



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.

```
Applications ▾ Places ▾ Terminal ▾ Wed 13:32
root@kali: ~/Desktop#
File Edit View Search Terminal Help
root@kali:~/Desktop# python volatility/vol.py -f memdump.mem --profile=Win10x64 17134 psscan | grep -i svchost
Volatility Foundation Volatility Framework 2.6
0x0000a780001d6080 svchost.exe      5048    804 0x000000003c40002 2018-08-01 19:21:00 UTC+0000
0x0000c20c6a5514c0 svchost.exe      8808    804 0x000000007900002 2018-08-06 18:12:05 UTC+0000
0x0000c20c6aa0d580 svchost.exe      8052    804 0x00000000a5c90002 2018-08-06 18:12:40 UTC+0000
0x0000c20c6aaaf9080 svchost.exe      1992    804 0x0000000006500002 2018-08-06 18:12:01 UTC+0000
0x0000c20c6ab2b580 svchost.exe.exe  6176    4824 0x0000000004d10002 2018-08-01 19:52:19 UTC+0000
0x0000c20c6ab70080 svchost.exe      8852    4824 0x00000000096f00002 2018-08-01 19:59:45 UTC+0000 2018-08-01 19:52:19 UTC+0000
0x0000c20c6b4c6080 svchost.exe      5048    804 0x000000003c40002 2018-08-01 19:21:00 UTC+0000 2018-08-01 20:00:08 UTC+0000
0x0000c20c6b513580 svchost.exe      5264    804 0x00000000b8950002 2018-08-01 19:21:11 UTC+0000
0x0000c20c6b585580 svchost.exe      3224    804 0x0000000078e00002 2018-08-01 19:43:30 UTC+0000
0x0000c20c6b5b6580 svchost.exe      4040    804 0x00000000b9770002 2018-08-01 19:21:04 UTC+0000
0x0000c20c6b6a5580 svchost.exe      2020    804 0x00000000023b00002 2018-08-01 19:20:54 UTC+0000
0x0000c20c6b6b5580 svchost.exe      4304    804 0x0000000002d400002 2018-08-01 19:20:55 UTC+0000
0x0000c20c6b6c3580 svchost.exe      4132    804 0x00000000028b00002 2018-08-01 19:20:54 UTC+0000
0x0000c20c6b8dd580 svchost.exe      924     804 0x0000000010e410002 2018-08-01 19:20:28 UTC+0000
0x0000c20c6b8df580 svchost.exe      904     804 0x0000000010ba10002 2018-08-01 19:20:28 UTC+0000
0x0000c20c6ba17580 svchost.exe      628     804 0x00000000110d10002 2018-08-01 19:20:28 UTC+0000
0x0000c20c6ba239580 svchost.exe      1020    804 0x00000000110930002 2018-08-01 19:20:28 UTC+0000
0x0000c20c6ba5f080 svchost.exe      476     804 0x00000000112620002 2018-08-01 19:20:29 UTC+0000
0x0000c20c6bad9580 svchost.exe      1196    804 0x00000000111f20002 2018-08-01 19:20:29 UTC+0000
0x0000c20c6bae1580 svchost.exe      1072    804 0x00000000110f20002 2018-08-01 19:20:29 UTC+0000
0x0000c20c6bae3580 svchost.exe      1056    804 0x00000000112ed0002 2018-08-01 19:20:29 UTC+0000
0x0000c20c6bae5580 svchost.exe      1040    804 0x00000000112f00002 2018-08-01 19:20:29 UTC+0000
0x0000c20c6bae9580 svchost.exe      800     804 0x00000000112610002 2018-08-01 19:20:29 UTC+0000 2018-08-06 18:11:48 UTC+0000
0x0000c20c6bb8e580 svchost.exe      1296    804 0x00000000113180002 2018-08-01 19:20:30 UTC+0000
0x0000c20c6bb9a580 svchost.exe      1392    804 0x00000000116a50002 2018-08-01 19:20:30 UTC+0000
0x0000c20c6bb9c580 svchost.exe      1384    804 0x00000000116b20002 2018-08-01 19:20:30 UTC+0000
0x0000c20c6bbf2580 svchost.exe      1480    804 0x00000000115ba0002 2018-08-01 19:20:30 UTC+0000
0x0000c20c6bbf4580 svchost.exe      1472    804 0x00000000115b70002 2018-08-01 19:20:30 UTC+0000
0x0000c20c6c35580 svchost.exe      1632    804 0x000000001169d0002 2018-08-01 19:20:30 UTC+0000
0x0000c20c6c3b580 svchost.exe      1600    804 0x00000000116910002 2018-08-01 19:20:30 UTC+0000
0x0000c20c6c3d580 svchost.exe      1592    804 0x000000001189c0002 2018-08-01 19:20:30 UTC+0000
0x0000c20c6c3f580 svchost.exe      1576    804 0x00000000118960002 2018-08-01 19:20:30 UTC+0000
0x0000c20c6c41580 svchost.exe      1568    804 0x00000000118810002 2018-08-01 19:20:30 UTC+0000
0x0000c20c6ca52c0 svchost.exe      1608    804 0x00000000116930002 2018-08-01 19:20:30 UTC+0000
0x0000c20c6cc9580 svchost.exe      1692    804 0x00000000119160002 2018-08-01 19:20:31 UTC+0000
0x0000c20c6cd1400 svchost.exe      2888    804 0x00000000097500002 2018-08-01 19:24:32 UTC+0000 2018-08-01 19:24:38 UTC+0000
0x0000c20c6cce580 svchost.exe      1944    804 0x0000000011b9d0002 2018-08-01 19:20:31 UTC+0000
0x0000c20c6cf6580 svchost.exe      1884    804 0x0000000011b230002 2018-08-01 19:20:31 UTC+0000
0x0000c20c6cbfc580 svchost.exe      1856    804 0x0000000011af60002 2018-08-01 19:20:31 UTC+0000
0x0000c20c6bd28580 svchost.exe      1768    804 0x00000000117d90002 2018-08-01 19:20:31 UTC+0000
0x0000c20c6bddc340 svchost.exe      1936    804 0x0000000011bb0b0002 2018-08-01 19:20:31 UTC+0000
0x0000c20c6be01580 svchost.exe      1964    804 0x00000000115630002 2018-08-01 19:20:31 UTC+0000
0x0000c20c6be13580 svchost.exe      2128    804 0x000000001178f0002 2018-08-01 19:20:32 UTC+0000
```

