1. get your volatility on - 5 pts

What is the SHA1 hash of triage.mem?

```
sha1sum /mnt/hgfs/mem/Triage-Memory.mem
```

Answer:flag<c95e8cc8c946f95a109ea8e47a6800de10a27abd>

2. pr0file - 10 pts

What profile is the most appropriate for this machine? (ex: Win10x86_14393)

```
vol.py -f /mnt/hgfs/mem/Triage-Memory.mem imageinfo
```

Usually, the first one works:

```
remnux@remnux:~/Documents/defcon19/mem$ vol.py -f /mnt/hgfs/mem/Triage-Memory.mem imageinfo
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: CryptographyD
eprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated
in cryptography, and will be removed in the next release.
  from cryptography.hazmat.backends.openssl import backend
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Win2008R2SP1
x64_23418, Win2008R2SP1x64, Win7SP1x64_24000, Win7SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/mnt/hgfs/mem/Triage-Memory.mem)
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf800029f80a0L
          Number of Processors : 2
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff800029f9d00L
                KPCR for CPU 1 : 0xfffff880009ee000L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2019-03-22 05:46:00 UTC+0000
     Image local date and time : 2019-03-22 01:46:00 -0400
```

Answer:flag<Win7SP1x64>

3. hey, write this down - 12 pts

What was the process ID of notepad.exe?

```
vol.py  -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64 pslist |
```

```
grep "notepad.exe"
```

```
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
------------------ ------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
```

By redirecting the output to grep, there is no need to look through al processes, given the above, the Process ID (PID) is 3032.

```
0xfffffa80054f9060 notepad.exe            3032   1432      1       60      1      0 2019-03-22 05:32:22 UTC+0000
```

Answer: flag<3032>

4. wscript can haz children - 14 pts

Name the child processes of wscript.exe.

```
vol.py  -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64 pstree
```

We can see that wscript.exe has a child process: UWkpjFjDzM.exe, which in turn also has a child process: cmd.exe

```
Name                                                  Pid   PPid   Thds   Hnds Time
----------------------------------------------- ------ ------ ------ ------ ----
 0xfffffa8003de39c0:explorer.exe                   1432   1308     28    976 2019-03-22 05:32:07 UTC+0000
. 0xfffffa80042aa430:cmd.exe                       1408   1432      1     23 2019-03-22 05:34:12 UTC+0000
. 0xfffffa8005d067d0:StikyNot.exe                  1628   1432      8    183 2019-03-22 05:34:42 UTC+0000
. 0xfffffa80042dbb30:chrome.exe                    3248   1432     32    841 2019-03-22 05:35:14 UTC+0000
.. 0xfffffa8005442b30:chrome.exe                   4232   3248     14    233 2019-03-22 05:35:17 UTC+0000
.. 0xfffffa80047beb30:chrome.exe                   3244   3248      7     91 2019-03-22 05:35:15 UTC+0000
.. 0xfffffa80053306f0:chrome.exe                   1816   3248     14    328 2019-03-22 05:35:16 UTC+0000
.. 0xfffffa8005300b30:chrome.exe                   4156   3248     14    216 2019-03-22 05:35:17 UTC+0000
.. 0xfffffa8005419b30:chrome.exe                   4240   3248     14    215 2019-03-22 05:35:17 UTC+0000
.. 0xfffffa800540db30:chrome.exe                   4520   3248     10    234 2019-03-22 05:35:18 UTC+0000
.. 0xfffffa80052f0060:chrome.exe                   2100   3248      2     59 2019-03-22 05:35:15 UTC+0000
.. 0xfffffa80053cbb30:chrome.exe                   4688   3248     13    168 2019-03-22 05:35:19 UTC+0000
. 0xfffffa800474c060:OUTLOOK.EXE                   3688   1432     30   2023 2019-03-22 05:34:37 UTC+0000
. 0xfffffa8004798320:calc.exe                      3548   1432      3     77 2019-03-22 05:34:43 UTC+0000
. 0xfffffa80053d3060:POWERPNT.EXE                  4048   1432     23    765 2019-03-22 05:35:09 UTC+0000
. 0xfffffa8004905620:hfs.exe                       3952   1432      6    214 2019-03-22 05:34:51 UTC+0000
.. 0xfffffa8005a80060:wscript.exe                  5116   3952      8    312 2019-03-22 05:35:32 UTC+0000
... 0xfffffa8005a1d9e0:UWkpjFjDzM.exe              3496   5116      5    109 2019-03-22 05:35:33 UTC+0000
.... 0xfffffa8005bb0060:cmd.exe                    4660   3496      1     33 2019-03-22 05:35:36 UTC+0000
```

