DEFCON 2019 MEMORY FORENSICS

Andrew Mayer

This is my personal write up of the 2019 DefCon Memory forensics challenge. This challenge was introduced at DefCon 2019 as part 4 of their 5 part CFT challenge for some of the best Cybersecurity professionals in the world to solve. Questions are precursed by their point value and the name of the flag challenge, and in bold.

get your volatility on - 5 pts
What is the SHA1 hash of triage.mem?
This was a simple question, but weird answer format. A simple Sha1sum command reveals the Sha sum of the file::

```
sansforensics@siftworkstation: ~/Desktop/work
$ sha1sum  Triage-Memory-001.mem
c95e8cc8c946f95a109ea8e47a6800de10a27abd  Triage-Memory-001.mem
```

pr0file - 10 pts
What profile is the most appropriate for this machine? (ex: Win10x86_14393)
this was also down to a single command, but this time we get to go into volatility for the first time. As with almost any memory image to analyze, the first command is imageinfo:

```
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Win2008R2SP1x64_23418, Win2008R2SP1x64,
Win7SP1x64_24000, Win7SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/sansforensics/Desktop/work/DefConCTF/Triage-Memory-001.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
```

hey, write this down - 12 pts
What was the process ID of notepad.exe?
One of the most useful commands in volatility to get started is pslist. You can ammend grep to the end, and find any PID, PPID, or process name, such as notepad. This is what I used, and the answer came very easy: 3032

```
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 pslist | grep notepad.exe
Volatility Foundation Volatility Framework 2.6.1
0xfffffa80054f9060 notepad.exe            3032   1432      1       60      1      0 2019-03-22 05:32:22 UTC+0000
```

wscript can haz children - 14 pts
Name the child processes of wscript.exe.

Another common command between linux and volatility is the pstree command. Using this with the -A option to show the preceeding lines after the searched term, I am able to see the child process: UwkpjFjDzM.exe

```
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 pstree | grep -A 10 wscript.exe
Volatility Foundation Volatility Framework 2.6.1
.. 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
```

tcpip settings - 18 pts

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

For this, I looked to the volatility command netscan, which scans for open connections at the time the image was taken, and saw the local ipv4 address to be 10.0.0.101

```
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 netscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Proto    Local Address                  Foreign Address      State          Pid      Owner          Created
0x13e057300        UDPv4    10.0.0.101:55736               *:*                                 2888     svchost.exe    2019-03-22 05:32
:20 UTC+0000
0x13e05b4f0        UDPv6    ::1:55735                      *:*                                 2888     svchost.exe    2019-03-22 05:32
:20 UTC+0000
0x13e05b790        UDPv6    fe80::7475:ef30:be18:7807:55734 *:*                                 2888     svchost.exe    2019-03-22 05:3
2:20 UTC+0000
0x13e05d4b0        UDPv6    fe80::7475:ef30:be18:7807:1900 *:*                                  2888     svchost.exe    2019-03-22 05:32
:20 UTC+0000
0x13e05dec0        UDPv4    127.0.0.1:55737                *:*                                 2888     svchost.exe    2019-03-22 05:32
:20 UTC+0000
0x13e05e3f0        UDPv4    10.0.0.101:1900                *:*                                 2888     svchost.exe    2019-03-22 05:32
:20 UTC+0000
0x13e05eab0        UDPv6    ::1:1900                       *:*                                 2888     svchost.exe    2019-03-22 05:32
:20 UTC+0000
0x13e064d70        UDPv4    127.0.0.1:1900                 *:*                                 2888     svchost.exe    2019-03-22 05:32
:20 UTC+0000
0x13e02bcf0        TCPv4    -:49220                        72.51.60.132:443     CLOSED         4048     POWERPNT.EXE
0x13e035790        TCPv4    -:49223                        72.51.60.132:443     CLOSED         4048     POWERPNT.EXE
0x13e036470        TCPv4    -:49224                        72.51.60.132:443     CLOSED         4048     POWERPNT.EXE
0x13e258010        UDPv4    127.0.0.1:55560                *:*                                 5116     wscript.exe    2019-03-22 05:35
:32 UTC+0000
0x13e305a50        UDPv4    0.0.0.0:5355                   *:*                                 232      svchost.exe    2019-03-22 05:32
:09 UTC+0000
0x13e360be0        UDPv4    0.0.0.0:63790                  *:*                                 504                     2019-03-22 05:45
:47 UTC+0000
0x13e490ec0        UDPv4    0.0.0.0:5355                   *:*                                 232      svchost.exe    2019-03-22 05:32
:09 UTC+0000
0x13e490ec0        UDPv6    :::5355                        *:*                                 232      svchost.exe    2019-03-22 05:32
:09 UTC+0000
0x13e5683e0        UDPv4    10.0.0.101:137                 *:*                                 4        System         2019-03-22 05:32
:06 UTC+0000
0x13e594250        UDPv4    10.0.0.101:138                 *:*                                 4        System         2019-03-22 05:32
:06 UTC+0000
0x13e597ec0        UDPv4    0.0.0.0:0                      *:*                                 232      svchost.exe    2019-03-22 05:32
:06 UTC+0000
```

