HTB- Sherlocks: Mellitus – Level: Medium
Amit Persky

Table of Contents:

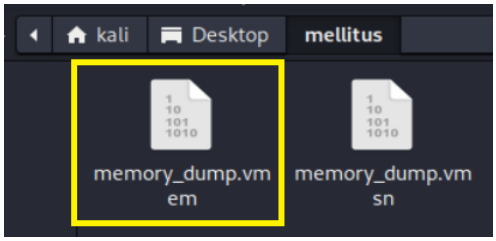
Intro:

You've been a SOC analyst for the last 4 years but you've been honing your incident response skills! It's about time you bite the bullet and go for your dream job as an Incident Responder as that's the path you'd like your career to follow. Currently you are going through the interview process for a medium size incident response internal team and the cocky interviewing responder has given you a tough technical challenge to test your memory forensics aptitude. Can you get all the questions right and secure the job?

Task 1:

What was the time on the system when the memory was captured?

So we get a zip file with two files inside him, we are going to work with the memory_dump.vmem file:



The tool that we are going to work with is volatility3 so make sure you have it:

```
└─$ git clone https://github.com/volatilityfoundation/volatility3.git
```

So now for answering the question I knew I need the flag for some info, so I checked in the help menu and the flag was found. *Windows.info.Info*

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ python3 vol.py --help
```

Then I started the tool with the correct syntax. It taking some time:

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ python3 vol.py -f /home/kali/Desktop/mellitus/memory_dump.vmem windows.info.Info
Volatility 3 Framework 2.7.2
Progress:  0.11                Reading file http://msdl.microsoft.com/download/symbols/ntkrnlmp.pdb/8B11040A5928757B11390AC
Progress:  0.22mp.pdb          Reading file http://msdl.microsoft.com/download/symbols/ntkrnlmp.pdb/8B11040A5928757B11390AC
Progress:  0.32mp.pdb          Reading file http://msdl.microsoft.com/download/symbols/ntkrnlmp.pdb/8B11040A5928757B11390AC
Progress:  0.43mp.pdb          Reading file http://msdl.microsoft.com/download/symbols/ntkrnlmp.pdb/8B11040A5928757B11390AC
```

At the end of this lines running we are getting info about the mem file:

```
MachineType     34404
KeNumberProcessors      2
SystemTime      [redacted]
NtSystemRoot    C:\Windows
NtProductType   NtProductWinNt
NtMajorVersion  10
NtMinorVersion  0
PE MajorOperatingSystemVersion  10
PE MinorOperatingSystemVersion  0
PE Machine      34404
PE TimeDateStamp        Thu Oct 28 12:04:50 2060
```

Task 2:

What is the IP address of the attacker?

So I looked up at the help menu again and I decided to use the *windows.netscen.NetScan* flag on the mem file:

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ python3 vol.py -f /home/kali/Desktop/mellitus/memory_dump.vmem windows.netscan.NetScan

Volatility 3 Framework 2.7.2
Progress:  100.00               PDB scanning finished
Offset  Proto    LocalAddr       LocalPort       ForeignAddr     ForeignPort     State   PID     Owner   Created

0×c40aa1be7050  TCPv4   0.0.0.0 135     0.0.0.0 0       LISTENING       880     svchost.exe     2023-10-31 13:30:57.000000
0×c40aa1be71a0  TCPv4   0.0.0.0 49664   0.0.0.0 0       LISTENING       472     wininit.exe     2023-10-31 13:30:57.000000
0×c40aa1be82b0  TCPv4   0.0.0.0 135     0.0.0.0 0       LISTENING       880     svchost.exe     2023-10-31 13:30:57.000000
```

Down I saw this and I realized that maybe im looking for ESTABLISHED connection:

