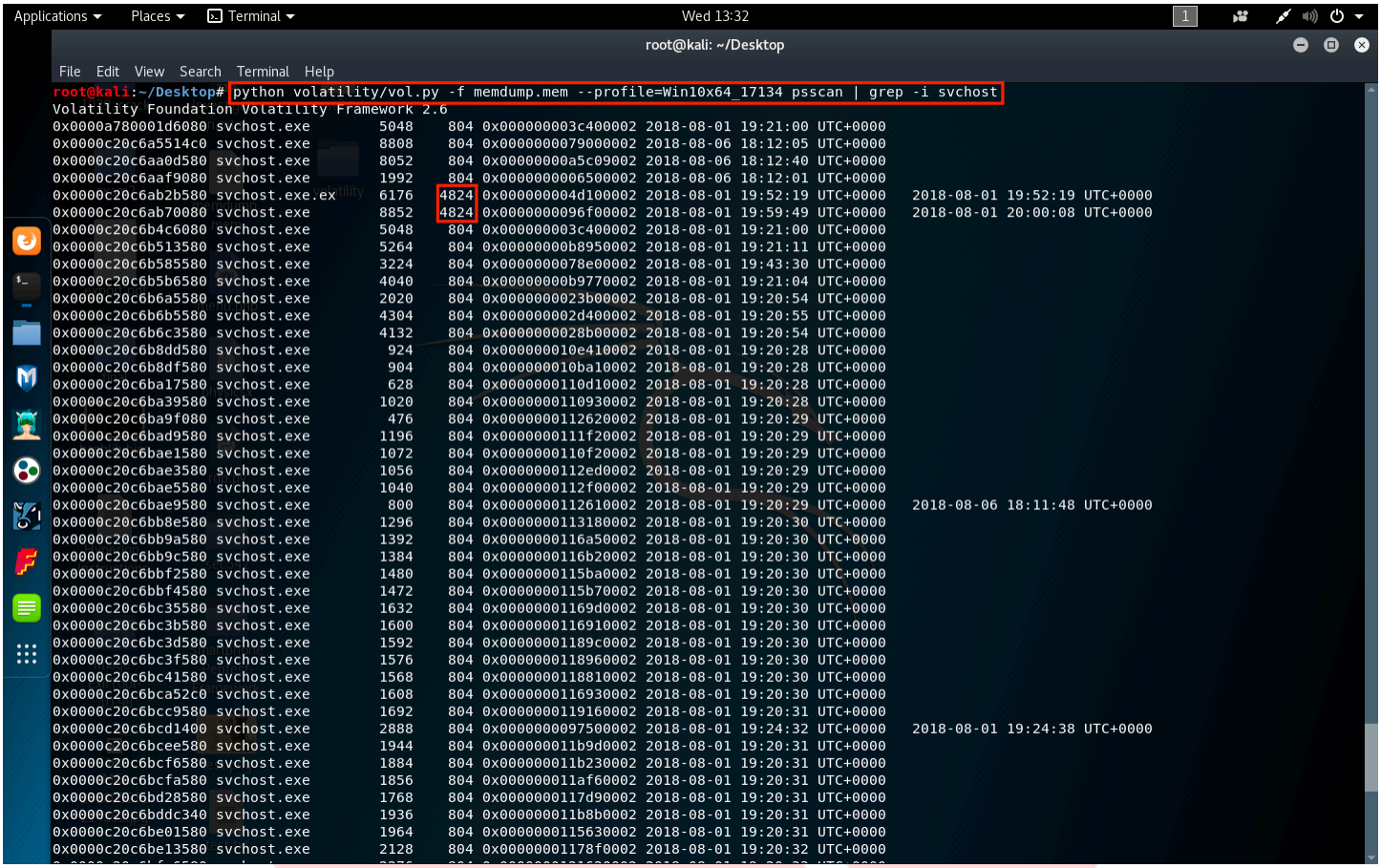
