CyberDefenders: NintendoHunt Lab

The following writeup is for NintendoHunt Lab hosted on CyberDefenders, it involves investigating a memory dump using volatility 2. Even though this room is rated as hard, it is honestly more leaning towards a medium difficulty. 13cubed is the creator of this room, and seeing as he produced incredible content I knew the challenge would be super fun. I highly recommend it for memory forensic enthusiasts out there.

Scenario: You have been hired as a SOC Analyst to investigate a potential security breach at a company. The company has recently noticed unusual network activity and suspects that there may be a malicious process running on one of their computers. Your task is to identify the malicious process and gather information about its activity.

What is the process ID of the currently running malicious process?

Starting off, let's determine what profile to use by issuing the imageinfo plugin:

```
vol.py -f memdump.mem imageinfo
```

This took a considerable amount of time, but eventually you can determine that the profile we need to use is Win10x64_17134. Now that we have the profile, we can start listing the processes running on the machine. In this instance, I am starting with the pstree command to see each process in hierarchical format:

```
vol.py -f memdump.mem --profile=Win10x64_17134 pstree
```

If you look through the results, we can see that there is multiple svchost.exe processes running with explorer.exe as their parent process:

```
.. 0xffffc20c69cfe580:explorer.exe                4824   4756
... 0xffffc20c6ddad580:svchost.exe                8560   4824
... 0xffffc20c6e495080:cmd.exe                    8868   4824
... 0xffffc20c6cdf4580:scvhost.exe                 360   4824
... 0xffffc20c6ab70080:svchost.exe                8852   4824
... 0xffffc20c6c095580:MSASCuiL.exe               6268   4824
... 0xffffc20c6d5ac340:svchost.exe.ex             5528   4824
... 0xffffc20c6ab2b580:svchost.exe.ex             6176   4824
... 0xffffc20c6d6fc580:svchost.exe               10012   4824
... 0xffffc20c6d4d2080:dxdiag.exe                 6324   4824
... 0xffffc20c6daf9580:notepad.exe                7968   4824
... 0xffffc20c6d789580:Bubbles.scr                6948   4824
... 0xffffc20c6e24f580:xwizard.exe                 252   4824
... 0xffffc20c6cfc2580:vmtoolsd.exe               3372   4824
... 0xffffc20c6d0d2080:Bubbles.scr               10204   4824
... 0xffffc20c6ab92580:ByteCodeGenera             6532   4824
... 0xffffc20c6dbc5340:svchost.exe                7852   4824
... 0xffffc20c6d36c080:scvhost.exe.ex              336   4824
... 0xffffc20c6d510080:notepad - Copy             6372   4824
... 0xffffc20c6cfb1580:OneDrive.exe               2200   4824
... 0xffffc20c6d82e080:svchost.exe                1404   4824
... 0xffffc20c6d86b080:cmd.exe                    3884   4824
.... 0xffffc20c6cec3080:conhost.exe               9912   3884
... 0xffffc20c6d732080:notepad.exe                9128   4824
... 0xffffc20c6abeb580:notepad.exe                1412   4824
... 0xffffc20c6d694080:notepad - Copy             3504   4824
... 0xffffc20c6e5ca200:notepad.exe                8800   4824
... 0xffffc20c6d99b580:svchost.exe.ex             8140   4824
... 0xffffc20c6e0bf580:scvhost.exe.ex             3016   4824
... 0xffffc20c6b588580:ie4uinit.exe               5716   4824
```

The expected parent process is most often services.exe. If you take a look at the command-line arguments for processes, we can see that the location of svchost.exe associated with PID 8560 is not correct:

```
vol.py -f memdump.mem --profile=Win10x64_17134 cmdline | grep svchost
```

```
svchost.exe pid:   8560
Command line : "C:\Windows\svchost.exe"
```

The expected path of the legitimate binary is system32\svchost.exe, not Windows\svchost.exe.

Answer: 8560

What is the md5 hash hidden in the malicious process memory?