```
0×c40aa58f4940  TCPv6   ::      445     ::      0       LISTENING       4       System  2023-10-31 13:30:59.000000
0×c40aa58f4a90  TCPv4   0.0.0.0 49668   0.0.0.0 0       LISTENING       620     services.exe    2023-10-31 13:30:59.000000
0×c40aa58f4a90  TCPv6   ::      49668   ::      0       LISTENING       620     services.exe    2023-10-31 13:30:59.000000
0×c40aa5ac6530  TCPv4   127.0.0.1       49867   127.0.0.1       49868   ESTABLISHED     -       -       N/A
0×c40aa5d25a30  TCPv4   192.168.157.144 50043   142.250.187.206 443     ESTABLISHED     -       -       N/A
0×c40aa5efe050  TCPv6   ::1     14147   ::      0       LISTENING       11048   FileZillaServe  2023-10-31 13:37:41.000000
0×c40aa5efe1a0  TCPv6   7f00:1::9870:b5a1:ac4:ffff      8005    ::      0       LISTENING       2696    java.exe        2023-
10-31 13:37:50.000000
0×c40aa5efe2f0  TCPv4   0.0.0.0 443     0.0.0.0 0       LISTENING       -       -       2023-10-31 13:37:41.000000
0×c40aa5efe2f0  TCPv6   ::      443     ::      0       LISTENING       -       -       2023-10-31 13:37:41.000000
0×c40aa5efe590  UDPv4   0.0.0.0 0       *       0               6772    powershell.exe  2023-10-31 13:42:37.000000
0×c40aa5efe830  TCPv4   0.0.0.0 7680    0.0.0.0 0       LISTENING       8444    svchost.exe     2023-10-31 13:32:59.000000
0×c40aa5efe830  TCPv6   ::      7680    ::      0       LISTENING       8444    svchost.exe     2023-10-31 13:32:59.000000
0×c40aa5efe980  TCPv4   0.0.0.0 21      0.0.0.0 0       LISTENING       11048   FileZillaServe  2023-10-31 13:37:41.000000
0×c40aa5efed70  TCPv4   0.0.0.0 443     0.0.0.0 0       LISTENING       -       -       2023-10-31 13:37:41.000000
0×c40aa5eff160  TCPv4   127.0.0.1       14147   0.0.0.0 0       LISTENING       11048   FileZillaServe  2023-10-31 13:37:41.0
00000
0×c40aa5eff400  TCPv4   0.0.0.0 8080    0.0.0.0 0       LISTENING       2696    java.exe        2023-10-31 13:37:48.000000
0×c40aa5eff400  TCPv6   ::      8080    ::      0       LISTENING       2696    java.exe        2023-10-31 13:37:48.000000
0×c40aa5effd30  TCPv4   0.0.0.0 3306    0.0.0.0 0       LISTENING       5212    mysqld.exe      2023-10-31 13:37:41.000000
0×c40aa5effd30  TCPv6   ::      3306    ::      0       LISTENING       5212    mysqld.exe      2023-10-31 13:37:41.000000
0×c40aa5effe80  TCPv4   0.0.0.0 21      0.0.0.0 0       LISTENING       11048   FileZillaServe  2023-10-31 13:37:41.000000
0×c40aa5effe80  TCPv6   ::      21      ::      0       LISTENING       11048   FileZillaServe  2023-10-31 13:37:41.000000
0×c40aa60fc1a0  UDPv6   ::1     1900    *       0               8140    svchost.exe     2023-10-31 13:31:20.000000
0×c40aa60fc2f0  UDPv6   ::1     56182   *       0               8140    svchost.exe     2023-10-31 13:31:20.000000
0×c40aa60fc440  UDPv4   127.0.0.1       1900    *       0               8140    svchost.exe     2023-10-31 13:31:20.000000
0×c40aa60fc6e0  UDPv4   192.168.157.144 56183   *       0               8140    svchost.exe     2023-10-31 13:31:20.000000
0×c40aa60fcad0  UDPv4   192.168.157.144 1900    *       0               8140    svchost.exe     2023-10-31 13:31:20.000000
0×c40aa60fd6a0  UDPv6   fe80::a94d:3c5d:b0c7:221b       56181   *       0               8140    svchost.exe     2023-10-31 13
:31:20.000000
0×c40aa60fd7f0  UDPv4   127.0.0.1       56184   *       0               8140    svchost.exe     2023-10-31 13:31:20.000000
0×c40aa60fd940  TCPv4   127.0.0.1       8888    0.0.0.0 0       LISTENING       2896    python.exe      2023-10-31 13:37:16.0
00000
```

So I decided to save the results for working with text manipulation:

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ python3 vol.py -f /home/kali/Desktop/mellitus/memory_dump.vmem windows.netscan.NetScan > /home/kali/Desktop/netscan_resul
ts.txt
```

