

## 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.

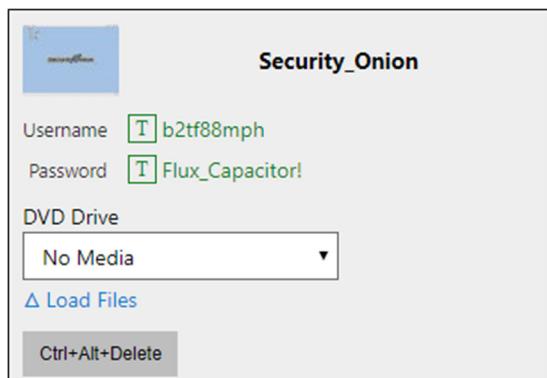
[Linux Cheat Sheet](#)

[Sans Computer Forensics Cheat Sheet](#)

[Volatility Cheat Sheet](#)

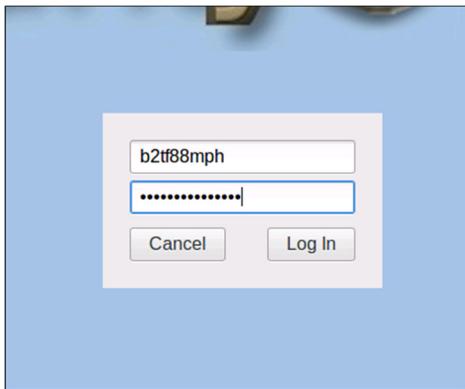
### Log in to the Lab Machine

Select the **Security\_Onion** machine on the Machines Tab.



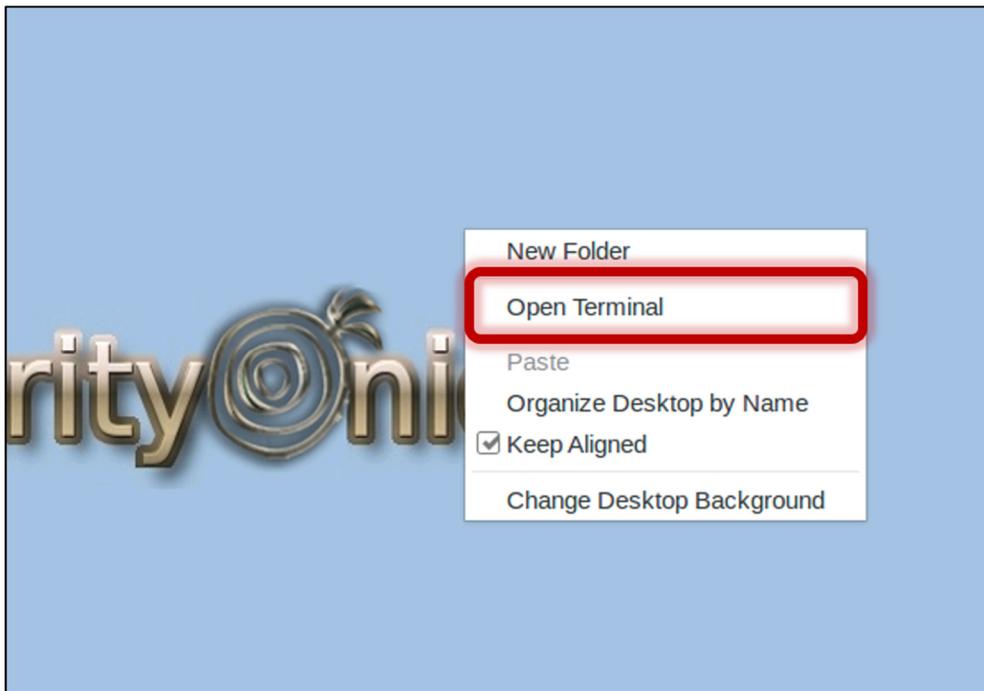


Enter the Username and Password on the **Security\_Onion** machine. Click Log In.