intel - 18 pts

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

Yes, the netscan tool with volatility also lists the PID of the connection that is established. By running netscan and grepping for the known PID, the foreign IP comes out to 10.0.0.106.

```
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 netscan | grep 3496
Volatility Foundation Volatility Framework 2.6.1
0x13e397190        TCPv4    10.0.0.101:49217               10.0.0.106:4444      ESTABLISHED    3496     UWkpjFjDzM.exe
```

i <3 windows dependencies - 20 pts

What process name is VCRUNTIME140.dll associated with?

This one took me a few extra minutes, as that .dll is associated with multiple processes. The dlllist command through volatility shows dlls associated with each PID running, load time, size, and path, among other things. I had to run through and try each of them since there were multiple processes utilizing this .dll, but using the command (vol.py -f Triage-Memory-001.mem –profile=Win7SP1x64 dlllist | grep -B 7- VCRUNTIME140.dll), I finally found the correct answer to be OfficeClickToR

```
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 dlllist | grep -B 70 VCRUNTIME140.dll
Volatility Foundation Volatility Framework 2.6.1
0x000007fefc310000            0xc000                0x9 2019-03-22 05:32:05 UTC+0000   c:\Windows\system32\VERSION.dll
0x000007fefb030000           0x53000                0x1 2019-03-22 05:32:05 UTC+0000   c:\Windows\system32\fwpuclnt.dll
0x000007fefe790000            0x8000               0x11 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\NSI.dll
0x000007fefd260000           0x36000                0x1 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\CFGMGR32.dll
0x000007fefe560000           0x71000                0x3 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\SHLWAPI.dll
0x000007fefcd80000            0xb000                0x1 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\secur32.dll
0x000007fefc640000            0xa000                0x1 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\credssp.dll
0x000007fefc4f0000           0x1e000                0x3 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\USERENV.dll
0x000007fefd170000            0xf000                0x3 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\profapi.dll
0x000007fefc4d0000           0x1b000                0x4 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\GPAPI.dll
0x000007feff3f0000           0x4d000                0x8 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\WS2_32.dll
0x000007fefb140000           0x27000                0x3 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\IPHLPAPI.DLL
0x000007fefb100000            0xb000                0x3 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\WINNSI.DLL
0x000007fefaf70000           0x11000                0x1 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\dhcpcsvc6.DLL
0x000007fefaeb0000           0x18000                0x1 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\dhcpcsvc.DLL
0x000007fefc9e0000           0x55000                0x3 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\mswsock.dll
0x000007fefc3e0000            0x7000                0x1 2019-03-22 05:32:05 UTC+0000   C:\Windows\System32\wshtcpip.dll
0x000007fefc9d0000            0x7000                0x1 2019-03-22 05:32:05 UTC+0000   C:\Windows\System32\wshtp6.dll
0x000007fefa610000            0xa000                0x1 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\wfapigp.dll
0x000007fefaa50000           0x2d000                0x2 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\ntmarta.dll
0x000007feff390000           0x52000                0x2 2019-03-22 05:32:05 UTC+0000   C:\Windows\system32\WLDAP32.dll
0x000007fef9830000           0x2c000                0x1 2019-03-22 05:32:07 UTC+0000   c:\windows\system32\dps.dll