After some text manipulation I got this results, because I knew I need ESTABLISHED and ForeignAddr:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ cat netscan_results.txt | grep ESTABLISHED | awk '{print $5}' | sort | uniq
127.0.0.1
13.107.21.200
142.250.187.206
[redacted]
204.79.197.222
20.90.152.133
216.58.204.67
216.58.204.78
```

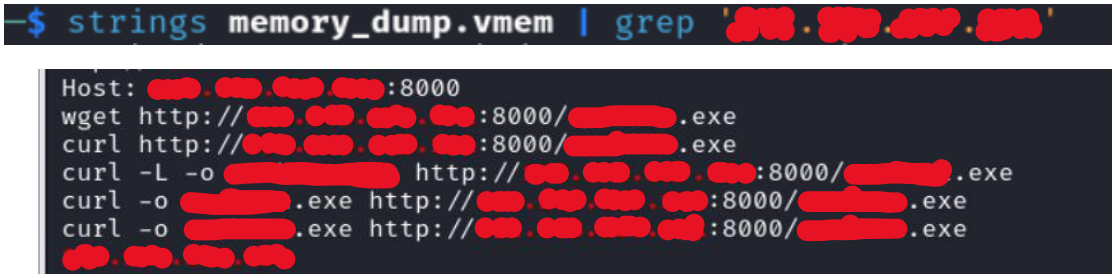
So beside of the loopback address there is only 1 private address so I chose her.

Task 3:

What is the name of the strange process?

For finding the answer I walked again to the –help menu and saw a lot of options, including *windows.pslist.PsList* and *windows.suspicious_threads.SupsiciousThreads* and no matter how much I tried, also putting in all the processes from pslist, I didn't have a match.

So I took step back and I tried to find something is related to the IP from the last question-[redacted]. I used strings and grep commands to find the answer:



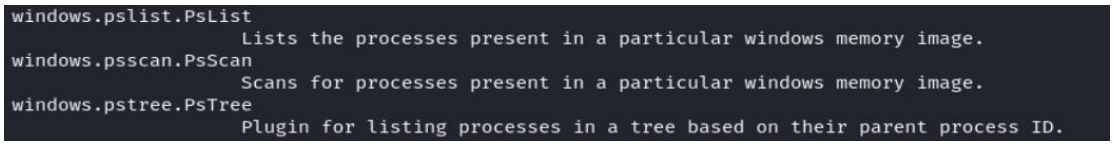
I realized that this is unusual because the legetime processe that we all know is svchost.exe and not like here, that it almost similar, In order to confuse you.

**Try this with capital letter..

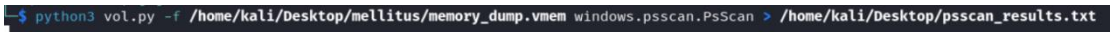
Task 4:

What is the PID of the process that launched the malicious binary?

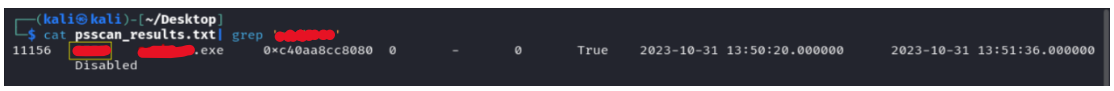
For this question I looked again for –help any flags of Lists processes… found this three:



I tried them all but this is the one who helped:



I searched in the results file the process from the last question I got the answer with grep of the name of the process from the last question:



Task 5:

What was the command that got the malicious binary into the machine?

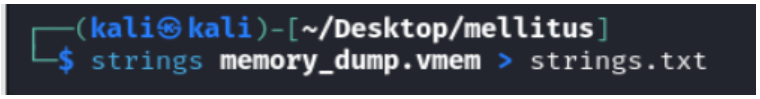
If you remember in task 3 we found this:

Curl it's known command for downloading files from the net, and we alrdy know that [redacted].exe.

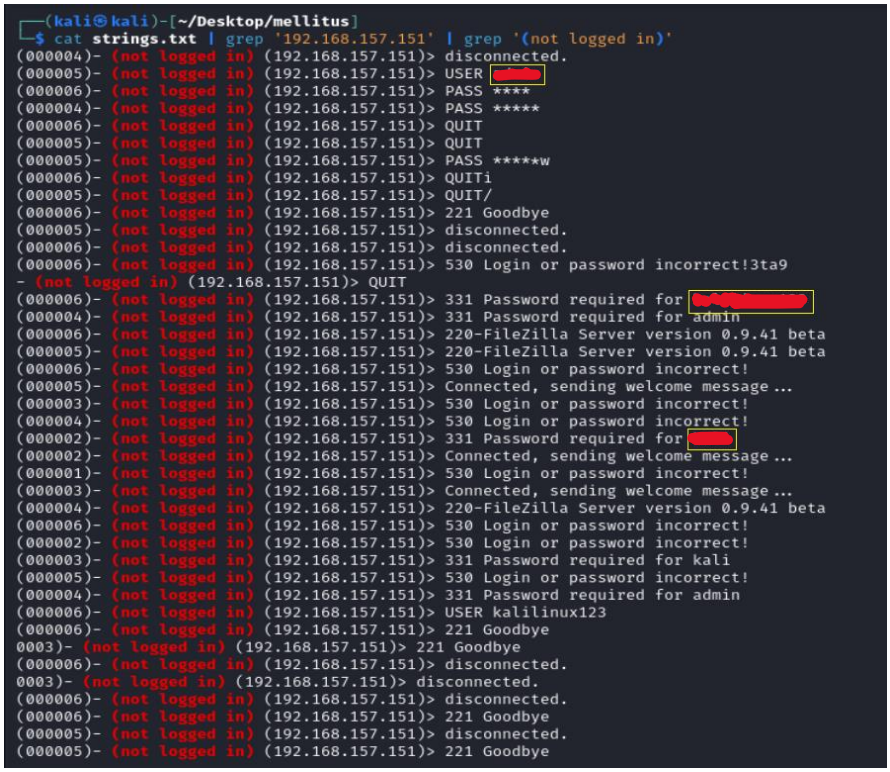
Task 6:

The attacker attempted to gain entry to our host via FTP. How many users did they attempt?

Ok so we need to look for FTP logs or some kind of logs from our attacker... I used "strings" command on the entire memory file:



I played with this file some times with grep until this command:



As we can see there is several attemps to log in, for some different users.

Task 7:

What is the full URL of the last website the attacker visited?

For that I looked at the "netscan" file result from before to see if I have clue for any web apps like chrome or egde or.

