SAT1: 009: **Analyzing a Forensic Memory Capture**

Overview

This lab will also cover analyzing captured RAM with volatility on a Security Onion machine.

Note: Please note that to properly view converted RAM folder contents instead of simply the original bits, a more complex, paid program such as EnCase will be required.

Time: 30 Minutes

Learning Objectives

Upon completion of this lab, you should be able to:

1. Examine a sample RAM dump in the volatility tool

Resources

The following documents have been provided to assist you in this lab.

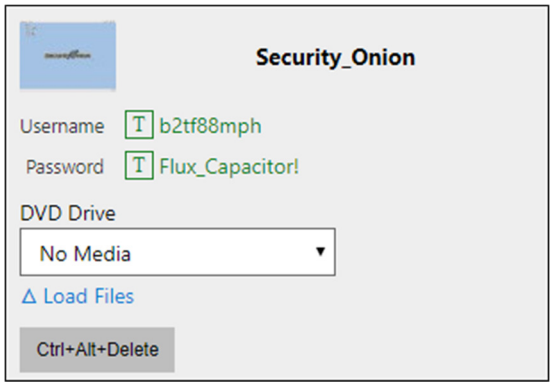
Linux Cheat Sheet

Sans Computer Forensics Cheat Sheet

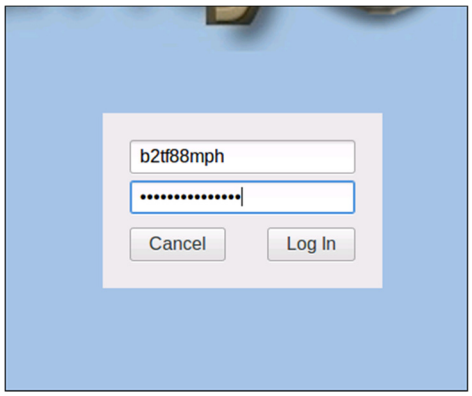
Volatility Cheat Sheet

Log in to the Lab Machine

Select the **Security_Onion** machine on the Machines Tab.

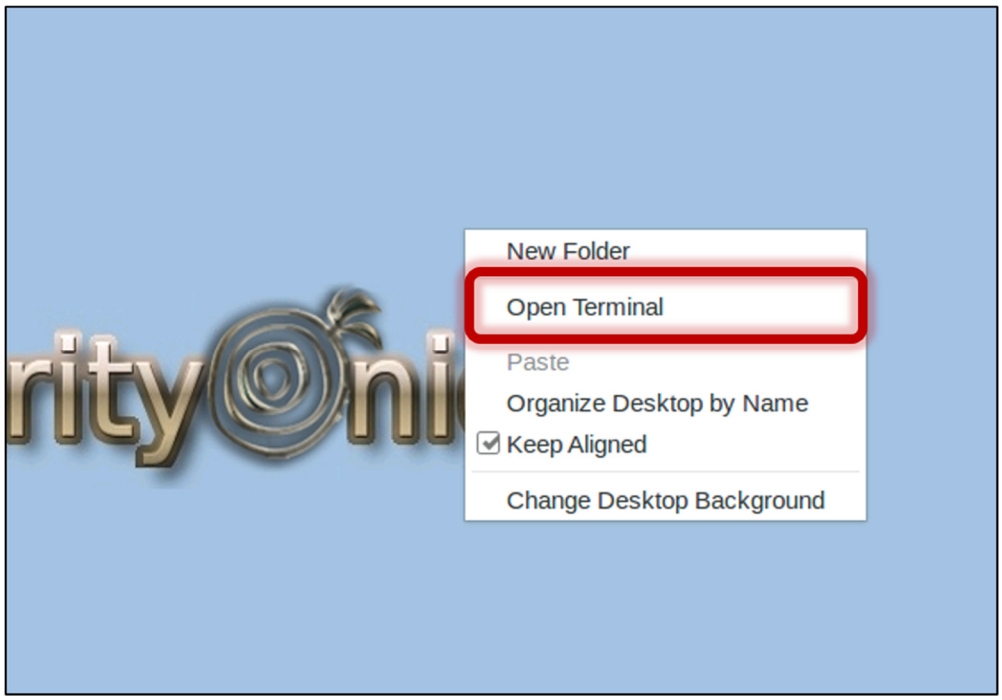


Enter the Username and Password on the **Security_Onion** machine. Click Log In.