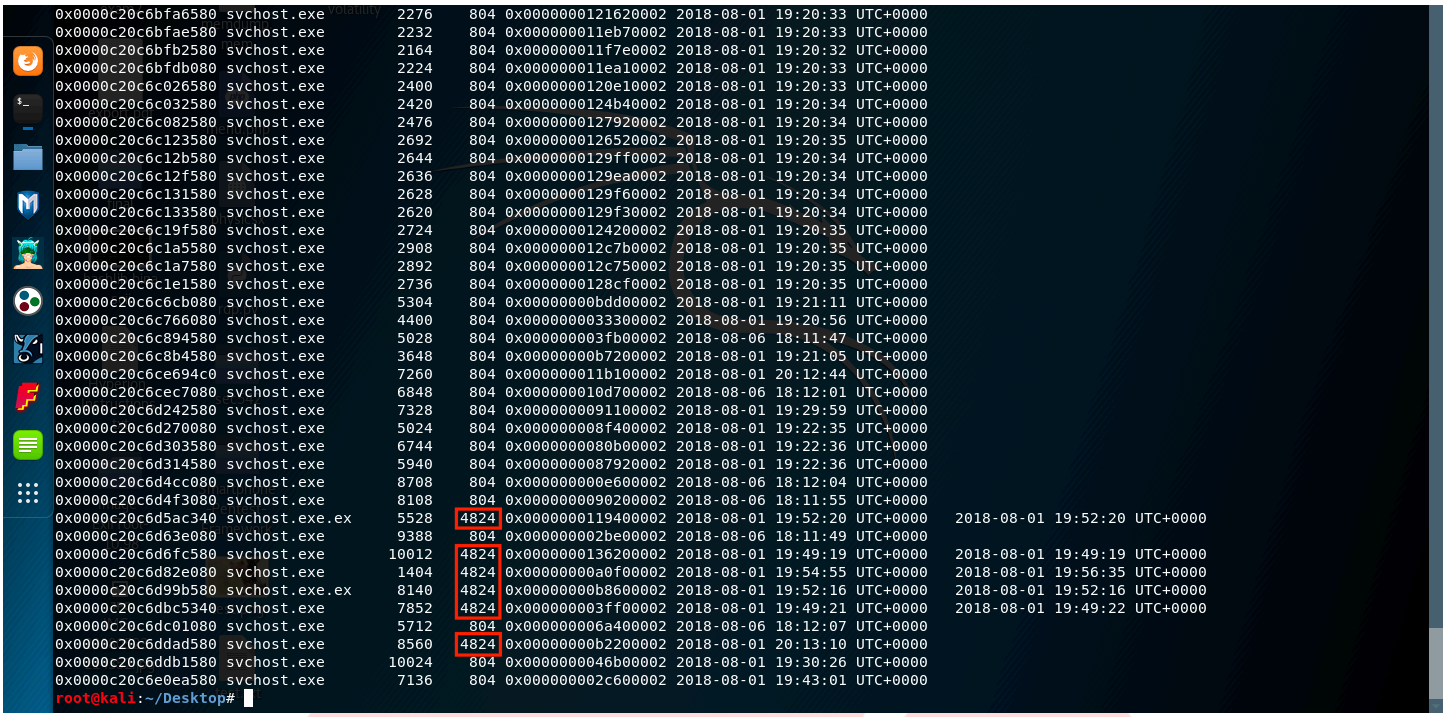
Mini Memory CTF
Solutions Guide