```
0×c40aaa0fa050  UDPv4   0.0.0.0 0       *       0               6772    powershell.exe  2023-10-31 13:42:37.000000
0×c40aaa0fa050  UDPv6   ::      0       *       0               6772    powershell.exe  2023-10-31 13:42:37.000000
0×c40aaa0fa1a0  UDPv4   0.0.0.0 5355    *       0               1876    svchost.exe     2023-10-31 13:55:22.000000
0×c40aaa0fa440  UDPv4   0.0.0.0 5353    *       0               8048    chrome.exe      2023-10-31 13:55:31.000000
0×c40aaa0fa590  UDPv4   0.0.0.0 0       *       0               6772    powershell.exe  2023-10-31 13:42:37.000000
0×c40aaa0fac20  UDPv4   0.0.0.0 5353    *       0               8048    chrome.exe      2023-10-31 13:55:31.000000
0×c40aaa0fac20  UDPv6   ::      5353    *       0               8048    chrome.exe      2023-10-31 13:55:31.000000
0×c40aaa0fb160  UDPv4   0.0.0.0 5355    *       0               1876    svchost.exe     2023-10-31 13:55:22.000000
0×c40aaa0fb160  UDPv6   ::      5355    *       0               1876    svchost.exe     2023-10-31 13:55:22.000000
0×c40aaa0fba90  UDPv4   0.0.0.0 0       *       0               6772    powershell.exe  2023-10-31 13:42:37.000000
0×c40aaa0fba90  UDPv6   ::      0       *       0               6772    powershell.exe  2023-10-31 13:42:37.000000
0×c40aaa5d7920  TCPv4   192.168.157.144 50044   204.79.197.222  443     ESTABLISHED     -       -       N/A
0×c40aaa7f79a0  TCPv4   192.168.157.144 50037   192.168.157.151 4545    ESTABLISHED     -       -       N/A
0×c40aaa8cb8a0  TCPv4   192.168.157.144 50041   216.58.204.78   443     ESTABLISHED     -       -       N/A
0×c40aaa8d7920  TCPv4   192.168.157.144 50039   20.31.169.57    443     CLOSED  -       -       N/A
```

As we can see we found chrome 8048 at the last spots and we can see that with the hour in the right side. So we know we need to search in chrome files. I used the flag of *windows.filescan.FileScan* for the try the find files to connect to chrome:

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ python3 vol.py -f /home/kali/Desktop/mellitus/memory_dump.vmem windows.filescan.FileScan > filescan_results.txt
```