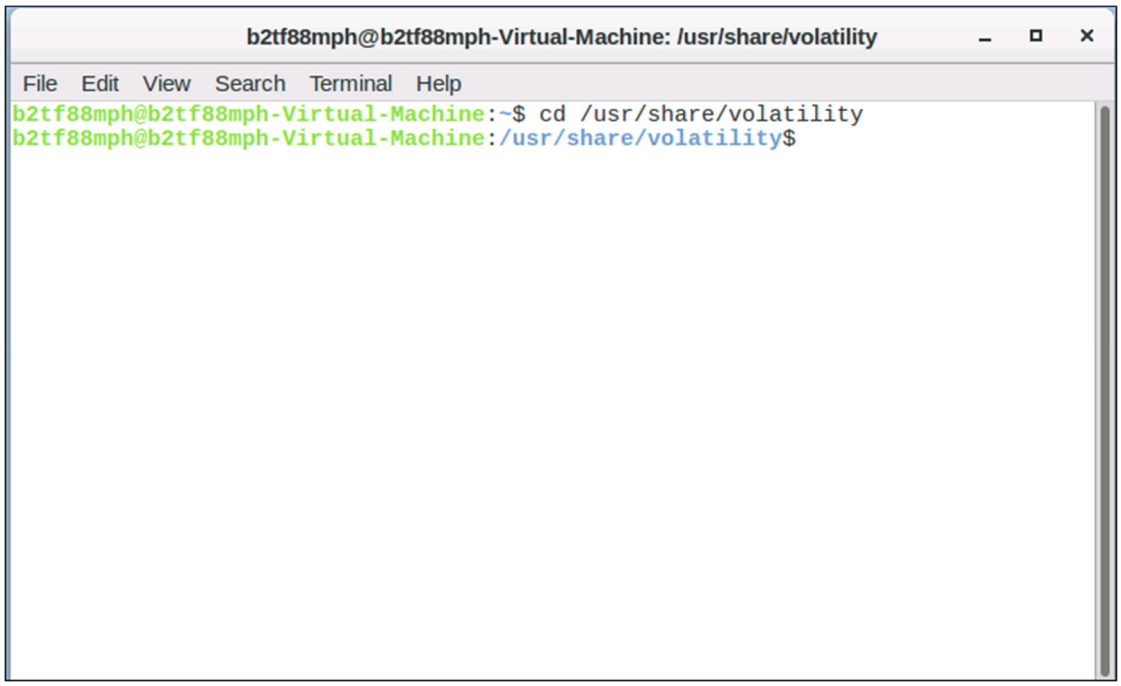


1.0 Analysis of a Memory Capture

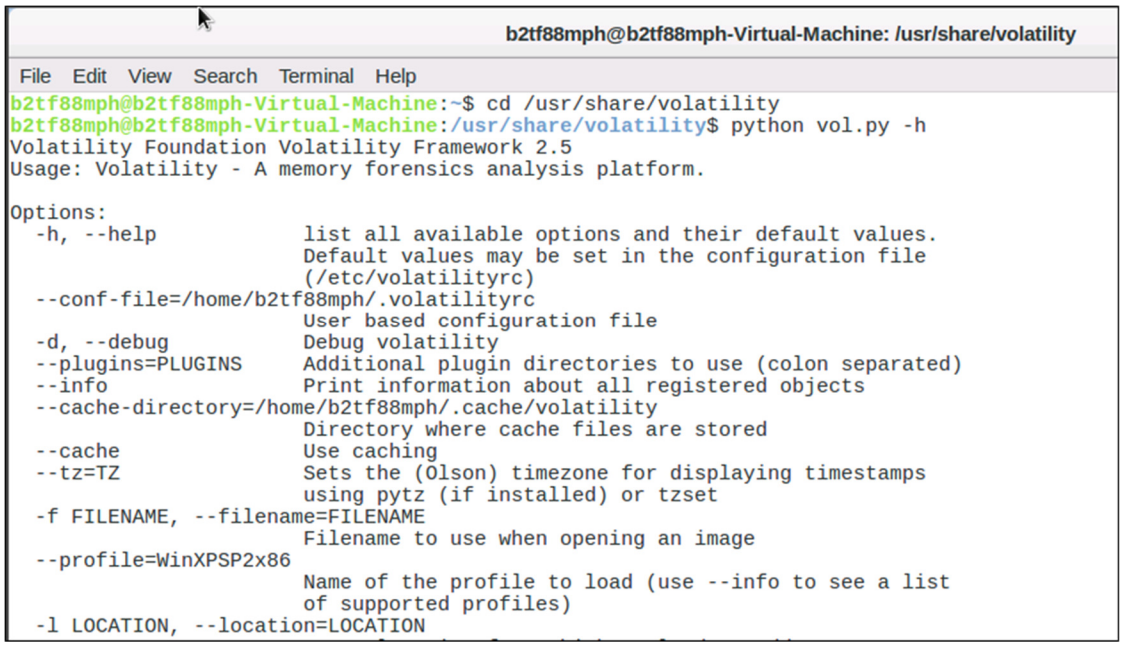
1.1 Right-click on the Security Onion Desktop, and select **Open Terminal**.



