCPv4    192.168.106.144:139            0.0.0.0:0            LISTENING        4        System
0xbdff1010         TCPv4    0.0.0.0:445                    0.0.0.0:0            LISTENING        4        System
0xbdff1010         TCPv6    :::445                         :::0                 LISTENING        4        System
0xbdff6ae0         TCPv4    0.0.0.0:49155                  0.0.0.0:0            LISTENING        468      services.exe
0xbdff6ae0         TCPv6    :::49155                       :::0                 LISTENING        468      services.exe
```

Volatilty - netscan

11. In the Command Prompt window, **review** the Volatility results.

Note: Having previously identified Hooker.exe and rvlkl.exe as potentially suspicious, you should attempt to identify whether either of these processes attempted to use the network.

12. **Document** whether the netscan module displays network usage associated with the Hooker.exe or rvlkl.exe processes.

Note: As you review the results, you may notice some processes with a PID value of -1. This value is used to designate processes with an unidentified PID. During a forensic investigation, processes with an unidentified PID often warrant further investigation, as the lack of information may be a sign of an deliberate attempt to obscure the process's purpose.

13. In the Command Prompt window, **identify** the **first process** with an **ESTABLISHED state** and an **offset value** of **0xbd7e6a50**.

Note: You should see that this connection was made between 192.168.106.144:56610 and 162.125.19.131:443, which should be read as a connection between a host with an IP address of 192.168.106.144 on port 56610 and a host with an IP address of 162.125.19.131 on port 443. Port 443 is a common web browsing port used primarily for HTTPS services. When analyzing a memory dump for signs of malware, it is a best practice to look up any unrecognized ports to determine if they are the default listening port for a specific piece of malware.

14. Using the Internet, **research port 56610**.

15. **Document** any information you were able to gather about port 56610.

Note: In the next steps, you will assess the Hooker.exe and rvlkl.exe processes using the DensityScout malware analysis utility. First, you will need to extract the associated executable files.

16. At the command prompt, **execute** `C:\volatility.exe -f "C:\Memory Forensics Evidence\WIN-RFP4FMQ1REJ.bin" --profile=Win7SP1x64 filescan | findstr rvlkl.exe` to open the memory dump file in Volatility with the Win7SP1x64 profile and scan the file for any processes containing the string "rvlkl.exe".

Note: This operation will take several minutes to run. Once concluded, you will be able to isolate the offset associated with the rlvlk.exe process, which you will use to extract the .exe file associated with the process.

17. In the Command Prompt window, **highlight** the **offset value**, then **press Ctrl+c** to copy it to the clipboard.

18. On the vWorkstation desktop, **right-click anywhere** and **select Create New > Text Document** from the context menu, then **name** the new file `offsets`.

19. **Open** the **offsets text file** and **paste** the contents of the clipboard.

20. **Repeat steps 16-19** for the Hooker.exe process, saving the offsets for both Hooker.exe results to the offsets file.

21. **Open** the **File Explorer**, then **create a new folder** in the C:\ directory titled **ExtractedFiles.**

22. From the offsets text file, **copy** the **offset** for the rvlkl.exe file.

Note: In the next steps, you will use the Volatility dumpfiles module to extract the files identified in the memory dump. To perform this operation, you will use the following Volatility options:

-Q : Use physical offset
-D : Specify data directory where file will be extracted
-u : Use unsafe mode
-n : Include file name in dump file

Processes can be also be extracted via PID, but using the offset allows you to extract other files detected during the scan.

23. At the command prompt, **execute** `C:\volatility.exe -f "C:\Memory Forensics Evidence\WIN-RFP4FMQ1REJ.bin" --profile=Win7SP1x64 dumpfiles -Q <offset> -D "C:\ExtractedFiles" -u -n`, pasting the offset in place of <offset>, to extract the file and save it to the C:\ExtractedFiles folder.

24. **Repeat steps 22-23** for the two Hooker.exe offsets.

Note: Now that the files have been extracted, you will use DensityScout to scan the files. DensityScout is designed to scan a target file path, calculate the density of each file, and output a list of files in descending order by density. Files that have a lower density score are typically considered to be suspicious, while higher density are considered normal. For example, Microsoft Windows executables are not packed or encrypted in any way, which give them a higher density. By comparison, malware is often packed and encrypted to obscure its true nature, which gives it a lower density.

25. At the command prompt, **execute** `C:\densityscout.exe -p 0.1 "C:\ExtractedFiles"` to run DensityScout on the extracted files.

Note: You should find that both files extracted from the dump received relatively high density scores,

which would indicate (but does not guarantee) that they do not contain malware.

26. **Make a screen capture** showing the **DensityScout results**.

27. **Close** any **open windows**.

Note: This concludes Section 2 of the lab.

Section 3: Challenge and Analysis

Note: The following exercises are provided to allow independent, unguided work using the skills you learned earlier in this lab - similar to what you would encounter in a real-world situation.

Part 1: Identify Malicious Connections

In this scenario, you will assume the role of a security analyst at a private company. It's Monday morning, and a member of the Desktop Support team has just sent you a ticket from Alice, a member of the Sales team who has reported unusual behavior and poor performance on her company laptop. In the ticket, the Desktop Support Engineer states that after reviewing Alice's laptop, she suspects the device has been infected with malware. In accordance with company policy, the device must be transferred to the security team for further analysis. Unfortunately, you are working remotely today, so you ask the Desktop Support Engineer to download the DumpIt utility to a USB flash drive, attach it to the laptop, create a memory dump, and send it to you.

Upon receiving memory dump, you decide to begin by looking for any suspicious network connections using Volatility's netscan module. If a bad actor obtained access to the machine, it is likely they are calling back home to their C2 (Command and Control) server from a malicious process on the machine, so go ahead and see if any such processes exist, using the following dump file and Volatility profile.

From the vWorkstation, open the command prompt, then invoke the Volatility netscan module using the following syntax and parameter values.