0x000007fefe6b0000           0xd7000                0x4 2019-03-22 05:32:07 UTC+0000   C:\Windows\system32\OLEAUT32.dll
0x000007fefe8d0000           0x99000                0x1 2019-03-22 05:32:07 UTC+0000   C:\Windows\system32\CLBCatQ.DLL
0x000007fefb660000          0x127000                0x1 2019-03-22 05:32:07 UTC+0000   C:\Windows\system32\taskschd.dll
0x000007fef5910000           0x19000                0x8 2019-03-22 05:32:14 UTC+0000   C:\Windows\system32\wdi.dll
0x000007fef5520000          0x14a000                0x3 2019-03-22 05:32:14 UTC+0000   C:\Windows\system32\diagperf.dll
0x000007fef53f0000            0x8000                0x1 2019-03-22 05:32:14 UTC+0000   C:\Windows\system32\pnpts.dll
0x000007fef7700000           0x74000                0x1 2019-03-22 05:32:14 UTC+0000   C:\Windows\System32\netprofm.dll
0x000007fefb7d0000           0x15000                0x1 2019-03-22 05:32:14 UTC+0000   C:\Windows\System32\nlaapi.dll
0x000007fefca40000           0x18000                0x1 2019-03-22 05:32:14 UTC+0000   C:\Windows\system32\CRYPTSP.dll
0x000007fefc740000           0x47000                0x1 2019-03-22 05:32:14 UTC+0000   C:\Windows\system32\rsaenh.dll
0x000007fef5420000            0xc000                0x1 2019-03-22 05:32:14 UTC+0000   C:\Windows\System32\npmproxy.dll
0x000007fef53d0000           0x1d000                0x4 2019-03-22 05:32:14 UTC+0000   C:\Windows\system32\radardt.dll
0x000007fefb2d0000           0x11000                0x4 2019-03-22 05:32:14 UTC+0000   C:\Windows\system32\WTSAPI32.dll
0x000007fef5300000            0xd000                0x1 2019-03-22 05:32:14 UTC+0000   C:\Windows\system32\wdiasqmmodule.dll
0x000007fefcb90000           0x22000                0x1 2019-03-22 05:32:15 UTC+0000   C:\Windows\system32\bcrypt.dll
**************************************************************************
OfficeClickToR pid:   1136
Command line : "C:\Program Files\Common Files\Microsoft Shared\ClickToRun\OfficeClickToRun.exe" /service
Service Pack 1
```

mal-ware-are-you - 20 pts

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

This needs to be done by isolating and dumping the process to a directory that it can be usable in. using the procdump command(vol.py -f Triage-Memory-001.mem –profile=Win7SP1x64 procdump -p 3496 –dump-dir=dump), volatility can dump the process 3496 into my dump directory. From there, all that is needed is a md5sum command to see the md5 hash:

```
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ ls dump
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 procdump -p 3496 --dump-dir=dump
Volatility Foundation Volatility Framework 2.6.1
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0xfffffa8005a1d9e0 0x0000000000400000 UWkpjFjDzM.exe       OK: executable.3496.exe
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ md5 dump/executable.3496.exe

Command 'md5' not found, did you mean:

  command 'mdl' from snap mdl (0.11.0)
  command 'mdp' from deb mdp
  command 'cd5' from deb cd5
  command 'mdu' from deb mtools

See 'snap info <snapname>' for additional versions.

sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ md dump/executable.3496.exe
md5deep           md5sum.textutils  mdatopbm          mdeltree          mdir              mdu
md5sum            mdadm             mdel              mdig              mdmon
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ md dump/executable.3496.exe
md5deep           md5sum.textutils  mdatopbm          mdeltree          mdir              mdu
md5sum            mdadm             mdel              mdig              mdmon
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ md5sum dump/executable.3496.exe
690ea20bc3bdfb328e23005d9a80c290  dump/executable.3496.exe
```

lm-get bobs hash - 24 pts

What is the LM hash of bobs account?