		Volatility	PID	PPID	Address	Date	Time	UTC
0x0000c20c6bfa6580	svchost.exe	2276	804	0x0000000121620002	2018-08-01	19:20:33	UTC+0000	
0x0000c20c6bfae580	svchost.exe	2232	804	0x000000011eb70002	2018-08-01	19:20:33	UTC+0000	
0x0000c20c6fb2580	svchost.exe	2164	804	0x000000011f7e0002	2018-08-01	19:20:32	UTC+0000	
0x0000c20c6fdb080	svchost.exe	2224	804	0x000000011ea10002	2018-08-01	19:20:33	UTC+0000	
0x0000c20c6c026580	svchost.exe	2400	804	0x0000000120e10002	2018-08-01	19:20:33	UTC+0000	
0x0000c20c6c032580	svchost.exe	2420	804	0x0000000124b40002	2018-08-01	19:20:34	UTC+0000	
0x0000c20c6c082580	svchost.exe	2476	804	0x0000000127920002	2018-08-01	19:20:34	UTC+0000	
0x0000c20c6c123580	svchost.exe	2692	804	0x0000000126520002	2018-08-01	19:20:35	UTC+0000	
0x0000c20c6c12b580	svchost.exe	2644	804	0x0000000129ff0002	2018-08-01	19:20:34	UTC+0000	
0x0000c20c6c12f580	svchost.exe	2636	804	0x0000000129ea0002	2018-08-01	19:20:34	UTC+0000	
0x0000c20c6c131580	svchost.exe	2628	804	0x0000000129f60002	2018-08-01	19:20:34	UTC+0000	
0x0000c20c6c133580	svchost.exe	2620	804	0x0000000129f30002	2018-08-01	19:20:34	UTC+0000	
0x0000c20c6c19f580	svchost.exe	2724	804	0x0000000124200002	2018-08-01	19:20:35	UTC+0000	
0x0000c20c6c1a5580	svchost.exe	2908	804	0x000000012c7b0002	2018-08-01	19:20:35	UTC+0000	
0x0000c20c6c1a7580	svchost.exe	2892	804	0x000000012c750002	2018-08-01	19:20:35	UTC+0000	
0x0000c20c6c1e1580	svchost.exe	2736	804	0x0000000128cf0002	2018-08-01	19:20:35	UTC+0000	
0x0000c20c6c6cb080	svchost.exe	5304	804	0x00000000bd000002	2018-08-01	19:21:11	UTC+0000	
0x0000c20c6c766080	svchost.exe	4400	804	0x0000000033300002	2018-08-01	19:20:56	UTC+0000	
0x0000c20c6c894580	svchost.exe	5028	804	0x000000003fb00002	2018-08-06	18:11:47	UTC+0000	
0x0000c20c6c8b04580	svchost.exe	3648	804	0x0000000067200002	2018-08-01	19:21:05	UTC+0000	
0x0000c20c6ce694c0	svchost.exe	7260	804	0x0000000011b100002	2018-08-01	20:12:44	UTC+0000	
0x0000c20c6cec7080	svchost.exe	6848	804	0x0000000010d700002	2018-08-06	18:12:01	UTC+0000	
0x0000c20c6d242580	svchost.exe	7328	804	0x00000000091100002	2018-08-01	19:29:59	UTC+0000	
0x0000c20c6d270080	svchost.exe	5024	804	0x000000008f400002	2018-08-01	19:22:35	UTC+0000	
0x0000c20c6d303580	svchost.exe	6744	804	0x0000000088b00002	2018-08-01	19:22:36	UTC+0000	
0x0000c20c6d314580	svchost.exe	5940	804	0x0000000087920002	2018-08-01	19:22:36	UTC+0000	
0x0000c20c6d4cc080	svchost.exe	8708	804	0x0000000006000002	2018-08-06	18:12:04	UTC+0000	
0x0000c20c6d4f3080	svchost.exe	8108	804	0x00000000090200002	2018-08-06	18:11:55	UTC+0000	
0x0000c20c6d5ac340	svchost.exe.ex	5528	4824	0x0000000019400002	2018-08-01	19:52:20	UTC+0000	
0x0000c20c6d63e080	svchost.exe	9388	804	0x000000002be00002	2018-08-06	18:11:49	UTC+0000	
0x0000c20c6d6fc580	svchost.exe	10012	4824	0x0000000136200002	2018-08-01	19:49:19	UTC+0000	
0x0000c20c6d82e080	svchost.exe	1404	4824	0x00000000aaef0002	2018-08-01	19:54:55	UTC+0000	
0x0000c20c6d99b580	svchost.exe	8140	4824	0x000000000b860002	2018-08-01	19:52:16	UTC+0000	
0x0000c20c6dbc5340	svchost.exe	7852	4824	0x0000000003ff00002	2018-08-01	19:49:21	UTC+0000	
0x0000c20c6dc01080	svchost.exe	5712	804	0x000000006a400002	2018-08-06	18:12:07	UTC+0000	
0x0000c20c6ddad580	svchost.exe	8560	4824	0x000000000b2200002	2018-08-01	20:13:10	UTC+0000	
0x0000c20c6ddb1580	svchost.exe	10024	804	0x00000000046b00002	2018-08-01	19:30:26	UTC+0000	
0x0000c20c6e0ea580	svchost.exe	7136	804	0x0000000002c600002	2018-08-01	19:43:01	UTC+0000	