#### 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**.

```
b2tf88mph@b2tf88mph-Virtual-Machine:~$ cd /usr/share/volatility
b2tf88mph@b2tf88mph-Virtual-Machine:/usr/share/volatility$
```

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

```
b2tf88mph@b2tf88mph-Virtual-Machine:~$ cd /usr/share/volatility
b2tf88mph@b2tf88mph-Virtual-Machine:/usr/share/volatility$ python vol.py -h
Volatility Foundation Volatility Framework 2.5
Usage: Volatility - A memory forensics analysis platform.

Options:
-h, --help            list all available options and their default values.
                      Default values may be set in the configuration file
                      (/etc/volatilityrc)
--conf-file=/home/b2tf88mph/.volatilityrc
                      User based configuration file
-d, --debug           Debug volatility
--plugins=PLUGINS     Additional plugin directories to use (colon separated)
--info                Print information about all registered objects
--cache-directory=/home/b2tf88mph/.cache/volatility
                      Directory where cache files are stored
--cache               Use caching
--tz=TZ               Sets the (Olson) timezone for displaying timestamps
                      using pytz (if installed) or tzset
-f FILENAME, --filename=FILENAME
                      Filename to use when opening an image
--profile=WinXPSP2x86
                      Name of the profile to load (use --info to see a list
                      of supported profiles)
-l LOCATION, --location=LOCATION
```

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 10 minutes to run the analysis.



```
b2tf88mph@b2tf88mph-Virtual-Machine:~$ cd /usr/share/volatility
b2tf88mph@b2tf88mph-Virtual-Machine:/usr/share/volatility$ python vol.py imageinfo -f /home/b2tf88mph/memdump.mem
Volatility Foundation Volatility Framework 2.5
INFO   : volatility.debug    : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP0x64, Win7SP1x64, Win2008R2SP0x64, Win2008R2SP1x64 1
                  AS Layer1 : AMD64PagedMemory (Kernel AS)
                  AS Layer2 : FileAddressSpace (/home/b2tf88mph/memdump.mem)
                  PAE type : No PAE
                  DTB : 0x187000L
                  KDBG : 0xf80002a37110L
Number of Processors : 1 2
Image Type (Service Pack) : 1
KPCR for CPU 0 : 0xfffffff80002a38d00L
KUSER_SHARED_DATA : 0xfffffff78000000000L
Image date and time : 2019-03-05 21:05:06 UTC+0000 3
Image local date and time : 2019-03-05 13:05:06 -0800
b2tf88mph@b2tf88mph-Virtual-Machine:/usr/share/volatility$
```

There is a lot 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**.

```
Applications Places Terminal Wed 18:04 ━ ━ ━
b2tf88mph@b2tf88mph-Virtual-Machine:~$ cd /usr/share/volatility
b2tf88mph@b2tf88mph-Virtual-Machine:/usr/share/volatility$ python vol.py pslist --profile=Win2008R2SP1x64 -f /home/b2tf88mph/me
memdump.mem
Volatility Foundation Volatility Framework 2.5
Offset(V) 1 Name 2 PID 3 PPID 4 Thds Hnds Sess Wow64 Start 5 Exit
0xffffffa8003c6bb10 System 4 0 86 665 ----- 0 2019-03-05 19:35:40 UTC+0000
0xffffffa8004d557a0 smss.exe 264 4 2 29 ----- 0 2019-03-05 19:35:40 UTC+0000
0xffffffa8005949b10 csrss.exe 348 332 9 456 0 0 2019-03-05 19:35:45 UTC+0000
0xffffffa8005a909c0 csrss.exe 412 404 19 495 1 0 2019-03-05 19:35:48 UTC+0000
0xffffffa8005a98b10 wininit.exe 420 332 3 77 0 0 2019-03-05 19:35:48 UTC+0000
0xffffffa8005aa8b10 winlogon.exe 456 404 3 109 1 0 2019-03-05 19:35:48 UTC+0000
0xffffffa8005b2db10 services.exe 524 420 6 225 0 0 2019-03-05 19:35:50 UTC+0000
0xffffffa8005b43b10 lsass.exe 532 420 7 617 0 0 2019-03-05 19:35:51 UTC+0000
0xffffffa8005b4ab10 lsm.exe 540 420 10 149 0 0 2019-03-05 19:35:51 UTC+0000
0xffffffa8005a84060 svchost.exe 636 524 9 357 0 0 2019-03-05 19:35:58 UTC+0000
0xffffffa8005ab32a0 vmacthlp.exe 696 524 3 54 0 0 2019-03-05 19:35:59 UTC+0000
```



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.