In order to find the md5 hash hidden in the malicious process' memory, lets use the memdump command followed by strings:

```
vol.py -f memdump.mem --profile=Win10x64_17134 memdump -p 8560 -D ./
```

I then proceeded to run the strings command against the file and look for anything interesting. I eventually found the following:

```
strings 8560.dmp -n 15 | less
```

```
"auto_complete":
       "selected_items":
              "contents": "da391kdasdaadssssss    t.h.e. fl.ag.is. M2ExOTY5N2YyOTA5NWJjMjg5YTk2ZTQ1MDQ2Nzk2ODA=",
              "settings":
                     "buffer_size": 85,
                     "line_ending": "Windows"
```

This appears to be a base64 encoded string, let's decode it via the command line:

```
remnux@remnux:~$ echo "M2ExOTY5N2YyOTA5NWJjMjg5YTk2ZTQ1MDQ2Nzk2ODA=" | base64 -d
3a19697f29095bc289a96e4504679680remnux@remnux:~$
```

Answer: 3a19697f29095bc289a96e4504679680

What is the process name of the malicious process parent?

We determined this earlier to be explorer.exe:

```
.. 0xffffc20c69cfe580:explorer.exe                4824   4756
... 0xffffc20c6ddad580:svchost.exe                8560   4824
```

Answer: explorer.exe

What is the MAC address of this machine's default gateway?

To find the MAC address of this machine's default gateway, we can explore the Unmanged registry key using the printkey plugin like as follows:

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

```
Subkeys:
  (S) 010103000F0000F0080000000F0000F0E3E937A4D0CD0A314266D2986CB7DED5D8B43B828FEEDCEFFD6DE7141DC1D15D

Values:
```

Let's dig into this subkey:

```
remnux@remnux:~$ vol.py -f memdump.mem --profile=Win10x64_17134 printkey -K "Microsoft\Windows NT\CurrentVersion\NetworkList\Signatures\Unmanaged\010103000F0000F0080000000F0000F0E3E937A4D0CD0A314266D2986CB7DED5D8B43B828FEEDCEFFD6DE7141DC1D15D"
```

Here we can find the DefaultGatewayMac:

```
Values:
REG_SZ        ProfileGuid     : (S) {596B8D0F-BFBC-4B67-9ED8-237BD3DDABF3}
REG_SZ        Description     : (S) Network
REG_DWORD     Source          : (S) 8
REG_SZ        DnsSuffix       : (S) localdomain
REG_SZ        FirstNetwork    : (S) Network
REG_BINARY    DefaultGatewayMac : (S)
0x00000000  00 50 56 fe d8 07                                     .PV...
```

Answer: 00:50:56:fe:d8:07

What is the name of the file that is hidden in the alternative data stream?

To hunt for the name of a file that is hidden in the ADS, let's look through the MFT (Master File Table) using the mftparser command and grep for "ads name" like as follows:

```
vol.py -f memdump.mem --profile=Win10x64_17134 mftparser > out.txt
```

```
remnux@remnux:~$ grep -i "ads name" out.txt
$DATA ADS Name: $Bad
$DATA ADS Name: $Max
$DATA ADS Name: Zone.Identifier
$DATA ADS Name: yes.txt
```

Answer: yes.txt

What is the full path of the browser cache created when the user visited "www.13cubed.com" ?

Simply grep the mft output for "13cubed" like as follows:

```
remnux@remnux:~$ grep -i "13cubed" out.txt
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\Packages\MICROS~1.MIC\AC\#!001\MICROS~1\Cache\AHF2COV9\13cubed[1].htm
2018-08-01 19:37:05 UTC+0000 2018-08-01 19:37:05 UTC+0000   2018-08-01 19:37:05 UTC+0000   2018-08-01 19:37:05 UTC+0000   Users\CTF\AppData\Local\Packages\MICROS~1.MIC\AC\#!001\MICROS~1\Cache\IQDBNKYD\13cubed[1].png
```

Answer:
C:\Users\CTF\AppData\Local\Packages\MICROS~1.MIC\AC\#!001\MICROS~1\Cache\AHF2COV9\13cubed[1].htm