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**!

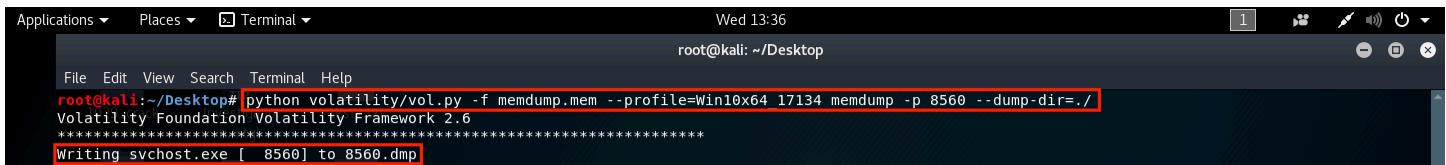
Now, let's focus on all the **svchost.exe** processes which have a **PPID** of **4824**.

```
Applications ▾ Places ▾ Terminal ▾ Wed 13:35
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# python volatility/vol.py -f memdump.mem --profile=Win10x64 17134 psscan | grep -i svchost | grep 4824
Volatility Foundation Volatility Framework 2.6
0x00000c20c6ab2b580 svchost.exe.ex 6176 4824 0x000000004d100002 2018-08-01 19:52:19 UTC+0000 2018-08-01 19:52:19 UTC+0000
0x00000c20c6ab70080 svchost.exe 8852 4824 0x0000000096f00002 2018-08-01 19:59:49 UTC+0000 2018-08-01 20:00:08 UTC+0000
0x00000c20c6d5ac340 svchost.exe.ex 5528 4824 0x0000000119400002 2018-08-01 19:52:20 UTC+0000 2018-08-01 19:52:20 UTC+0000
0x00000c20c6d6fc580 svchost.exe 10012 4824 0x0000000136200002 2018-08-01 19:49:19 UTC+0000 2018-08-01 19:49:19 UTC+0000
0x00000c20c6d82e080 svchost.exe.volatility 1404 4824 0x00000000a0f00002 2018-08-01 19:54:55 UTC+0000 2018-08-01 19:56:35 UTC+0000
0x00000c20c6d99b580 svchost.exe.ex 8140 4824 0x00000000b8600002 2018-08-01 19:52:16 UTC+0000 2018-08-01 19:52:16 UTC+0000
0x00000c20c6db5340 svchost.exe 7852 4824 0x000000003ff00002 2018-08-01 19:49:21 UTC+0000 2018-08-01 19:49:22 UTC+0000
0x00000c20c6ddad580 svchost.exe 8560 4824 0x00000000602200002 2018-08-01 20:13:10 UTC+0000
```

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**.