0xffffffa8003c6bb10	System	4	0	86	665	-----	0	2019-03-05 19:35:40 UTC+0000
0xffffffa8004d557a0	smss.exe	264	4	2	29	-----	0	2019-03-05 19:35:40 UTC+0000
0xffffffa8005949b10	csrss.exe	348	332	9	456	0	0	2019-03-05 19:35:45 UTC+0000
0xffffffa8005a909c0	csrss.exe	412	404	19	495	1	0	2019-03-05 19:35:48 UTC+0000
0xffffffa8005a98b10	wininit.exe	420	332	3	77	0	0	2019-03-05 19:35:48 UTC+0000
0xffffffa8005aa8b10	winlogon.exe	456	404	3	109	1	0	2019-03-05 19:35:48 UTC+0000
0xffffffa8005b2db10	services.exe	524	420	6	225	0	0	2019-03-05 19:35:50 UTC+0000
0xffffffa8005b43b10	lsass.exe	532	420	7	617	0	0	2019-03-05 19:35:51 UTC+0000
0xffffffa8005b4ab10	lsm.exe	540	420	10	149	0	0	2019-03-05 19:35:51 UTC+0000
0xffffffa8005a84060	svchost.exe	636	524	9	357	0	0	2019-03-05 19:35:58 UTC+0000
0xffffffa8005ab32a0	vmacthl.exe	696	524	3	54	0	0	2019-03-05 19:35:59 UTC+0000
0xffffffa8005bad1d0	svchost.exe	728	524	7	263	0	0	2019-03-05 19:35:59 UTC+0000



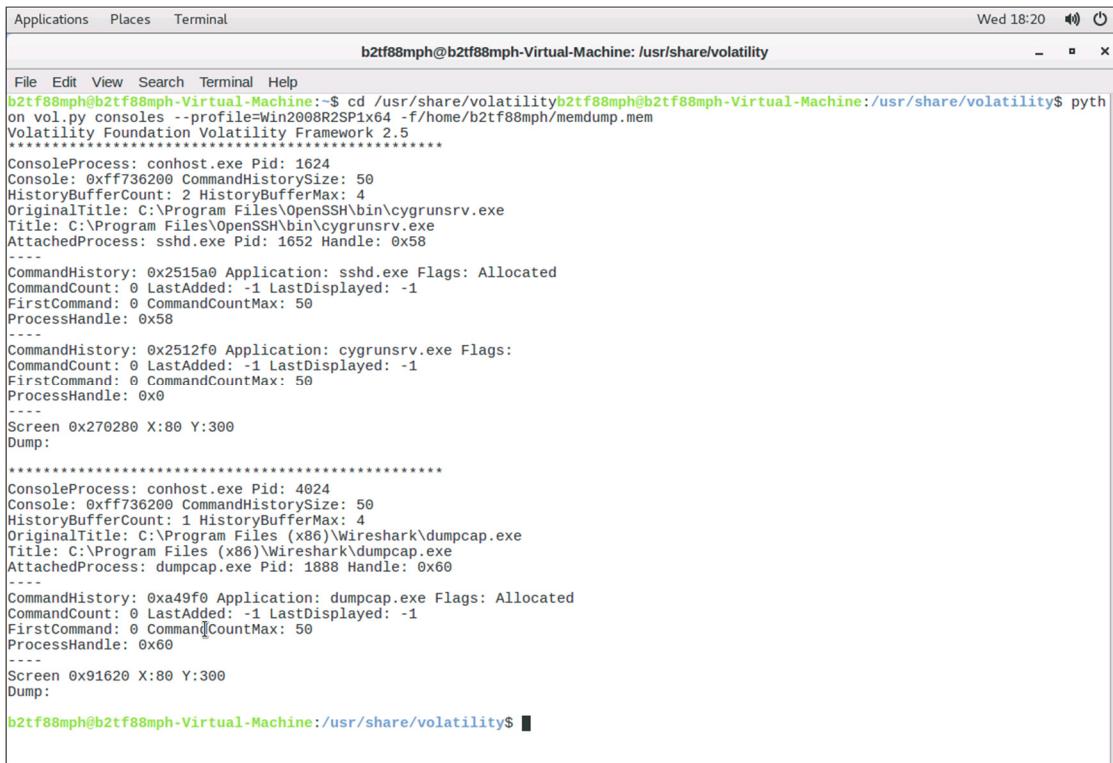
1.6 Type `python vol.py netscan --profile=Win7SP1x64 -f /home/b2tf88mph/memdump.mem`, and press **Enter**.