Answer: flag<UWkpjFjDzM.exe>

5. tcpip settings - 18 pts

What was the IP address of the machine at the time the RAM dump was created?

In question 1, the **imageinfo** plugin displayed the date and time of the image – **2019-03-22 05:46:00 UTC+0000.** To find the IP address of the machine, we can use the **netscan** plugin:

```
vol.py  -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64  netscan
```

Answer: flag<10.0.0.101>

6. intel - 18 pts

Based on the answer regarding to the infected PID, can you determine what the IP of the attacker was?

```
vol.py  -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64  netscan
```

Using the same plug-in as before, there is an IP address on the 4444 Listening port, which is Metasploit's default port!

```
0x13e397190        TCPv4    10.0.0.101:49217            10.0.0.106:4444      ESTABLISHED      3496     UWkpjFjDzM.exe
0x13e3986d0        TCPv4    -:49378                     213.209.1.129:25     CLOSED           504
```

Answer:flag<10.0.0.106>

7. i <3 windows dependencies - 20 pts

What process name is VCRUNTIME140.dll associated with?

```
vol.py  -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64  dlllist
```

The output for this command is massive, so using “Find” to look for VCRUNTIME140.dll shows it’s associated with Office16, however it’s also associated with ‘**OfficeClickToRun.exe’**, the process name is “OfficeClickToR” (as seen on the first line of the output):

```
OfficeClickToR pid:   1136
Command line : "C:\Program Files\Common Files\Microsoft Shared\ClickToRun\OfficeClickToRun.exe" /service
Service Pack 1

Base                             Size          LoadCount LoadTime                     Path
------------------ ------------------ ------------------ ---------------------------- ----
0x000000013f420000           0xa9d000             0xffff 1970-01-01 00:00:00 UTC+0000   C:\Program Files\Common Files\Microsoft Shared\ClickToRun\OfficeClickToRun.exe
0x0000000077260000           0x1a9000             0xffff 1970-01-01 00:00:00 UTC+0000   C:\Windows\SYSTEM32\ntdll.dll
0x0000000077040000           0x11f000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\kernel32.dll
0x000007fefd380000            0x6c000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\KERNELBASE.dll
0x000007fefe970000            0xdb000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\ADVAPI32.dll
0x000007fefd6b0000            0x9f000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\msvcrt.dll
0x000007feff160000            0x1f000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\SYSTEM32\sechost.dll
0x000007fefe7a0000           0x12d000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\RPCRT4.dll
0x000007fefef60000            0x67000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\GDI32.dll
0x0000000077160000            0xfa000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\USER32.dll
0x000007feff150000            0xe000              0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\LPK.dll
0x000007fefe5e0000            0xc9000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\USP10.dll
0x000007fefb140000            0x27000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007fefe790000             0x8000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007fefb100000             0xb000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007feff180000           0x203000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007fefe6b0000            0xd7000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007feff3f0000            0x4d000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007fefa5e0000            0x1b000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007fefd2a0000            0x3b000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007fefd3f0000           0x16d000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007fefd250000             0xf000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007fefb2d0000            0x11000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007fefed80000           0x1d7000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007fefd260000            0x36000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\sy
0x000007fefd560000            0x1a000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\DEVOBJ.dll
0x000007fefa5c0000            0x16000             0xffff 2019-03-22 05:32:05 UTC+0000   C:\Program Files\Common Files\Microsoft Shared\ClickToRun\VCRUNTIME140.dll
```

```
Find
VCRUNTIME140.dll
Match case
Match entire word only
Match as regular expression
Wrap around
```

Answer: flag<OfficeClickToR>

8. mal-ware-are-you - 20 pts

What is the md5 hash value the potential malware on the system?