This one took a few more steps. First, I needed to find the virtual address of the SAM and system hives, which can be found using the hivedump command:

```
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 hivelist
Volatility Foundation Volatility Framework 2.6.1
Virtual            Physical           Name
------------------ ------------------ ----
0xfffff8a003ad2010 0x0000000125598010 \??\C:\System Volume Information\Syscache.hve
0xfffff8a00469c010 0x00000000a779d010 \SystemRoot\System32\Config\DEFAULT
0xfffff8a00000e010 0x00000000a9740010 [no name]
0xfffff8a000024010 0x00000000a97cb010 \REGISTRY\MACHINE\SYSTEM
0xfffff8a000053320 0x00000000a977a320 \REGISTRY\MACHINE\HARDWARE
0xfffff8a0000fe010 0x00000000a9625010 \SystemRoot\System32\Config\SECURITY
0xfffff8a0004db010 0x00000000a8599010 \Device\HarddiskVolume1\Boot\BCD
0xfffff8a00054b010 0x00000000a7fe3010 \SystemRoot\System32\Config\SOFTWARE
0xfffff8a000e66010 0x000000009ce84010 \SystemRoot\System32\Config\SAM
0xfffff8a000efe410 0x000000009be5c410 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0xfffff8a000f43010 0x000000009ba60010 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0xfffff8a00125d010 0x000000009635a010 \??\C:\Users\Bob\AppData\Local\Microsoft\Windows\UsrClass.dat
0xfffff8a0012ea010 0x0000000096937010 \??\C:\Users\Bob\ntuser.dat
```

Once I had these, hashdump, another volitality command, is used to dump the contents of hashes by using these(I could have used JUST the SAM *or* System location).by running this command, the hashes of all three accounts are found:

```
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 hashdump -s 0xfffff8a000e66010 -y 0xfffff8a000024010 > hashs.txt
Volatility Foundation Volatility Framework 2.6.1
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ cat hashs.txt
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Bob:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

vad the impaler - 25 pts

What protections does the VAD node at 0xfffffa800577ba10 have?

The VAD system is something I was not completely familiar with, but some quick reading and digging helped me understand it better. Volatility has a vadinfo command that shows multiple info on all the vad entries in the mem dump. Using this and grepping for the requested memory address, I was able to find the protection to be PAGE_READONLY.

```
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 vadinfo | grep -A 5 0xfffffa800577ba10
Volatility Foundation Volatility Framework 2.6.1
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
```

more vads?! - 25 pts

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

This was solved the same as the last question, using the vadinfo command and grepping for the requested vad table address. The answer revealed itself ot be PAGE_NOACCESS

```
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 vadinfo | grep -A 5 0x00000000033c0000
Volatility Foundation Volatility Framework 2.6.1
VAD node @ 0xfffffa800431cbf0 Start 0x00000000033c0000 End 0x00000000034bffff Tag VadS
Flags: CommitCharge: 4, PrivateMemory: 1, Protection: 4
Protection: PAGE_READWRITE
Vad Type: VadNone

VAD node @ 0xfffffa800548f610 Start 0x0000000003960000 End 0x00000000039dffff Tag VadS
--
VAD node @ 0xfffffa80052652b0 Start 0x00000000033c0000 End 0x00000000033dffff Tag VadS
Flags: CommitCharge: 32, PrivateMemory: 1, Protection: 24
Protection: PAGE_NOACCESS
Vad Type: VadNone

VAD node @ 0xfffffa8003f416d0 Start 0x00000000033a0000 End 0x00000000033bffff Tag VadS
```

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)

To do this, we needed a volatility command that allows us to see previously run terminal commands. After trying the incorrect command(Bash), a quick google search refreshed my memory that the command was cmdline. Grepping for vbs, the answer was revealed as vhjReUDEuumrX