After that we got the locations of files on the memory dump and their details.

I didn't know where to find the chrome history so I visited this website:

https://www.foxtonforensics.com/browser-history-examiner/chrome-history-location

this site gave me the idea what to look for with some grep flags in the filescan results:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ cat filescan_results.txt | grep -i 'history'
0×c40aa54d5290  \Windows\System32\CallHistoryClient.dll 216
0×c40aa6235c70  \ProgramData\Microsoft\Windows Defender\Scans\History\CacheManager\5B84E85D-C26F-433C-850B-0EFEA165D272-1.bin
216
0×c40aa6724e20  \Windows\System32\winevt\Logs\Microsoft-Windows-FileHistory-Core%4WHC.evtx      216
0×c40aa9259df0  \Users\BantingFG\AppData\Local\Google\Chrome\User Data\Default\History  216
0×c40aa952e740  \Users\BantingFG\AppData\Local\Google\Chrome\User Data\Default\History-journal  216
```

As we can see we found some results that looking good, we can see that every path here got offset that can help us.

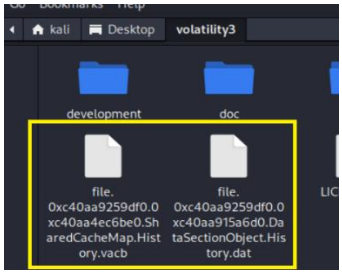
The string "0xc40aa9259df0" looks like a virtual memory address in hexadecimal format. I found in Google that virtual addresses like that are often used to reference locations in the memory space of a process or the system.

For going forward, I understood that I need to dump some file/files that connected to the chrome history. I tried to search in the volatility dumpfiles help and I realized that I can dump it with some additional flag.

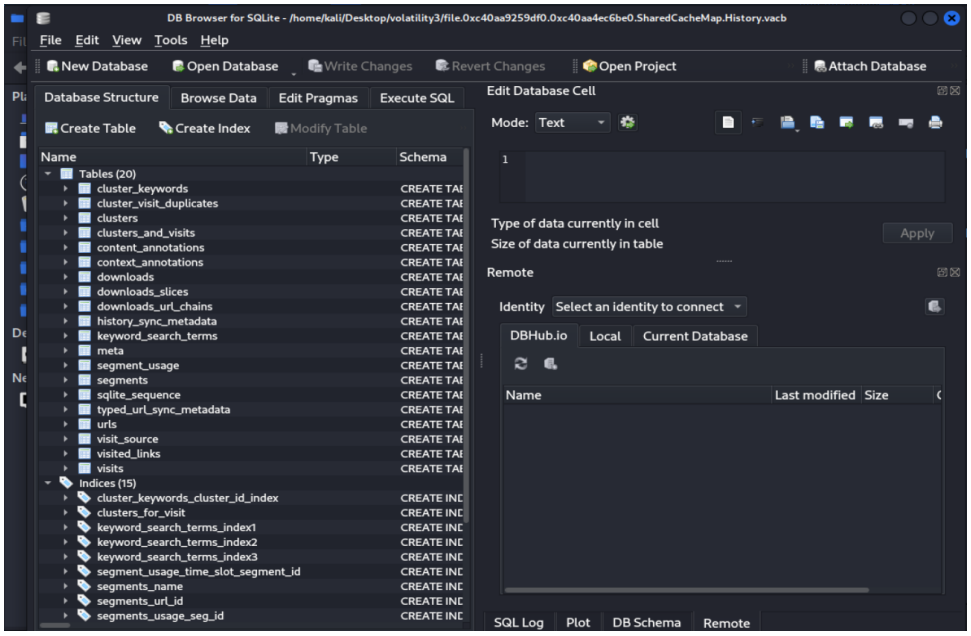
back to the terminal quickly for that syntax gave me some interesting results:

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ python3 vol.py -f /home/kali/Desktop/mellitus/memory_dump.vmem windows.dumpfiles --virtaddr 0xc40aa9259df0

Volatility 3 Framework 2.7.2
Progress:  100.00               PDB scanning finished
Cache   FileObject      FileName        Result

DataSectionObject       0xc40aa9259df0  History file.0xc40aa9259df0.0xc40aa915a6d0.DataSectionObject.History.dat
SharedCacheMap  0xc40aa9259df0  History file.0xc40aa9259df0.0xc40aa4ec6be0.SharedCacheMap.History.vacb
```