The **potential malware**, is the executable from question 4, which has a pid of 3496. Using the dump command to extract it:

```
vol.py -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64 procdump -D ./
```

```
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0xfffffa8005a1d9e0 0x0000000000400000 UWkpjFjDzM.exe       OK: executable.3496.exe
```

Then, using the md5sum on the dumped executable to find its hash value:

```
md5sum executable.3496.exe
```

```
remnux@remnux:~/Documents/defcon19/mem$ md5sum executable.3496.exe
690ea20bc3bdfb328e23005d9a80c290  executable.3496.exe
```

Answer: flag<690ea20bc3bdfb328e23005d9a80c290>

9. lm-get bobs hash - 24 pts

What is the LM hash of bobs account?

```
vol.py  -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64  hashdump
```

This is the fastest way to get a **LM hash** for all Users on the machine:

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Bob:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

10. vad the impaler - 25 pts

What protections does the VAD node at 0xfffffa800577ba10 have?

```
vol.py  -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64 vadinfo |
grep "0xfffffa800577ba10" -A 10
```

Open the **Basic_info** table and you will see that **macOS Version is 10.15**

```
VAD node @ 0xfffffa800577ba10 Start 0x0000000000030000 End 0x0000000000033fff Tag Vad
Flags: NoChange: 1, Protection: 1
Protection: PAGE_READONLY
Vad Type: VadNone
ControlArea @fffffa8005687a50 Segment fffff8a000c4f870
NumberOfSectionReferences:          1 NumberOfPfnReferences:           0
NumberOfMappedViews:               29 NumberOfUserReferences:         30
Control Flags: Commit: 1
First prototype PTE: fffff8a000c4f8b8 Last contiguous PTE: fffff8a000c4f8d0
Flags2: Inherit: 1, SecNoChange: 1
```

Answer:flag<PAGE_READONLY>

11. more vads?! - 25 pts

What protections did the VAD starting at 0x00000000033c0000 and ending at 0x00000000033dffff have?

```
vol.py  -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64 vadinfo |
grep "0x00000000033c0000" -A 10
```

This outputs several VAD nodes starting at 0x00000000033c0000, but only one ending at 0x00000000033dffff:

```
VAD node @ 0xfffffa80052652b0 Start 0x00000000033c0000 End 0x00000000033dffff Tag VadS
Flags: CommitCharge: 32, PrivateMemory: 1, Protection: 24
Protection: PAGE_NOACCESS
Vad Type: VadNone
```

Answer: flag<PAGE_NOACCESS>

12. vacation bible school - 25 pts

There was a VBS script run on the machine. What is the name of the script? (submit without file extension)

```
vol.py -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64 cmdline
```

The cmdline plug-in shows everything, but, look what we got here!!!

```
wscript.exe pid:   5116
Command line : "C:\Windows\System32\wscript.exe" //B //NOLOGO %TEMP%\vhjReUDEuumrX.vbs
**********************************************************************
UWkpjFjDzM.exe pid:   3496
Command line : "C:\Users\Bob\AppData\Local\Temp\rad93398.tmp\UWkpjFjDzM.exe"
**********************************************************************
```

Process ID of the .vbs script is the parent process of the malware!!

13. thx microsoft - 25 pts

An application was run at 2019-03-07 23:06:58 UTC, what is the name of the program? (Include extension)

```
vol.py -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64 timeliner |
grep "2019-03-07 23:06:58" -B 5 -A 5
```

This process is slow, so I recommend working on the next questions while it's at work, but here's the grepped output!

Answer: flag<Skype.exe>

14. lightbulb moment - 35 pts

What was written in notepad.exe in the time of the memory dump?