Question #1
Find the running rogue (malicious) process. The flag is the MD5 hash of its PID.

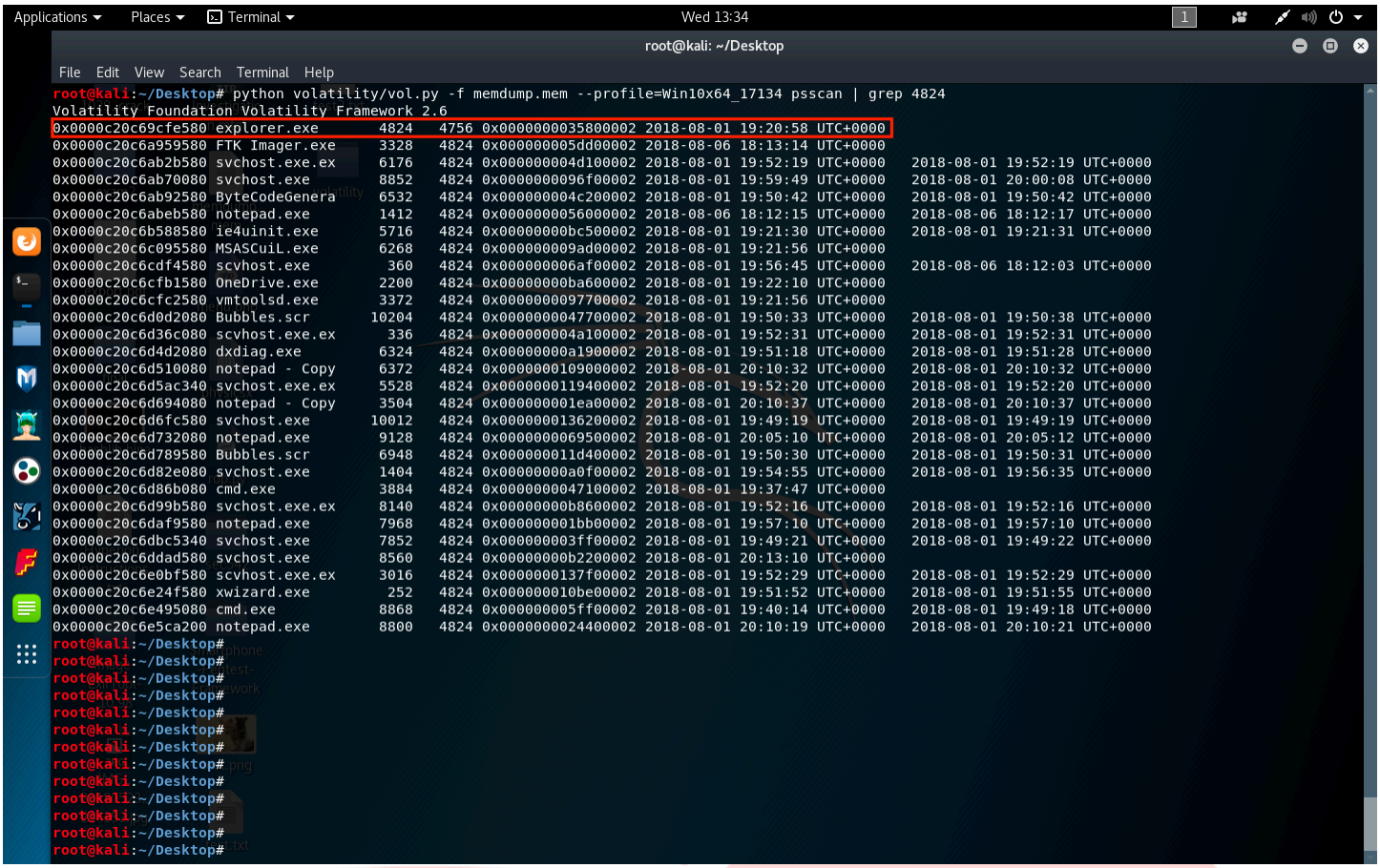
The ubiquitous Windows **svchost.exe** process is a favorite of malware authors. A normal system will have numerous **svchost.exe** processes running at any given time; therefore, it is common for malware to hide amongst these legitimate processes in an effort to blend in and appear normal. The real **svchost.exe** process will have a **parent** of **services.exe**, will reside on disk in **%SYSTEMROOT%\System32**, and will have one or more **-k** parameters present.





