CFRS 772: Forensic Artifact Extraction
Homework 7

Overview: complete a python program to extract E_PROCESS and E_THREAD blocks (process and thread structures) from a memory image and compare the results to Volatility. You do not need to turn in anything for Steps 1-3; they provide some background information and data that may be useful as you work on the python code. You may use Volatility 2.5 or 2.6; instructions below renamed the volatility executable to vol.exe

1. Volatility

- Download the vm.vmem memory image from BlackBoard (HW7 folder)
  - This is a memory image from a Windows XP system
- Running the Volatility imageinfo plugin gives these results (truncated for readability)

```
> vol.exe -f vm.vmem imageinfo
Volatility Foundation Volatility Framework 2.X
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86, ...
           Image date and time : 2014-09-10 16:46:56 UTC+0000
     Image local date and time : 2014-09-10 12:46:56 -0400
```

- Running the Volatility pslist plugin gives these results (truncated for readability)

```
>vol.exe -f vm.vmem --profile=WinXPSP2x86 pslist

Volatility Foundation Volatility Framework 2.X

Name          PID    PPID    Thds
---------- ------ -------  ------
System 4      0      59
smss.exe      548    4        3
csrss.exe     620    548      11
winlogon.exe  644    548      18
services.exe  688    644      16
lsass.exe     700    644      21
vmacthlp.exe  800    688      1
svchost.exe   884    688      19
svchost.exe   980    688      11
svchost.exe   1084   688      70
svchost.exe   1160   688      5
svchost.exe   1272   688      16
explorer.exe  1532   1504     13
spoolsv.exe   1600   688      10
vmtoolsd.exe  1716   1532     6
vmtoolsd.exe  2020   688      7
wscntfy.exe   500    1084     1
alg.exe       1288   688      7
wuauclt.exe   1052   1084     8
wuauclt.exe   1328   1084     4
cmd.exe       1252   1532     1
wmiprvse.exe  1136   884      7
```

```
cmd.exe       1676   2020     0
ipconfig.exe  1364   1676     0
```

2. Manual identification of EPROCESS blocks (an extension of Lab 7)

- Open the vm.vmem memory image in HxD
- An EPROCESS block has a known header; search for the hex string:

```
03 00 1B 00 00 00 00 00
```

- Your first two hits will be a false positive and the idle process; continue (F3) on to a subsequent hit.
- An EPROCESS block also has a known structure. Set the HxD View: OffsetBase to decimal, then look at the text located at offset 372 bytes from the start of the EPROCESS block (the block starts at 03 00 ...). This is the process (image) name; record the text here (the ASCII characters until you see a 0x00):

  _____

- Now look at the two bytes at offset 132 bytes. These bytes are the process ID (PID). Convert the two bytes (little endian) to decimal as follows:

  - reverse the bytes (e.g., 2C A1 becomes A1 2C)
  - convert the hex bytes to decimal (use the converter at http://www.mathsisfun.com/binary-decimal-hexadecimal-converter.html or similar)

- Record the PID in decimal here:

  _____

- The Parent PID (PPID) is at offset 332, also two bytes stored little endian. Record the PPID in decimal here:

  _____

- Repeat the procedure above and record the process name and PID for 2 more processes; record this information below (process name, PID, PPID, offset):

  _____ _____ _____ _____

  _____ _____ _____ _____

- Do these values match the Volatility output? (Y/N) _____

  - If so, continue; if not, something is wrong...repeat step 2

3. Manual identification of ETHREAD blocks

- An ETHREAD block has a known header; search for the hex string:

```
06 00 70 00 00 00 00 00
```

- Your first two hits will be a false positive and the idle process; continue (F3) on to a subsequent hit.
- An ETHREAD block also has a known structure. Set the HxD View: OffsetBase to decimal, then look at the text located at offset 492 bytes from the start of the ETHREAD block (the block starts at 06 00 ...). This is the process ID (PID) of the thread's parent. Convert the two bytes (little endian) to decimal as follows:

    - reverse the bytes (e.g., 2C A1 becomes A1 2C)
    - convert the hex bytes to decimal (use the converter at http://www.mathsisfun.com/binary-decimal-hexadecimal-converter.html or similar)

- Record the thread's parent PID in decimal here:

    _____

    FYI, the Thread also has an ID, called the Thread ID (TID) which is drawn from the same pool of numbers as the Process ID (0-65535). The TID is located at offset 496.

- Repeat the procedure above and record the thread's parent PID for 2 more processes; record this PID below

- _____

- _____

If none of the PIDs you find match PIDs in the Voaltility process output, keep going until you do find one that matches (you don't need to record all the ones that don't match).

Which process did you find a match for: _____

Try to find more than one thread for that process; if so, list a couple of ETHREAD offsets here

_____

_____

4. **Python extraction of Process and Thread block information**

- Download HW7_template.py from Blackboard
- Complete the code to extract process and thread blocks (see notes in code)
- Check this data against what you found manually and with Volatility; in your submission, comment on any discrepencies (in the BlackBoard submission comments section; not a separate document)

On BlackBoard, submit your Python code; name your program hw7.py, zip to the file hw7.zip, and submit the file hw7.zip only. The file hw7.zip will only contain your program, named hw7.py. In the comments section on BlackBoard, briefly comment on any discrepencies between the python code output and Volatility.