As we can see we got two files. I looked in the folder of volatility (I didn't chose for them spcefic place for extract) and saw them:

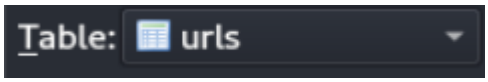


I tried to open the file that ends with .dat - the file didn't opend so I tried to open the file that ends with .vacb it opened like that:

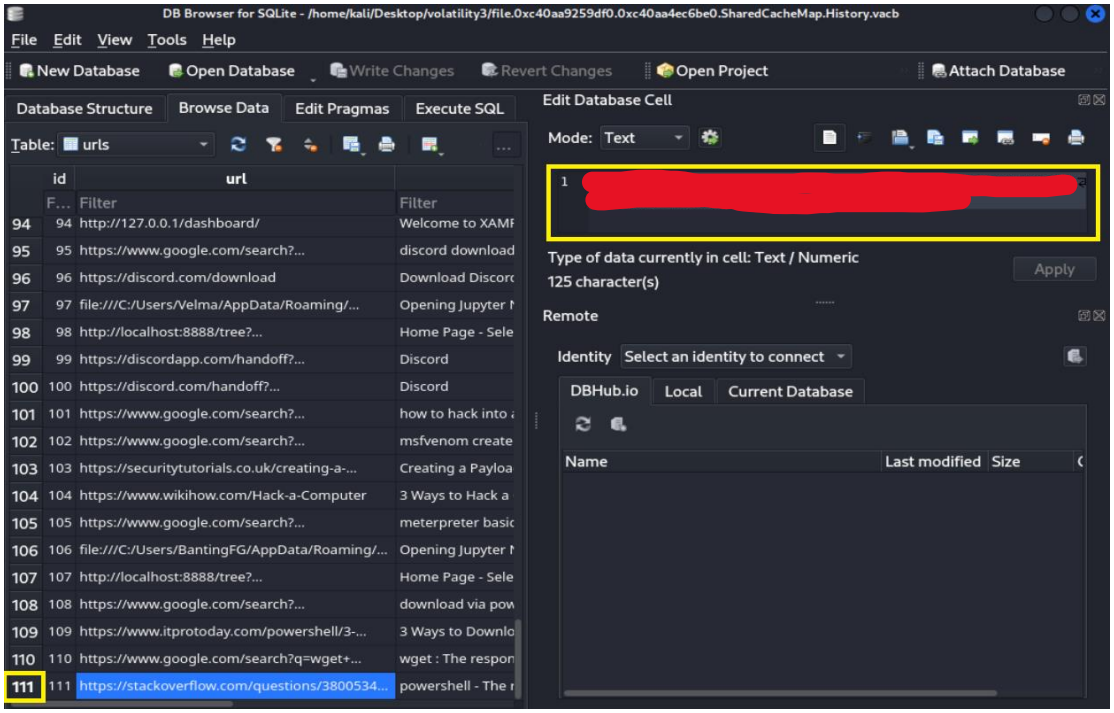


I looked in and I opened the Browse Data tab

Moved to Table urls:



Then I got list with urls. I scrolled down to the last one 111:



I pressed on that and copy the url from the right side.

Task 8:

What is the affected users password?

For that task I realized that I need to dump the hash passwords from the mem file and I thought that I got good flag for that in the –help menu so I got the flag *windows.hashdump.Hashdump*

I tried to use several times the flag *windows.hashdump.Hashdump* but It didn't work to me.. so I tried to search in the volatility3 directory and I opend the file "requirements.txt" and then I saw this:

```
# This is required by plugins that decrypt passwords, password hashes, etc.
pycryptodome
```

, then I understood that I need to install some plugin for volatility if I want that this flag will work for me…

I installed the plugin "pycryptodome"

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ pip3 install pycryptodome
Defaulting to user installation because normal site-packages is not writeable
Collecting pycryptodome
  Using cached pycryptodome-3.20.0-cp35-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.4 kB)
Using cached pycryptodome-3.20.0-cp35-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.1 MB)
Installing collected packages: pycryptodome
Successfully installed pycryptodome-3.20.0
```

Then I tried again the flag windows.hashdump.Hashdump and its worked to me

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ python3 vol.py -f /home/kali/Desktop/mellitus/memory_dump.vmem windows.hashdump.Hashdump

Volatility 3 Framework 2.7.2
Progress:  100.00               PDB scanning finished
User    rid     lmhash  nthash

Administrator   500     aad3b435b51404eeaad3b435b51404ee        31d6cfe0d16ae931b73c59d7e0c089c0
Guest   501     aad3b435b51404eeaad3b435b51404ee        31d6cfe0d16ae931b73c59d7e0c089c0
DefaultAccount  503     aad3b435b51404eeaad3b435b51404ee        31d6cfe0d16ae931b73c59d7e0c089c0
WDAGUtilityAccount      504     aad3b435b51404eeaad3b435b51404ee        b47a9f2da3e6d7b88213822b52232627
Admin   1001    aad3b435b51404eeaad3b435b51404ee        3dbde697d71690a769204beb12283678
BantingFG       1002    aad3b435b51404eeaad3b435b51404ee        5a4a40e43197cd4dfb7c72e691536e92
```

I saved the hashes to txt file :

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ python3 vol.py -f /home/kali/Desktop/mellitus/memory_dump.vmem windows.hashdump.Hashdump >hashes.txt
```

Now, I checked again the text file to see if he ok, and as you can see the lines are not organized as NTLM shape as he supposed to be.

```
1  Volatility 3 Framework 2.7.2
2
3  User    rid lmhash  nthash
4
5  Administrator   500 aad3b435b51404eeaad3b435b51404ee    31d6cfe0d16ae931b73c59d7e0c089c0
6  Guest   501 aad3b435b51404eeaad3b435b51404ee    31d6cfe0d16ae931b73c59d7e0c089c0
7  DefaultAccount  503 aad3b435b51404eeaad3b435b51404ee    31d6cfe0d16ae931b73c59d7e0c089c0
8  WDAGUtilityAccount  504 aad3b435b51404eeaad3b435b51404ee    b47a9f2da3e6d7b88213822b52232627
9  Admin   1001    aad3b435b51404eeaad3b435b51404ee    3dbde697d71690a769204beb12283678
10 BantingFG   1002    aad3b435b51404eeaad3b435b51404ee    5a4a40e43197cd4dfb7c72e691536e92
11
```

So I decided to delete the spaces and inserting the ":" instead them and at the end I wrote ":::" three times at the normal structure. After the changes It looks like that:

```
1  Volatility 3 Framework 2.7.2
2
3  User    rid lmhash  nthash
4
5  Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
6  Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
7  DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
8  WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:b47a9f2da3e6d7b88213822b52232627:::
9  Admin:1001:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
10 BantingFG:1002:aad3b435b51404eeaad3b435b51404ee:5a4a40e43197cd4dfb7c72e691536e92:::
11
```

Now we need to decrypt the hashes I tried to use hashcat for that with rockyou.txt list:

```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ hashcat -m 1000 hashes.txt /home/kali/Desktop/rockyou.txt
```

And then I found this results:

```
Dictionary cache built:
* Filename..: /home/kali/Desktop/rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 2 secs

3dbde697d71690a769204beb12283678:[redacted]
31d6cfe0d16ae931b73c59d7e0c089c0:
5a4a40e43197cd4dfb7c72e691536e92:[redacted]
```

As we can see of found 2 passwords, I tried them and and one of them is the answer.

Task 9:

There is a flag hidden related to PID 5116. Can you confirm what it is?

Ok so I thought that I need to get something that related to PID 5116 I tried to dump it with the PID 5116 trying for get something:

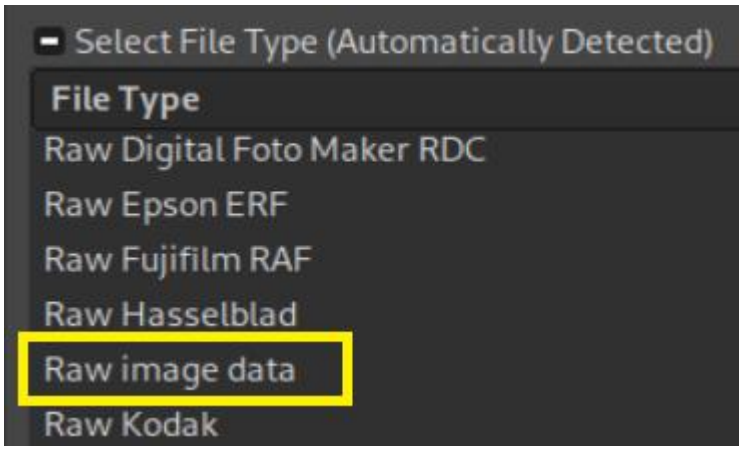
```
┌──(kali㉿kali)-[~/Desktop/volatility3]
└─$ python3 vol.py -f /home/kali/Desktop/mellitus/memory_dump.vmem windows.dumpfiles.DumpFiles --pid 5116
```

read the task 9 hint:

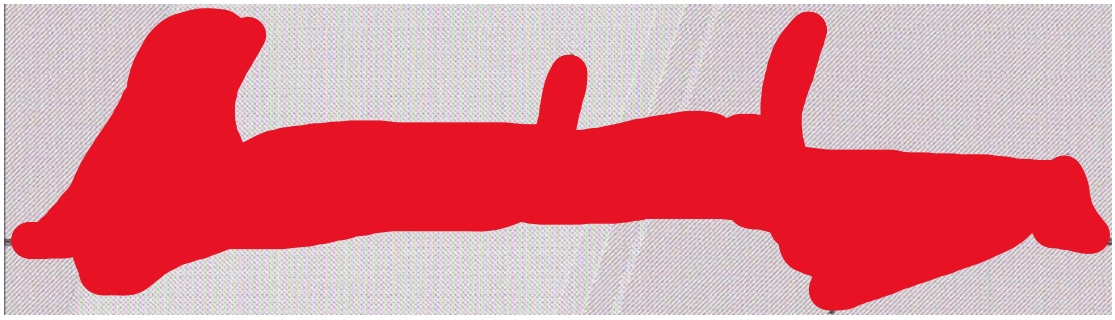
" Dump PID 5116, you may to utilise GIMP or something similar to find the flag."

Ok so I opened the file using GIMP (install it if you don't have)

So I opend the file as a raw image data



Then I played with the options, adjusting the offset until I say a 2 words:



I saw in the place for the answer in HTB that the first word is 3 letters then I got the idea for the order.