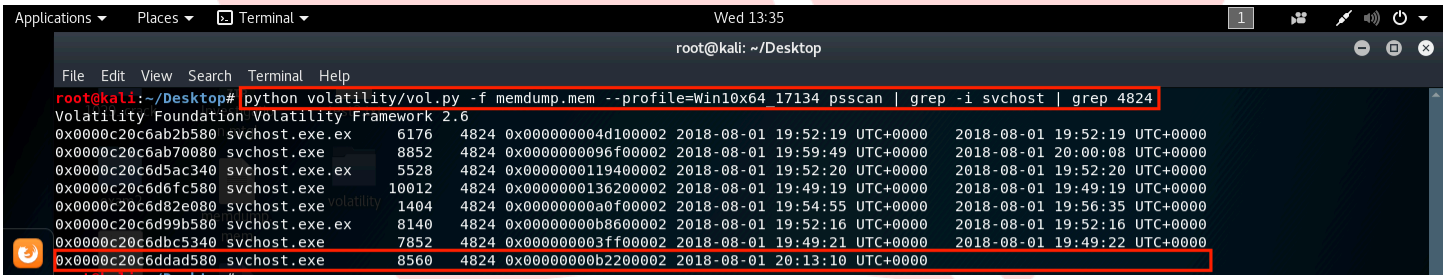
In the screen shots above, we're using Volatility with the **Win10x64_17134** profile to analyze the provided memory image. We've used the **psscan** plugin, which is similar to **pslist** but will show unlinked and hidden processes. We are filtering the results to look for the string **svchost**, and we can immediately see that the majority of the processes have a Parent PID (PPID) of 804. You'll note, however, that several of the processes have a different parent. This is a red flag and warrants further investigation.

We can use **pstree**, **pslist**, or **psscan** to verify that **PID 804** is indeed **services.exe**, which is expected (not shown). But, what about **PID 4824**?



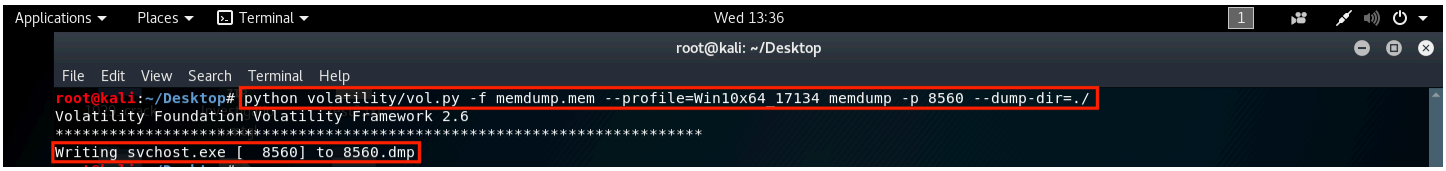
According to the output above, it appears **PID 4824** is **explorer.exe**, which has no business executing **svchost.exe**!

Now, let's focus on all the **svchost.exe** processes which have a **PPID** of **4824**.

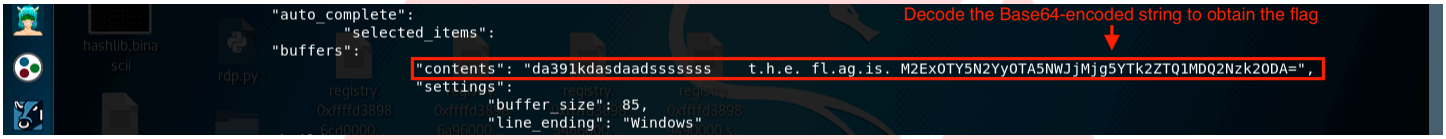


As you can see from the output above, only one process is still active in memory, and that is **PID 8560**. The MD5 hash of **8560** is **bc05ca60f2f0d67d0525f41d1d8f8717** and is the answer to **Question #1**.

Question #2
Find the running rogue (malicious) process and dump its memory to disk. You'll find the 32-character flag within that process's memory.



As shown in the screen shot above, we've used the **memdump** plugin to accomplish this, and the resulting process memory has been written to a file entitled **8560.dmp**. If we run **strings** against this file and look for anything interesting, we'll come across this:



The string displayed is a Base64-encoded version of the flag. If we decode it, we'll be left with **3a19697f29095bc289a96e4504679680**, which is the answer to **Question #2**.

Question #3
What is the MAC address of this machine's default gateway? The flag is the MD5 hash of that MAC address in uppercase with dashes (-) as delimiters. Example: 01-00-A4-FB-AF-C2.