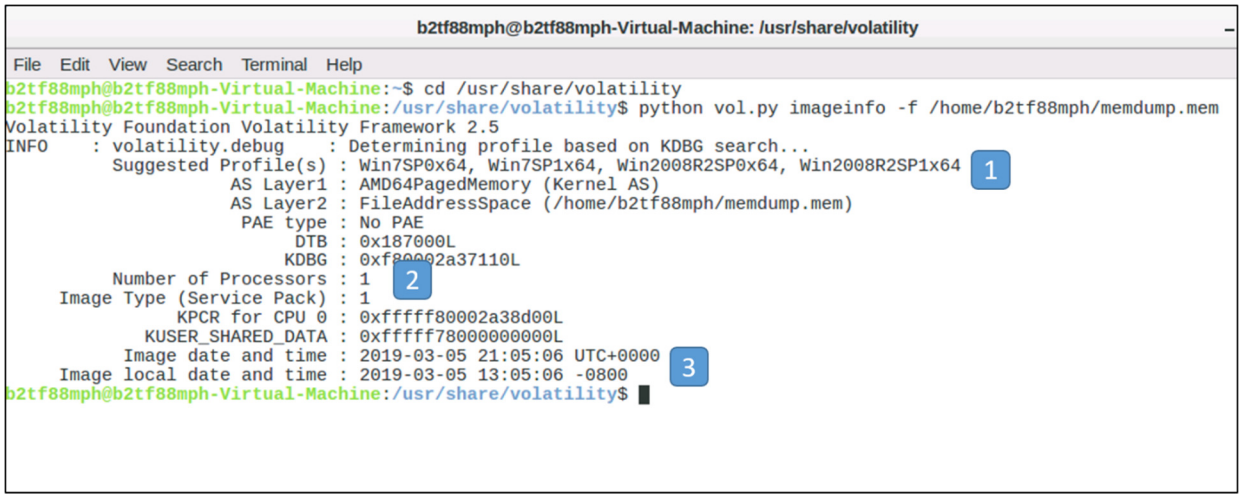
1.2 Type `cd /usr/share/volatility`,and press **Enter**.



1.3 Type `python vol.py -h`, this will display the help pages for volatility.



1.4 Type `python vol.py imageinfo -f /home/b2tf88mph/memdump.mem` and press **Enter**, this will give you vital information about the memory capture you are working with. It will take about `10minutes` to run the analysis.