```
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 Bash
Volatility Foundation Volatility Framework 2.6.1
ERROR   : volatility.debug    : You must specify something to do (try -h)
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 cmdline | grep vbs
Volatility Foundation Volatility Framework 2.6.1
Command line : "C:\Windows\System32\wscript.exe" //B //NOLOGO %TEMP%\vhjReUDEuumrX.vbs
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
```

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)
I was able to find this by using one of volatilitys registry parsing commands. After failing to find it via userassist, I tried shimcache and grepped for that specific time, and was able to find the application to be skype.exe:

```
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 shimcache | grep 23:06:58
Volatility Foundation Volatility Framework 2.6.1
2019-03-07 23:06:58 UTC+0000   \??\C:\Program Files (x86)\Microsoft\Skype for Desktop\Skype.exe
```

lightbulb moment - 35 pts
What was written in notepad.exe in the time of the memory dump?
I was expecting this to be easier than it was, as I know there is a notepad command in volatility to do this exact thing, but that command is not supported with this OS profile. So, what I had to do was dump the process and manually search for strings in it to find what was typed. First, a pslist to find the PID of notepad, which was revealed to be 3032. next up, a procdump of the process(vol.py -f Triage-Memory-001.mem –profile=Win7SP1x64 memdump -p 3032 –dump-dir notepad).

```
$ strings -e l 3032.dmp | grep flag
flag<REDBULL_IS_LIFE>
\Registry\Machine\Software\Microsoft\Windows nt\currentversion\appcompatflags\AIT
```

After this, it was a case of searching the process for a flag. I tried to just search with the command strings -e l 3032.dmp, (-e l is indicating the strings to search from 16 bit and little endisn). That was a wash, as there were MANY lines of strings that it output. I decided to give the word flag a shot as a grep string, and sure enough, the string was the first thing that it found, REDBULL_IS_LIFE:

```
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 memdump -p 3032 --dump-dir dump
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
Writing notepad.exe [  3032] to 3032.dmp
sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
```

8675309 - 35 pts
What is the shortname of the file at file record 59045?
To go about this, I needed to find *how to find* the file record. I knew of a few file system commands, and upon checking some volatility docs, I saw that mftparser displayed file record numbers. After narrowing down my grep command a bit, I was able to use the command vol.py -f Triage-Memory-001.mem –profile=Win7SP1x64 mftparser | grep -A 15 -B 2 59045 to find the filename, which is EMPLOY~1.XLS

```
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 mftparser | grep -A 15 -B 2  59045
Volatility Foundation Volatility Framework 2.6.1
0000000000: 7b 22 74 72 65 65 5f 73 69 7a 65 22 3a 35 33 30   {"tree_size":530
0000000010: 35 30 2c 22 74 69 6d 65 73 74 61 6d 70 22 3a 31   50,"timestamp":1
0000000020: 35 35 33 31 36 30 39 35 39 30 34 35 2c 22 73 68   553160959045,"sh
0000000030: 61 32 35 36 5f 72 6f 6f 74 5f 68 61 73 68 22 3a   a256_root_hash":
0000000040: 22 53 33 42 2f 45 6f 55 38 4a 33 76 57 61 56 6d   "S3B/EoU8J3vWaVm
0000000050: 51 61 36 30 2b 47 53 62 67 67 4c 70 46 68 49 47   Qa60+GSbggLpFhIG
0000000060: 38 7a 36 6c 6f 56 79 49 35 39 53 30 3d 22 2c 22   8z6loVyI59S0=","
0000000070: 74 72 65 65 5f 68 65 61 64 5f 73 69 67 6e 61 74   tree_head_signat
0000000080: 75 72 65 22 3a 22 42 41 45 42 41 45 64 61 79 49   ure":"BAEBAEdayI
0000000090: 6a 58 6f 39 45 43 77 52 2b 36 71 74 71 50 43 43   jXo9ECwR+6qtqPCC
00000000a0: 37 71 35 59 42 37 35 2b 6d 31 32 56 63 63 4c 52   7q5YB75+m12VccLR
00000000b0: 73 7a 78 68 48 52 2b 72 33 38 66 48 6a 67 38 76   szxhHR+r38fHjg8v
00000000c0: 57 78 42 2f 66 31 59 44 31 75 55 45 46 54 31 62   WxB/f1YD1uUEFT1b
00000000d0: 68 38 79 33 53 41 59 6e 61 71 57 77 6f 55 46 49   h8y3SAYnaqWwoUFI
00000000e0: 76 38 63 71 44 76 78 2b 50 79 67 71 6a 76 68 42   v8cqDvx+PygqjvhB
00000000f0: 6e 5a 45 57 31 33 6d 44 76 30 2b 6a 42 6d 35 59   nZEW13mDv0+jBm5Y
0000000100: 43 68 59 36 55 4d 4b 4f 49 64 52 35 54 4d 31 35   ChY6UMKOIdR5TM15
0000000110: 72 4a 41 54 37 41 56 79 78 69 31 46 2f 36 51 36   rJAT7AVyxi1F/6Q6
--
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\DOCUM
E~1\EMPLOY~1\EMPLOY~1.XLS