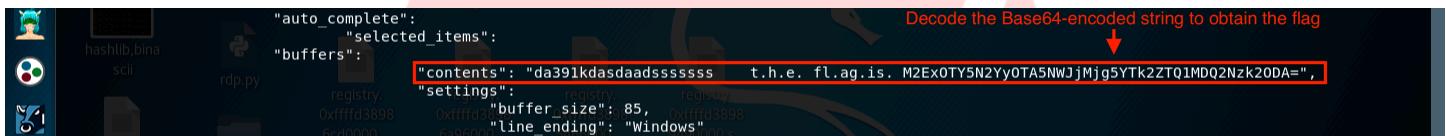
Question #2

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



```
root@kali:~/Desktop# python volatility/vol.py -f memdump.mem --profile=Win10x64_17134 memdump -p 8560 --dump-dir=.
Volatility Foundation Volatility Framework 2.6
*****
Writing svchost.exe [ 8560] to 8560.dmp
```

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:



```
Decode the Base64-encoded string to obtain the flag
↓
"contents": "da391kdasdadssssssss t.h.e. fl.ag.is. M2Ex0TY5N2Yy0TA5NWJjMjg5YTk2ZT01MDQ2Nzk20DA=",
```

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 **"010103000F0000F0080000000F0000F0E3E937A4D0CD0A314266D2986CB7DED5D8B43B828FE EDCEFFD6DE7141DC1D15D."** 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:

```
root@kali:~/Desktop# python volatility/vol.py -f memdump.mem --profile=Win10x64_17134 dumpregistry --dump-dir=./  
Volatility Foundation Volatility Framework 2.6  
*****  
Writing out registry: registry.0xfffffd38986bc4000.SAM.reg  
*****  
*****  
  
(Output Truncated)
```

```

root@kali:~/Desktop# ls *.reg
151552 Sep 5 13:37 registry.0xffffd38985466000.HARDWARE.reg
32768 Sep 5 13:37 registry.0xffffd38985e5a000.BCD.reg
72097792 Sep 5 13:37 registry.0xffffd38985eb3000.SOFTWARE.reg
270336 Sep 5 13:37 registry.0xffffd38986a9600.DEFAULT.reg
32768 Sep 5 13:37 registry.0xffffd38986bba000.SECURITY.reg
40960 Sep 5 13:37 registry.0xffffd38986c4000.SAM.reg
176128 Sep 5 13:37 registry.0xffffd38986c40000.NTUSERDAT.reg
319488 Sep 5 13:37 registry.0xffffd38986dc6000.BBT.reg
196608 Sep 5 13:37 registry.0xffffd38986dea000.NTUSERDAT.reg
1093632 Sep 5 13:37 registry.0xffffd389873c1000.ntuserdat.reg
3076096 Sep 5 13:37 registry.0xffffd389873fb000.UsrClassdat.reg
839680 Sep 5 13:37 registry.0xffffd38987c56000.Amcachehive.reg
114688 Sep 5 13:37 registry.0xffffd389892e2000.ActivationStoredat.reg
339968 Sep 5 13:37 registry.0xffffd389893e4000.ActivationStoredat.reg
8192 Sep 5 13:38 registry.0xffffd38989490000.settingsdat.reg
49152 Sep 5 13:37 registry.0xffffd3898a6e6000.dosvcStatedat.reg
188416 Sep 5 13:38 registry.0xffffd3898a6e6000.dosvcStatedat.reg
28672 Sep 5 13:37 registry.0xffffd3898c555000.settingsdat.reg
28672 Sep 5 13:37 registry.0xffffd3898ca19000.ActivationStoredat.reg
200704 Sep 5 13:38 registry.0xffffd3898e2c7000.ActivationStoredat.reg
8192 Sep 5 13:38 registry.0xffffd3898e336000.settingsdat.reg

root@kali:~/Desktop# rip.pl -r registry.0xffffd38985eb3000.SOFTWARE.reg -f software > out

```

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList

- \Signatures
 - \Unmanaged (record **DefaultGatewayMac**, **DnsSuffix**, **FirstNetwork (SSID)**, **ProfileGuid**)
 - \Managed
- \Nla
 - \Cache
- Profiles

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
menu.php	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	0xfffffd3898	0xfffffd3898	0xfffffd3898	0xfffffd3898	
\$FILE_NAME	Creation	Modified	MFT Altered	Access Date	Name/Path
physicsX	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	6bc4000 S...	6deaa000...	92e2000	0x7000 A...	

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.