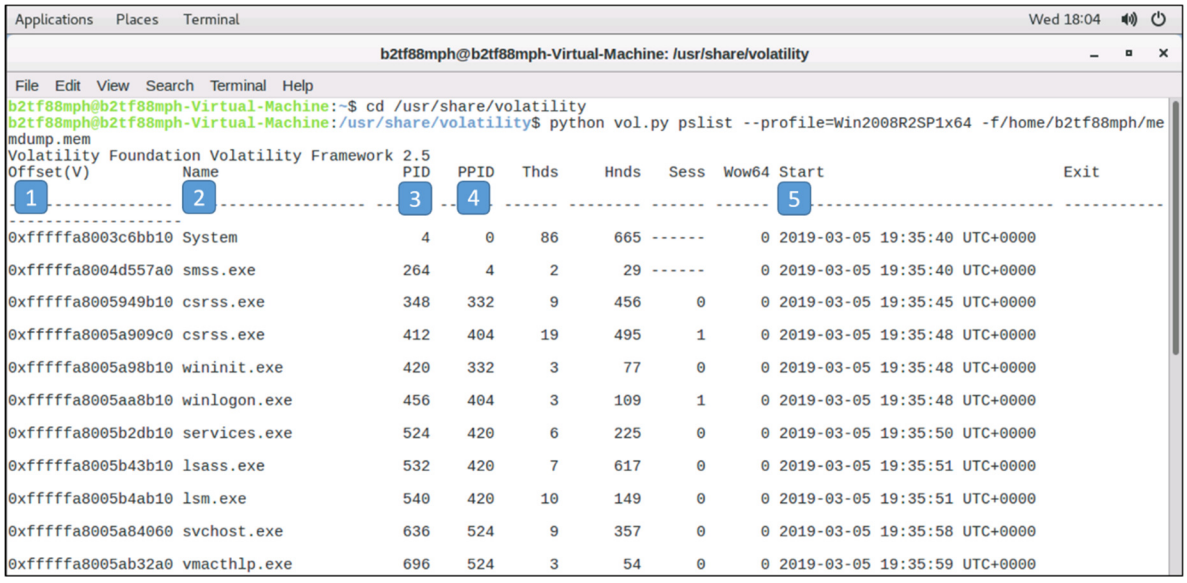


There is alot of good information including:

1. Suggested Volatility profiles to be used in analysis.
2. Number of Processors on the imaged machine.
3. Date and Time that the image was captured.

Note: Volatility needs to know what operating system was imaged in order to interpret the memory image correctly. The default profile is Win7SP1x64, but we used Win2008R2SP1x64, so we'll have to include that information in all future volatility command-lines.

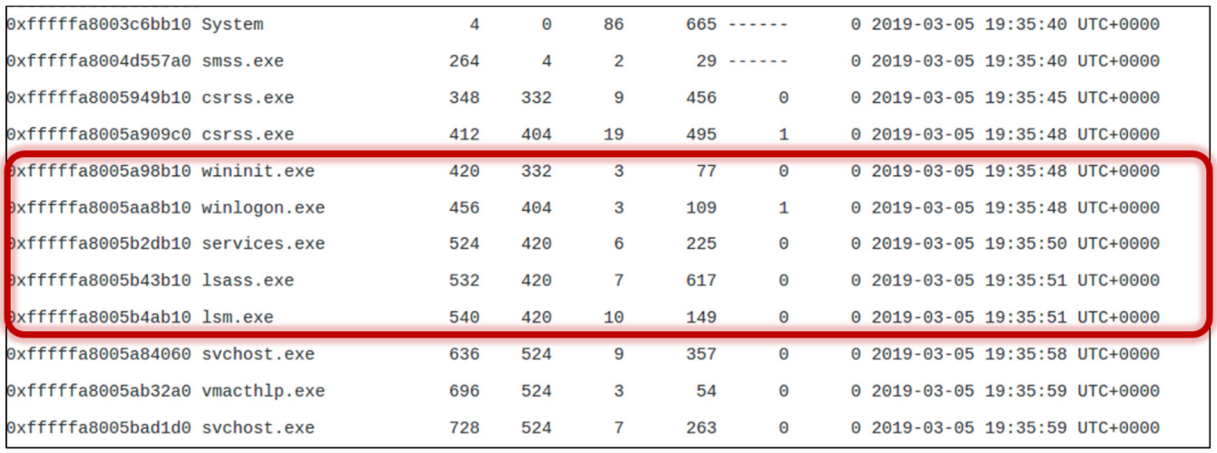
1.5 Type `python vol.py pslist --profile=Win2008R2SP1x64 -f/home/b2tf88mph/memdump.mem`, and press **Enter**.



Notice these columns:

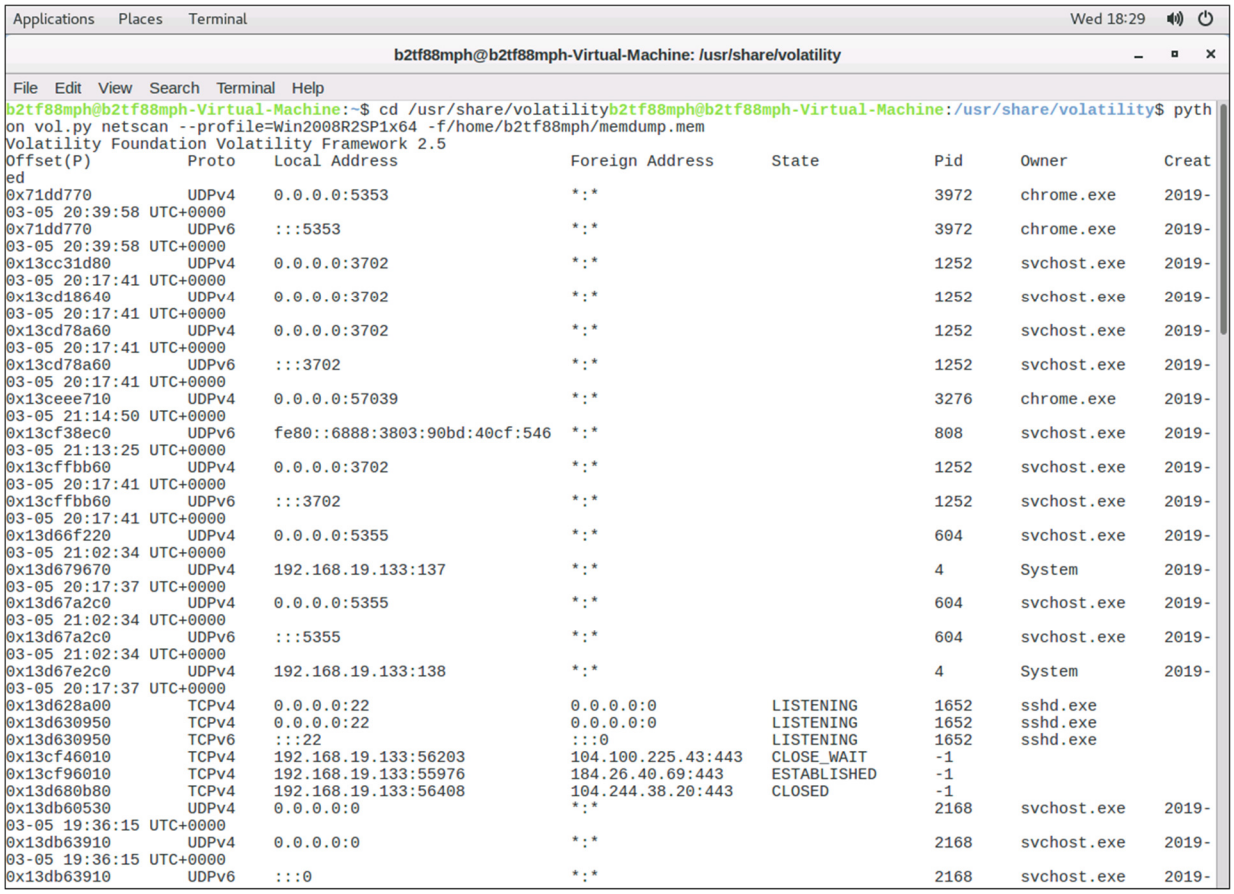
1. Offset: The location in RAM of the process, in hexadecimal.
2. Name: The process name, as it would be shown in Task Manager.
3. PID: The process ID.
4. PPID: The parent process ID--that is, the process that launched this process.
5. Time: The time the process was started.

It is also possible to see any recent console commands that have been executed. Though this may not always be useful, in this case we can see that there were some console commands recently run that contain information about user accounts recently created, along with the passwords that were set for those users.