$FILE_NAME
```

whats-a-metasploit? - 50 pts

This box was exploited and is running meterpreter. What PID was infected?
I looked at this very straightforward. If metasploit is being used, volatility can probably find what it affected using malfind. I hit it with "vol.py -f Triage-Memory-001.mem –profile=Win7SP1x64 malfind", and the last answer down was a nice looking process called UwkpjFjDzM.exe, with PID 3496. I gave it a shot as the flag, and sure enough it was right.

```
Process: UWkpjFjDzM.exe Pid: 3496 Address: 0x2ad0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 55, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x02ad0000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x02ad0010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x02ad0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x02ad0030  00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00   ................

0x02ad0000 4d              DEC EBP
0x02ad0001 5a              POP EDX
0x02ad0002 90              NOP
0x02ad0003 0003            ADD [EBX], AL
0x02ad0005 0000            ADD [EAX], AL
0x02ad0007 000400          ADD [EAX+EAX], AL
0x02ad000a 0000            ADD [EAX], AL
0x02ad000c ff              DB 0xff
0x02ad000d ff00            INC DWORD [EAX]
0x02ad000f 00b800000000    ADD [EAX+0x0], BH
0x02ad0015 0000            ADD [EAX], AL
0x02ad0017 004000          ADD [EAX+0x0], AL
0x02ad001a 0000            ADD [EAX], AL
0x02ad001c 0000            ADD [EAX], AL
0x02ad001e 0000            ADD [EAX], AL
0x02ad0020 0000            ADD [EAX], AL
0x02ad0022 0000            ADD [EAX], AL
0x02ad0024 0000            ADD [EAX], AL
0x02ad0026 0000            ADD [EAX], AL
0x02ad0028 0000            ADD [EAX], AL
0x02ad002a 0000            ADD [EAX], AL
0x02ad002c 0000            ADD [EAX], AL
0x02ad002e 0000            ADD [EAX], AL
0x02ad0030 0000            ADD [EAX], AL
0x02ad0032 0000            ADD [EAX], AL
0x02ad0034 0000            ADD [EAX], AL
0x02ad0036 0000            ADD [EAX], AL
0x02ad0038 0000            ADD [EAX], AL
0x02ad003a 0000            ADD [EAX], AL
0x02ad003c 0001            ADD [ECX], AL
0x02ad003e 0000            ADD [EAX], AL

sansforensics@siftworkstation: ~/Desktop/work/DefConCTF
$ vol.py -f Triage-Memory-001.mem --profile=Win7SP1x64 malfind
```

To conclude this challenge, I found it to be around a 7/10 difficulty for me. I had to do some research on a few concepts and lookup correct commands, but was able to finish it in a decent time of around 5 hours. It allowed me to learn about some new concepts in memory and new commands with volatility, and gave me a nice sense of accomplishment in being able to solve it relatively easily and without help. I really do enjoy memory forensics more than most other things in information security, and can’t wait to find another challenge like this to face.