b2tf88mph@b2tf88mph-Virtual-Machine: /usr/share/volatility						
File	Edit	View	Search	Terminal	Help	
<pre>b2tf88mph@b2tf88mph-Virtual-Machine:~\$ cd /usr/share/volatility b2tf88mph@b2tf88mph-Virtual-Machine:/usr/share/volatility\$ python on vol.py netscan --profile=Win2008R2SP1x64 -f /home/b2tf88mph/memdump.mem Volatility Foundation Volatility Framework 2.5</pre>						
<pre>Offset(P) Proto Local Address Foreign Address State Pid Owner Creat ed 0x71dd770 UDPv4 0.0.0.0:5353 *:* 0x3-05 20:39:58 UTC+0000 0x71dd770 UDPv6 :::5353 *:* 0x3-05 20:39:58 UTC+0000 0x13cc31d80 UDPv4 0.0.0.0:3702 *:* 0x3-05 20:17:41 UTC+0000 0x13cd18640 UDPv4 0.0.0.0:3702 *:* 0x3-05 20:17:41 UTC+0000 0x13cd78a60 UDPv4 0.0.0.0:3702 *:* 0x3-05 20:17:41 UTC+0000 0x13cd78a60 UDPv6 :::3702 *:* 0x3-05 20:17:41 UTC+0000 0x13ceee710 UDPv4 0.0.0.0:57039 *:* 0x3-05 21:14:50 UTC+0000 0x13cf38ec0 UDPv6 fe80::6888:3803:90bd:40cf:546 *:* 0x3-05 21:13:25 UTC+0000 0x13cffbb60 UDPv4 0.0.0.0:3702 *:* 0x3-05 20:17:41 UTC+0000 0x13cffbb60 UDPv6 :::3702 *:* 0x3-05 20:17:41 UTC+0000 0x13d66f220 UDPv4 0.0.0.0:5355 *:* 0x3-05 21:02:34 UTC+0000 0x13d679670 UDPv4 192.168.19.133:137 *:* 0x3-05 20:17:37 UTC+0000 0x13d67a2c0 UDPv4 0.0.0.0:5355 *:* 0x3-05 21:02:34 UTC+0000 0x13d67a2c0 UDPv6 :::5355 *:* 0x3-05 21:02:34 UTC+0000 0x13d67e2c0 UDPv4 192.168.19.133:138 *:* 0x3-05 20:17:37 UTC+0000 0x13d628a00 TCPv4 0.0.0.0:22 0.0.0.0:0 LISTENING 1652 sshd.exe 0x13d630950 TCPv4 0.0.0.0:22 0.0.0.0:0 LISTENING 1652 sshd.exe 0x13d630950 TCPv6 :::22 :::0 LISTENING 1652 sshd.exe 0x13cf46010 TCPv4 192.168.19.133:56203 104.100.225.43:443 CLOSE_WAIT -1 0x13cf96010 TCPv4 192.168.19.133:55976 184.26.40.69:443 ESTABLISHED -1 0x13d688b80 TCPv4 192.168.19.133:56408 104.244.38.20:443 CLOSED -1 0x13db60530 UDPv4 0.0.0.0:0 *:* 0x3-05 19:36:15 UTC+0000 0x13db63910 UDPv4 0.0.0.0:0 *:* 0x3-05 19:36:15 UTC+0000 0x13db63910 UDPv6 :::0 *:*</pre>						
<pre>2168 svhost.exe 2019- 2168 svhost.exe 2019-</pre>						