First things first, we need the PID of notepad.exe. Question 3 already asked us about it, it's **3032**.
To find the contents, I followed these instructions. There are other ways to do it, but this worked well for me. It will basically dump everything related to notepad.exe. To make it easily searchable, we string each file in the directory and direct the output to a single text file:

```
mkdir vads
vol.py -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64 vaddump -p 3032
strings -e l vads/* > vadsall.txt
```

In memory, we can look for contents of notepad files by searching for this string:
"Would you like to save the %% file non-transactionally?"

```
grep "Would you like to save" -A 10 vadsall.txt
```

There are two matches, and the second one contains the flag!

```
Would you like to save the %% file non-transactionally?
Text Document
New Text Document
Notepad"Do you want to save changes to %%?
&Save
Do&n't Save
,This computer is not running genuine Windows
Software Licensing9mshelp://windows/?id=6f58b1c8-e296-4498-8454-99558fde70bf-http://go.microsoft.com/fwlink/?Link
VS_VERSION_INFO
StringFileInfo
040904B0
--
Would you like to save the %% file non-transactionally?
ER\S-1-5-21-1497316740-3
9761-3945674337-1000
ws\s
pi.d
en-US
flag<REDBULL_IS_LIFE>
t.wi
C:\Windows\WinSxS\amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.17514_none_fa396087175ac9ac\
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
   Ln 1, Col 21
```

Answer: flag<REDBULL_IS_LIFE>

15. 8675309 - 35 pts

What is the shortname of the file at file record 59045?

```
vol.py -f /mnt/hgfs/mem/Triage-Memory.mem --profile=Win7SP1x64 mftparser > mft
strings mft.dmp | grep "Record Number: 59045" -B 10 -A 20
```

```
remnux@remnux:~/Documents/defcon19/mem$  strings mft.dmp | grep "Record Number: 59045" -B 10 -A 20
$DATA
$OBJECT_ID
Object ID: 40000000-0000-0000-0010-000000000000
Birth Volume ID: 19050000-0000-0000-1905-000000000000
Birth Object ID: 31015ed0-1900-ffff-ffff-ffff82794711
Birth Domain ID: ffffffff-8279-4711-0000-000000000000
**************************************************************************
**************************************************************************
MFT entry found at offset 0x2193d400
Attribute: In Use & File
Record Number: 59045
Link count: 2
$STANDARD_INFORMATION
Creation                       Modified                       MFT Altered                    Access Date                    Type
------------------------------ ------------------------------ ------------------------------ ------------------------------ ----
2019-03-17 06:50:07 UTC+0000 2019-03-17 07:04:43 UTC+0000   2019-03-17 07:04:43 UTC+0000   2019-03-17 07:04:42 UTC+0000   Archive
$FILE_NAME
Creation                       Modified                       MFT Altered                    Access Date                    Name/Path
------------------------------ ------------------------------ ------------------------------ ------------------------------ ---------
2019-03-17 06:50:07 UTC+0000 2019-03-17 07:04:43 UTC+0000   2019-03-17 07:04:43 UTC+0000   2019-03-17 07:04:42 UTC+0000   Users\Bob\DOCUME~1\EMPLOY~1\EMPLOY~1.XLS
$FILE_NAME
Creation                       Modified                       MFT Altered                    Access Date                    Name/Path
------------------------------ ------------------------------ ------------------------------ ------------------------------ ---------
2019-03-17 06:50:07 UTC+0000 2019-03-17 07:04:43 UTC+0000   2019-03-17 07:04:43 UTC+0000   2019-03-17 07:04:42 UTC+0000   Users\Bob\DOCUME~1\EMPLOY~1\EmployeeInformation.xlsx
$OBJECT_ID
Object ID: 00fe50d2-4841-e911-8751-000c2958bc5f
Birth Volume ID: 80000000-4800-0000-0100-000000000300
Birth Object ID: 00000000-0000-0000-0200-000000000000
Birth Domain ID: 40000000-0000-0000-0030-000000000000
$DATA
```

$OBJECT ID

The first $File_Name entry is the short name of the file: EMPLOY~1.xls

Answer: flag<EMPLOY~1.XLS>

16. whats-a-metasploit? - 50 pts

This box was exploited and is running meterpreter. What PID was infected?

Submitting the md5 hash (from question 8) to Virustotal confirms the executable is a Trojan virus. The affected PID is **3496**!