You'll recall from traditional disk-based forensics that such information is available in the **NetworkList** key, which is located in the **SOFTWARE** registry hive.

There is more than one way to find this information. We can use the Volatility **printkey** plugin to print the known location of the **NetworkList** key:

```
python volatility/vol.py -f memdump.mem --profile=Win10x64_17134 printkey -K
"Microsoft\Windows NT\CurrentVersion\NetworkList\Signatures\Unmanaged"
```

This output would show us the subkey that contains the value we're after. In this case, it's "**010103000F0000F0080000000F0000F0E3E937A4D0CD0A314266D2986CB7DED5D8B43B828FEEDCEFFD6DE7141DC1D15D**." Now, repeat the Volatility command, adding this subkey:

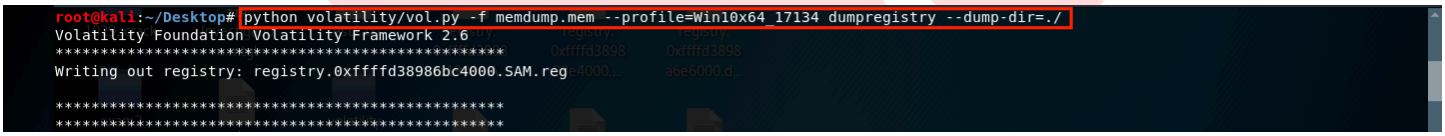
```
python volatility/vol.py -f memdump.mem --profile=Win10x64_17134 printkey -K
"Microsoft\Windows NT\CurrentVersion\NetworkList\Signatures\Unmanaged\
010103000F0000F0080000000F0000F0E3E937A4D0CD0A314266D2986CB7DED5D8B43B828FE
EDCEFFD6DE7141DC1D15D"
```

You will see the output below:

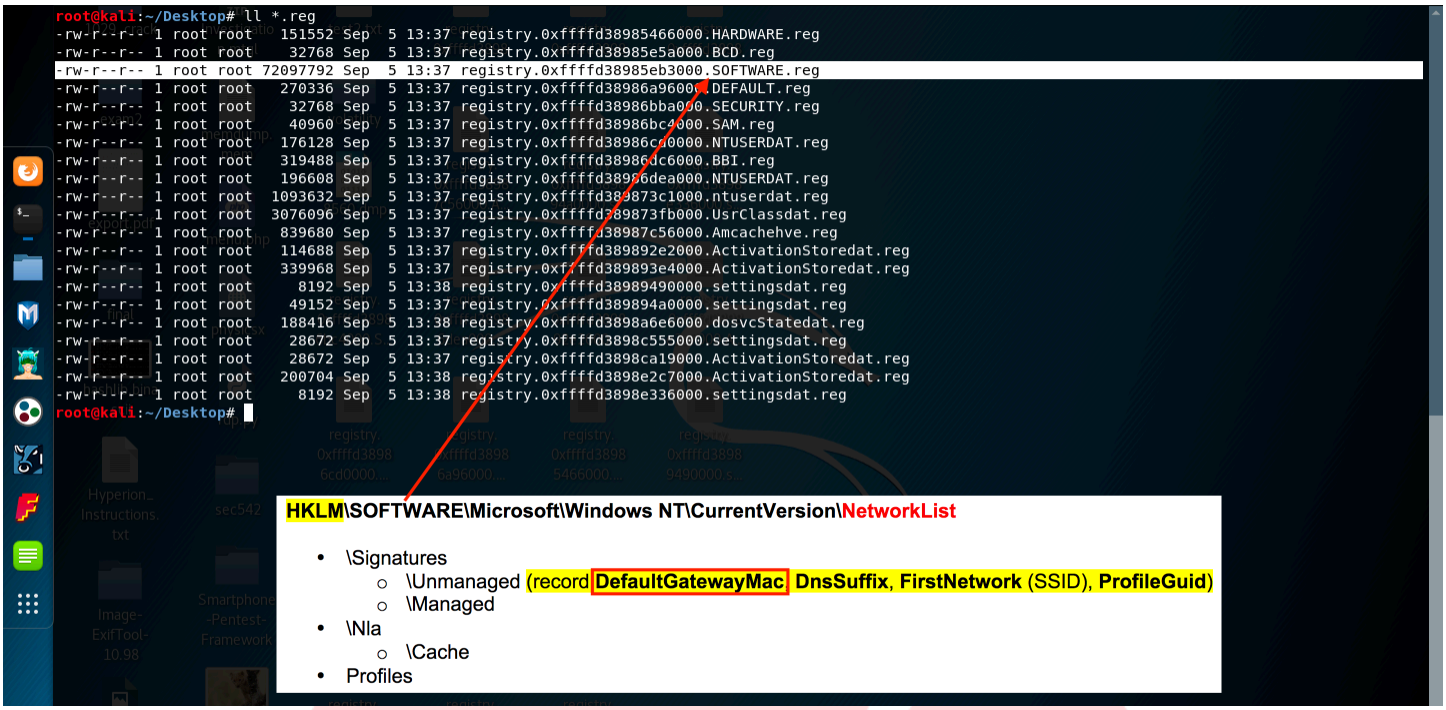
REG_BINARY **DefaultGatewayMac** : (S)
0x00000000 **00 50 56 fe d8 07**