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

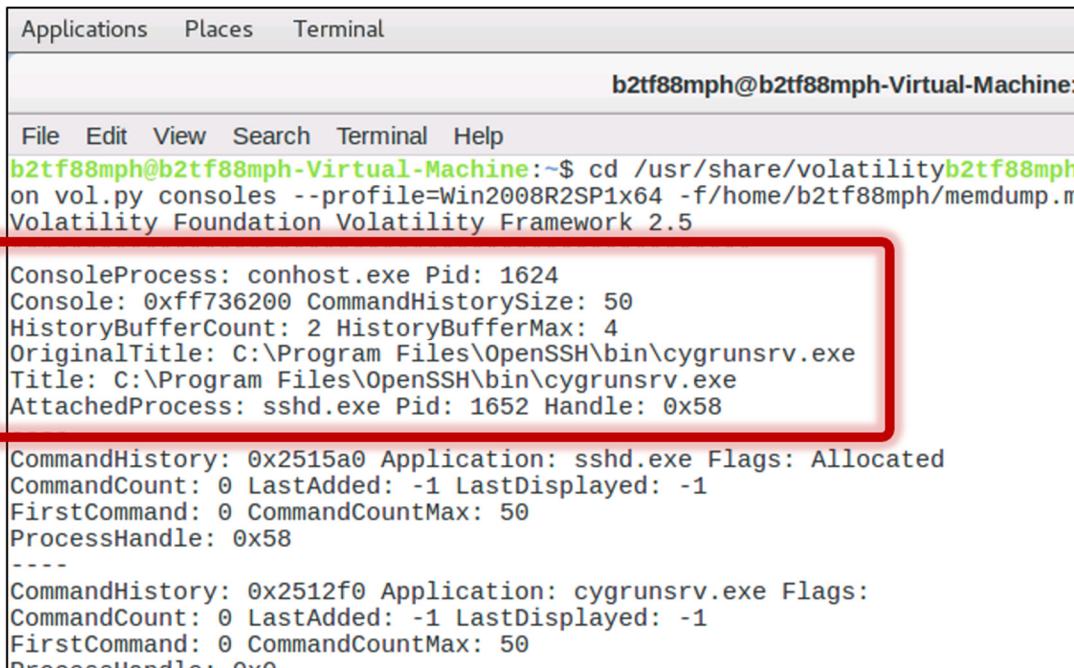


1.7 Type `python vol.py consoles --profile=Win2008SP1x86 -f/home/b2tf88mph/memdump.mem`, and press **Enter**.



```
Applications Places Terminal Wed 18:20
b2tf88mph@b2tf88mph-Virtual-Machine: /usr/share/volatility
File Edit View Search Terminal Help
b2tf88mph@b2tf88mph-Virtual-Machine:~$ cd /usr/share/volatility
b2tf88mph@b2tf88mph-Virtual-Machine: /usr/share/volatility$ python vol.py consoles --profile=Win2008R2SP1x64 -f /home/b2tf88mph/memdump.mem
Volatility Foundation Volatility Framework 2.5
*****
ConsoleProcess: conhost.exe Pid: 1624
Console: 0xff736200 CommandHistorySize: 50
HistoryBufferCount: 2 HistoryBufferMax: 4
OriginalTitle: C:\Program Files\OpenSSH\bin\cygrunsrv.exe
Title: C:\Program Files\OpenSSH\bin\cygrunsrv.exe
AttachedProcess: sshd.exe Pid: 1652 Handle: 0x58
----
CommandHistory: 0x2515a0 Application: sshd.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x58
----
CommandHistory: 0x2512f0 Application: cygrunsrv.exe Flags:
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x0
----
Screen 0x270280 X:80 Y:300
Dump:
*****
ConsoleProcess: conhost.exe Pid: 4024
Console: 0xff736200 CommandHistorySize: 50
HistoryBufferCount: 1 HistoryBufferMax: 4
OriginalTitle: C:\Program Files (x86)\Wireshark\dumpcap.exe
Title: C:\Program Files (x86)\Wireshark\dumpcap.exe
AttachedProcess: dumpcap.exe Pid: 1888 Handle: 0x60
----
CommandHistory: 0xa49f0 Application: dumpcap.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x60
----
Screen 0x91620 X:80 Y:300
Dump:
b2tf88mph@b2tf88mph-Virtual-Machine: /usr/share/volatility$
```

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.



```
Applications Places Terminal
b2tf88mph@b2tf88mph-Virtual-Machine:
File Edit View Search Terminal Help
b2tf88mph@b2tf88mph-Virtual-Machine:~$ cd /usr/share/volatility
b2tf88mph@b2tf88mph-Virtual-Machine: /usr/share/volatility$ python vol.py consoles --profile=Win2008R2SP1x64 -f /home/b2tf88mph/memdump.mem
Volatility Foundation Volatility Framework 2.5
*****
ConsoleProcess: conhost.exe Pid: 1624
Console: 0xff736200 CommandHistorySize: 50
HistoryBufferCount: 2 HistoryBufferMax: 4
OriginalTitle: C:\Program Files\OpenSSH\bin\cygrunsrv.exe
Title: C:\Program Files\OpenSSH\bin\cygrunsrv.exe
AttachedProcess: sshd.exe Pid: 1652 Handle: 0x58
CommandHistory: 0x2515a0 Application: sshd.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x58
CommandHistory: 0x2512f0 Application: cygrunsrv.exe Flags:
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x0
```