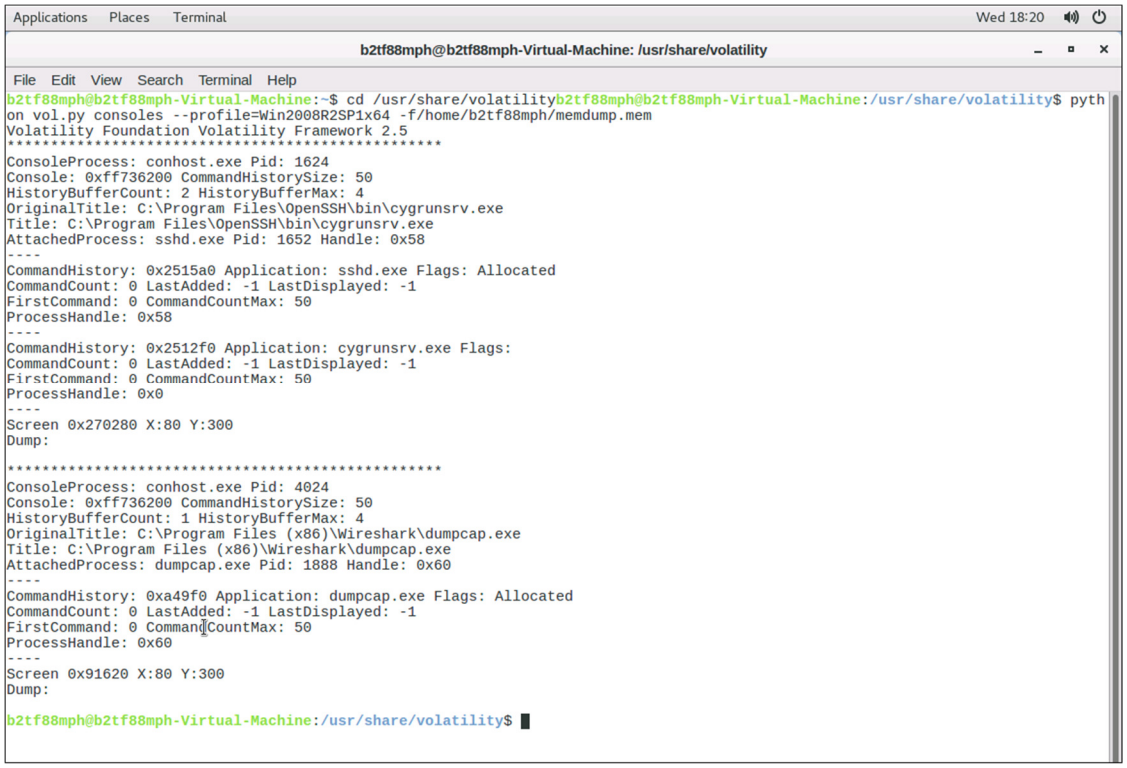
0xfffffa8003c6bb10	System	4	0	86	665	------	0	2019-03-05 19:35:40 UTC+0000
0xfffffa8004d557a0	smss.exe	264	4	2	29	------	0	2019-03-05 19:35:40 UTC+0000
0xfffffa8005949b10	csrss.exe	348	332	9	456	0	0	2019-03-05 19:35:45 UTC+0000
0xfffffa8005a909c0	csrss.exe	412	404	19	495	1	0	2019-03-05 19:35:48 UTC+0000
0xfffffa8005a98b10	wininit.exe	420	332	3	77	0	0	2019-03-05 19:35:48 UTC+0000
0xfffffa8005aa8b10	winlogon.exe	456	404	3	109	1	0	2019-03-05 19:35:48 UTC+0000
0xfffffa8005b2db10	services.exe	524	420	6	225	0	0	2019-03-05 19:35:50 UTC+0000
0xfffffa8005b43b10	lsass.exe	532	420	7	617	0	0	2019-03-05 19:35:51 UTC+0000
0xfffffa8005b4ab10	lsm.exe	540	420	10	149	0	0	2019-03-05 19:35:51 UTC+0000
0xfffffa8005a84060	svchost.exe	636	524	9	357	0	0	2019-03-05 19:35:58 UTC+0000
0xfffffa8005ab32a0	vmacthlp.exe	696	524	3	54	0	0	2019-03-05 19:35:59 UTC+0000
0xfffffa8005bad1d0	svchost.exe	728	524	7	263	0	0	2019-03-05 19:35:59 UTC+0000