```
C:\volatility.exe -f <memoryDump> --profile=<OperatingSystem> netscan
```

- **Dump file:** "C:\Memory Forensics Evidence\ALICE-PC-Win7.raw"
- **Profile:** Win7SP1x64

You should notice that at the bottom of the output, there appears to be an outgoing connection to 205.134.253.10 on port 4444 from the fixtureComputer.exe application. There also appears to be two other processes with an established connection to the same address. You will need to take a note of these processes for further investigation. Also, port 4444 sounds familiar, so you decide to take a closer look at what it is typically used for.

Document the three processes that connected to 205.134.253.10:4444.

Using the Internet, research whether any software commonly uses port 4444 as its default listener.

Document the name and purpose of the software you discovered.

Part 2: Identify Malicious Processes

Based on your findings in Part 1, you have realized that Alice's laptop has been compromised by a

malicious third-party. Now that you have identified the malicious processes, you will review them in context and attempt to determine how far the intruder has made it.

At the command prompt, invoke the Volatility pslist module using the following syntax and parameter values:

```
C:\volatility.exe -f <memoryDump> --profile=<OperatingSystem> pslist
```

- **Dump file:** "C:\Memory Forensics Evidence\ALICE-PC-Win7.raw"
- **Profile:** Win7SP1x64

Make a screen capture showing the **fixtureComputer.exe process, and all those below it, in the pslist output.**

It appears fixtureComputer.exe was indeed opened by Firefox, indicated by its PPID of 2444 (the PID of one of the Firefox processes). Two minutes later, you can see whoami.exe was run, which looks like the hacker attempting to determine their current privileges. If they found that they do not yet have administrative privileges, you would expect them to attempt to elevate their privileges – which perhaps explains the purpose of the second connection. Three seconds after the whoami.exe command, you can see a strange tior.exe command pop into existence before being terminated two seconds later. Immediately after, you should see the QaNoQBC.exe process, which you know connected back to the strange IP address.

After conducting some research online, you have discovered tior.exe is commonly used in a User Account Control bypass exploit, which circumvents those pesky Windows access control prompts ("Do you want to allow the following application…?") that show up when trying to run applications with administrative privileges. Because tior.exe was terminated, it may be difficult to find many details on it using the usual tactics, so you decide to scan the memory dump for signs of tior.exe using Volatility's yarascan pattern-matching module.

At the command prompt, invoke the Volatility yarascan module using the following syntax and parameter values:

```
C:\volatility.exe -f <memoryDump> --profile=<OperatingSystem> yarascan -Y
"<stringPattern>"
```

- **Dump file:** "C:\Memory Forensics\ALICE-PC-Win7.raw"
- **Profile:** Win7SP1x64
- **String Pattern:** tior.exe

The **-Y** option indicates you will specify your pattern inline as a string, rather than pointing to a predefined rule. After the first result appears on-screen, you can press Ctrl+c to end the search.

Otherwise, the scan may take roughly 5 minutes.

Make a screen capture showing the **output of the yarascan**.

Part 3: Identify Privilege Escalation

Another bleak discovery. Tior.exe was found in the address space of svchost.exe, as you can see at the top of the ASCII output in the rightmost column. Svchost is a legitimate Windows service, so it is possible the attacker is using the address space assigned to this process to run its injected code. Uh oh.

Alright, so if the hacker attempted to bypass UAC in order to launch a privileged connection back to their malicious server, then that second connection, QaNoQBC.exe (which was sprung immediately after tior.exe), should reflect what kind of privileges they were able to obtain.

This should be as easy as comparing the privileges between the first connection, fixtureComputer.exe, and the second connection, QaNoQBC.exe. The tior.exe application was issued right in between, so you expect to see elevated privileges for the QaNoQBC.exe process, if the hacker was successful. However, you figure you can make the output more concise by using the **`-silent`** flag to show only privileges that were explicitly enabled by the process (not available by default), effectively providing you with the difference.

First, determine the PIDs for both the fixtureComputer.exe and QaNoQBC.exe processes. You can refer to your earlier pslist output, if still available in your cmd window, or you can execute Volatility's pslist module again to obtain them.

Next, at the command prompt, invoke Volatility's privs module using the following syntax and parameter values:

```
C:\volatility.exe -f <memoryDump> --profile=<OperatingSystem> privs -p
<fixtureComputer.exe PID>,<QaNoQBC.exe PID> --silent
```

- **Dump file:** "C:\Memory Forensics Evidence\ALICE-PC-Win7.raw"
- **Profile:** Win7SP1x64

Make a screen capture showing the **output of your privilege comparison**.

You should notice several concerning privileges the QaNoQBC.exe process bestowed to itself – quite the power upgrade. At this point, you've seen enough to know that you will need to report this as a security incident as quickly as possible.

Note: This concludes Section 3 of the lab.