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**.

```
Applications Places Terminal Wed 18:57
b2tf88mph@b2tf88mph-Virtual-Machine: /usr/share/volatility
File Edit View Search Terminal Help
b2tf88mph@b2tf88mph-Virtual-Machine:~$ cd /usr/share/volatility
b2tf88mph@b2tf88mph-Virtual-Machine:/usr/share/volatility$ python
on vol.py svcscan --profile=Win2008R2SP1x64 -f/home/b2tf88mph/memdump.mem
Volatility Foundation Volatility Framework 2.5
Offset: 0xcccc80
Order: 361
Start: SERVICE_AUTO_START
Process ID: 696
Service Name: VMware Physical Disk Helper Service
Display Name: VMware Physical Disk Helper Service
Service Type: SERVICE_WIN32_OWN_PROCESS
Service State: SERVICE_RUNNING
Binary Path: "C:\Program Files\VMware\VMware Tools\vmacthlp.exe"

Offset: 0xcccc240
Order: 360
Start: SERVICE_DEMAND_START
Process ID: -
Service Name: vmvss
Display Name: VMware Snapshot Provider
Service Type: SERVICE_WIN32_OWN_PROCESS
Service State: SERVICE_STOPPED
Binary Path: -

Offset: 0xcccc150
Order: 359
Start: SERVICE_AUTO_START
Process ID: 1584
Service Name: VMTools
Display Name: VMware Tools
Service Type: SERVICE_WIN32_OWN_PROCESS
Service State: SERVICE_RUNNING
Binary Path: "C:\Program Files\VMware\VMware Tools\vmtoolsd.exe"

Offset: 0xcccc060
Order: 358
Start: SERVICE_SYSTEM_START
Process ID: -
Service Name: vmrawdsk
Display Name: VMware Physical Disk Helper
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_RUNNING
Binary Path: \Driver\vmrawdsk

Offset: 0xcccbf70
Order: 357
Start: SERVICE_DEMAND_START
```

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**.



```
b2tf88mph@b2tf88mph-Virtual-Machine:~$ cd /usr/share/volatility
b2tf88mph@b2tf88mph-Virtual-Machine:/usr/share/volatility$ python vol.py hivelist --profile=Win2008R2SP1x64 -f /home/b2tf88mph/memdump.mem
Volatility Foundation Volatility Framework 2.5
Virtual Physical Name
-----
0xffffffff8a000024010 0x00000000a972e010 \REGISTRY\MACHINE\SYSTEM
0xffffffff8a00005cb010 0x00000000a657f010 \Device\HarddiskVolume1\Boot\BCD
0xffffffff8a0000f4d010 0x00000000a5daa010 \SystemRoot\System32\Config\SOFTWARE
0xffffffff8a0011a7010 0x00000000a7c10 \SystemRoot\System32\Config\SECURITY
0xffffffff8a0011a7010 0x0000000097bb6010 \SystemRoot\System32\Config\SAM
0xffffffff8a0012d2010 0x00000000996430010 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0xffffffff8a001585010 0x0000000092ce7010 \??\C:\Users\sshd_server\ntuser.dat
0xffffffff8a001623010 0x0000000092baa010 \??\C:\Users\sshd_server\AppData\Local\Microsoft\Windows\UsrClass.dat
0xffffffff8a001784010 0x00000000910de010 \??\C:\Users\IEUser\ntuser.dat
0xffffffff8a0019f7010 0x0000000088b3d7010 \??\C:\Users\IEUser\AppData\Local\Microsoft\Windows\UsrClass.dat
0xffffffff8a002ea8010 0x00000000001e27d010 \??\C:\System Volume Information\Syscache.hve
0xffffffff8a004c71010 0x00000000a56ac010 \SystemRoot\System32\Config\DEFAULT
b2tf88mph@b2tf88mph-Virtual-Machine:/usr/share/volatility$
```

We can see the system hives, including the SAM and SYSTEM hives we are interested in, as well as their hexadecimal 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!

```
b2tf88mph@b2tf88mph-Virtual-Machine:~$ cd /usr/share/volatility
b2tf88mph@b2tf88mph-Virtual-Machine:/usr/share/volatility$ python vol.py hashdump --profile=Win2008R2SP1x64 -f /home/b2tf88mph/memdump.mem -y
0xffffffff8a000024010 -s 0xffffffff8a0011a7010
Volatility Foundation Volatility Framework 2.5
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfed16ae931b73c59d7e0c089c0:::
IEUser:1000:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
ssh0:1001:aad3b435b51404eeaad3b435b51404ee:31d6cfed16ae931b73c59d7e0c089c0:::
sshd_server:1002:aad3b435b51404eeaad3b435b51404ee:8d0a16fc061c3359db455d00ec27035:::
b2tf88mph@b2tf88mph-Virtual-Machine:/usr/share/volatility$
```

## 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.