1.6 Type `python vol.py netscan --profile=Win7SP1x64 -f /home/b2tf88mph/memdump.mem`, and press **Enter**.

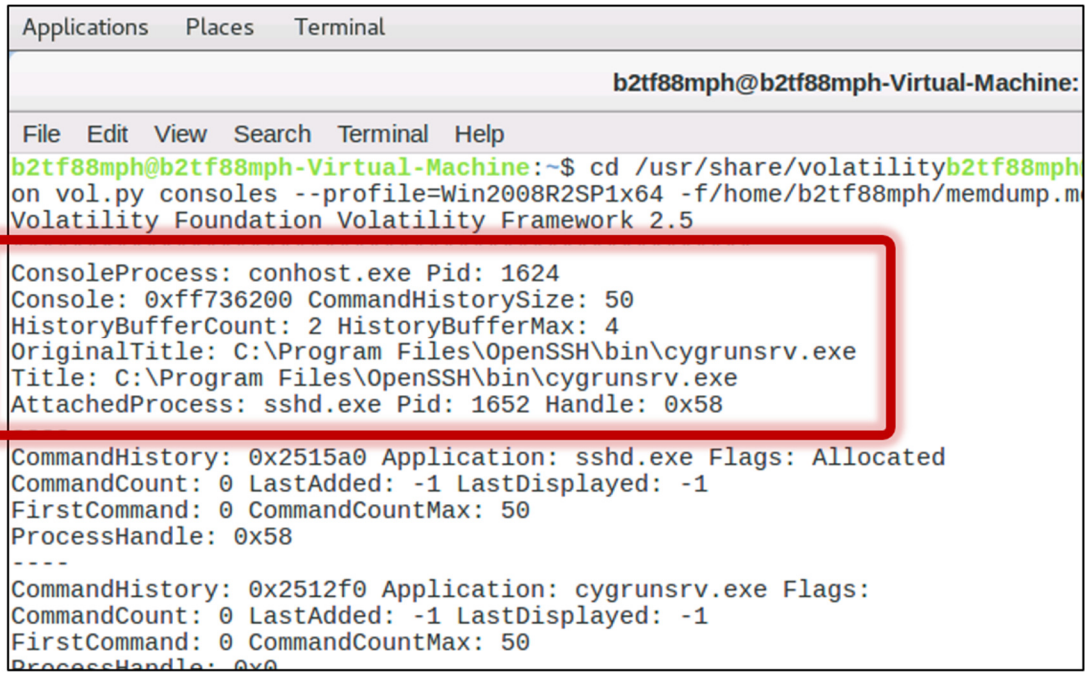


Looking through the network connections everything looks to be normal. Except one connection. Can you find the suspect Traffic?

1.7 Type `python vol.py consoles --profile=Win2008SP1x86 -f/home/b2tf88mph/memdump.mem`, and press **Enter**.

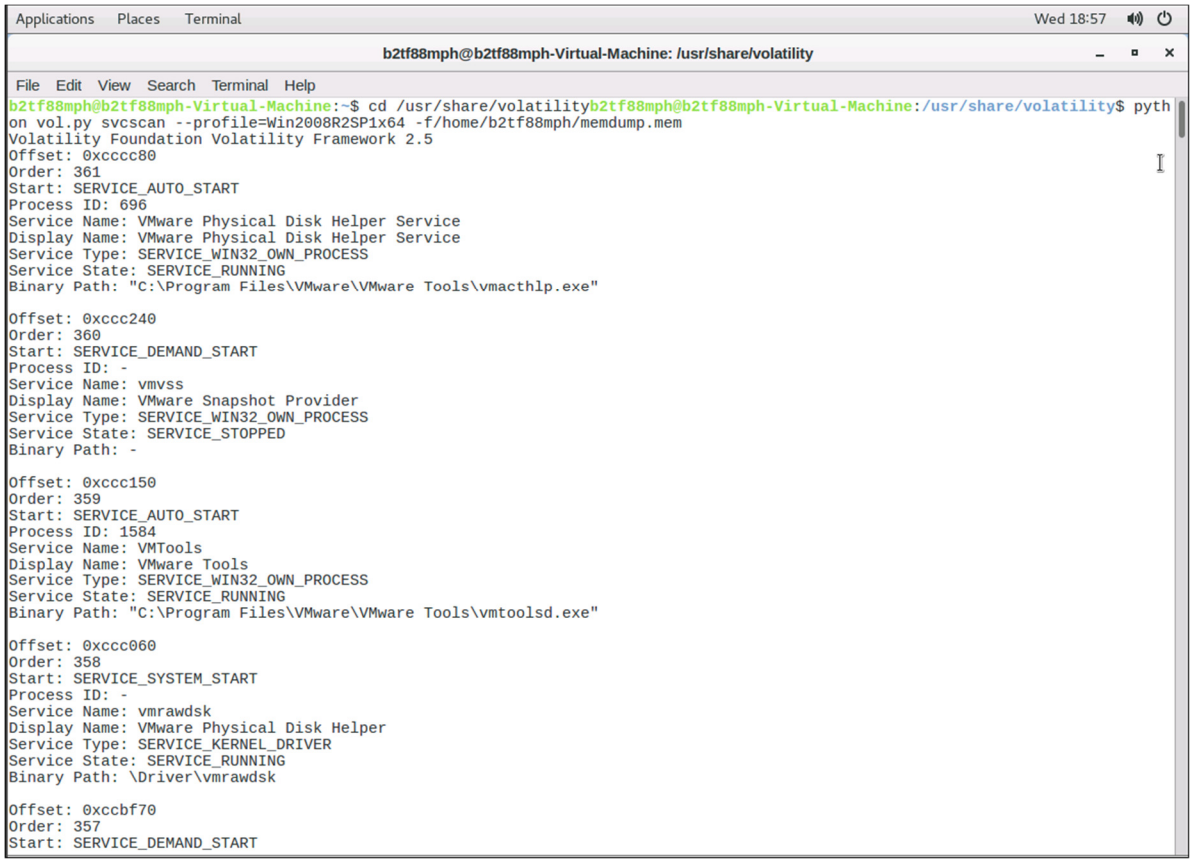


As you can see here, it is possible to see the services that were running at the time of the memory dump. This can be used to identify malware or other malicious programs running at the time of the memory dump.