Convert the MAC address to UPPERCASE and add dashes (-) as delimiters. The MD5 hash of this value (**00-50-56-FE-D8-07**) is **6496d43b622a2ad241b4d08699320f4e** and is the answer to **Question #3**.

Alternatively, we can use Volatility's **dumpregistry** plugin to extract registry hives from memory and write them to disk for further processing and analysis, as shown below:



(Output Truncated)



Now use a program such as **RegRipper** to parse the contents:

rip.pl -r registry.0xffffd38985eb3000.SOFTWARE.reg -f software > out

Search for "**DefaultGatewayMac**" within the output. As in the first method above, you'll find that the gateway's MAC address is **00-50-56-FE-D8-07** (as displayed in UPPERCASE, with dashes as delimiters). The MD5 hash of this value is **6496d43b622a2ad241b4d08699320f4e** and is a second way to obtain the answer to **Question #3**.

Question #4
Find the full path of the browser cache created when an analyst visited "www.13cubed.com." The path will begin with "Users\." Convert the path to uppercase. The flag is the MD5 hash of that string.

To answer the final question, we'll need to utilize an artifact that can provide us with full file paths for disk-based content. It turns out that Volatility provides a plugin called **mftparser**, which will scan for and parse entries in the Windows NTFS Master File Table (MFT).

```
root@kali:~/Desktop# python volatility/vol.py -f memdump.mem --profile=Win10x64_17134 mftparser > out
Volatility Foundation Volatility Framework 2.6
```

```
$FILE_NAME
Creation                       Modified                       MFT Altered                    Access Date                    Name/Path
------------------------------ ------------------------------ ------------------------------ ------------------------------ ---------
2018-08-01 19:29:27 UTC+0000 2018-08-01 19:29:27 UTC+0000   2018-08-01 19:29:27 UTC+0000   2018-08-01 19:29:27 UTC+0000   Users\CTF\AppData\Local\Packag
es\MICROS~1.MIC\AC\#!001\MICROS~1\Cache\AHF2COV9\13CUBE~1.HTM

$FILE_NAME
Creation                       Modified                       MFT Altered                    Access Date                    Name/Path
------------------------------ ------------------------------ ------------------------------ ------------------------------ ---------
2018-08-01 19:29:27 UTC+0000 2018-08-01 19:29:27 UTC+0000   2018-08-01 19:29:27 UTC+0000   2018-08-01 19:29:27 UTC+0000   Users\CTF\AppData\Local\Packag
es\MICROS~1.MIC\AC\#!001\MICROS~1\Cache\AHF2COV9\13cubed[1].htm
```

The uppercase path of the file is **USERS\CTF\APPDATA\LOCAL\PACKAGES\MICROS~1.MIC\AC\#!001\MICROS~1\CACHE\AHF2COV9\13CUBED[1].HTM**, and the MD5 hash of this value is **b5bdd048030cd26ab2d0e7f7e351224d**, which is the answer to **Question #4**.