Item of Interest: It might be odd that an executable with an attached ssh process ran from the console.

1.8 Type `python vol.py svcscan --profile=Win2008SP1x86 -f /root/Desktop/memdump.mem`, and press **Enter**.

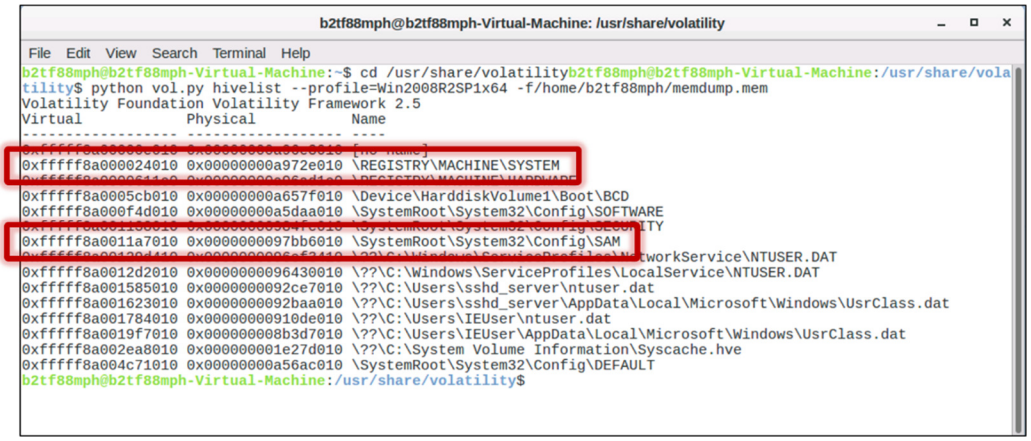


This output contains information about all processes running in memory.

Now we need to find the SAM and SYSTEM hives, while they are in this output we just created, there is a better way to find them.

Item of Interest: Those two hives together contain enough information to extract Windows password hashes.

1.9 We can use another command to identify the system hives directly. Type `python vol.py hivelist--profile=Win2008SP1x86 -f /root/Desktop/memdump.mem`, and press **Enter**.

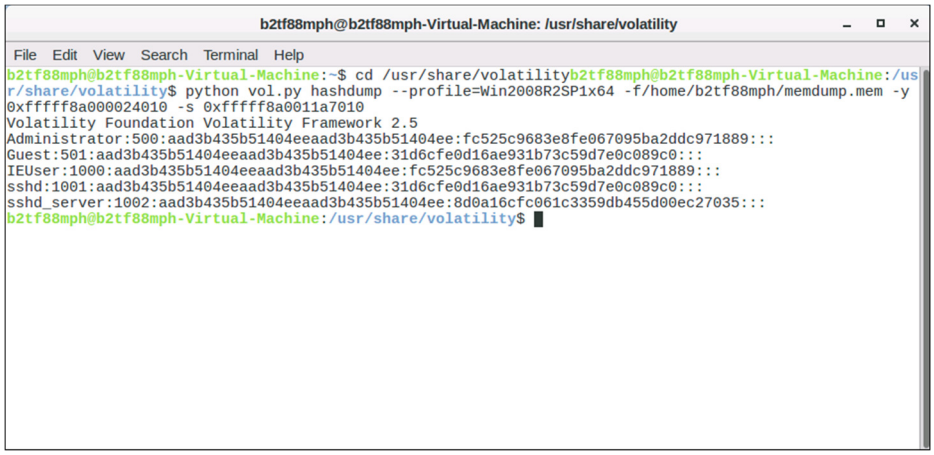


We can see the system hives, including the SAM and SYSTEM hives we are interested in, as well as their hexidecimal locations in memory.

1.10 Type `python vol.py hashdump --profile=Win2008SP1x86 -f /root/Desktop/memdump.mem-y SYSTEM -s SAM`, and press **Enter**.

Note: You will have to replace the two hexadecimal addresses with the correct virtual addresses of your hives, in this format: `-y SYSTEM -s SAM`

You can then attempt to crack the hashes in an online hash cracker such as crack station. This can be useful if an attacker has locked you out of a system but you were able to take a remote memory capture. Feel free to give it a try!



2.0 Can you locate the malware that is in memory?

Use the tools in Volatility and the provided cheat sheet to find the Malware running in memory.

Great job, you have completed LAB009!

Thank You, you may now close this module.