University of Utah
School of Computing
CS 6967: Security Operations

# Lab 11 Response Outline

## Lab 11: Threat Hunting with Memory Dump

**Test 1: Capture a Memory Dump**

**Summary of test experience**:

For this test, I used the Winpmem memory acquisition tool to capture a memory dump of the Windows 10 VM.

**Screenshot of test results:**

Administrator: Command Prompt

```
C:\Users\yeaseen\Downloads\mem>.\winpmem_mini_x64_rc2.exe memdump.raw
WinPmem64
Extracting driver to C:\Users\yeaseen\AppData\Local\Temp\pme9A82.tmp
Driver Unloaded.
Loaded Driver C:\Users\yeaseen\AppData\Local\Temp\pme9A82.tmp.
Deleting C:\Users\yeaseen\AppData\Local\Temp\pme9A82.tmp
The system time is: 02:14:36
Will generate a RAW image
 - buffer_size_: 0x1000
CR3: 0x00001AA002
 4 memory ranges:
Start 0x00001000 - Length 0x0009E000
Start 0x00100000 - Length 0x00002000
Start 0x00103000 - Length 0xDFEED000
Start 0x100000000 - Length 0x20000000
max_physical_memory_ 0x120000000
Acquitision mode PTE Remapping
Padding from 0x00000000 to 0x00001000
pad
 - length: 0x1000

00% 0x00000000 .
copy_memory
 - start: 0x1000
 - end: 0x9f000

00% 0x00001000 .
Padding from 0x0009F000 to 0x00100000
pad
```

```
00% 0x00100000 .
Padding from 0x00102000 to 0x00103000
pad
 - length: 0x1000

00% 0x00102000 .
copy_memory
 - start: 0x103000
 - end: 0xdfff0000

00% 0x00103000 ..................................................
17% 0x32103000 ..................................................
34% 0x64103000 ..................................................
52% 0x96103000 ..................................................
69% 0xC8103000 ......................
Padding from 0xDFFF0000 to 0x100000000
pad
 - length: 0x20010000

77% 0xDFFF0000 ...............................
copy_memory
 - start: 0x100000000
 - end: 0x120000000

88% 0x100000000 ..............................
The system time is: 02:14:55
Driver Unloaded.
```

**Provide a brief (1–2 sentences) explanation of the results answering the question: How does the image demonstrate that you completed the test?**

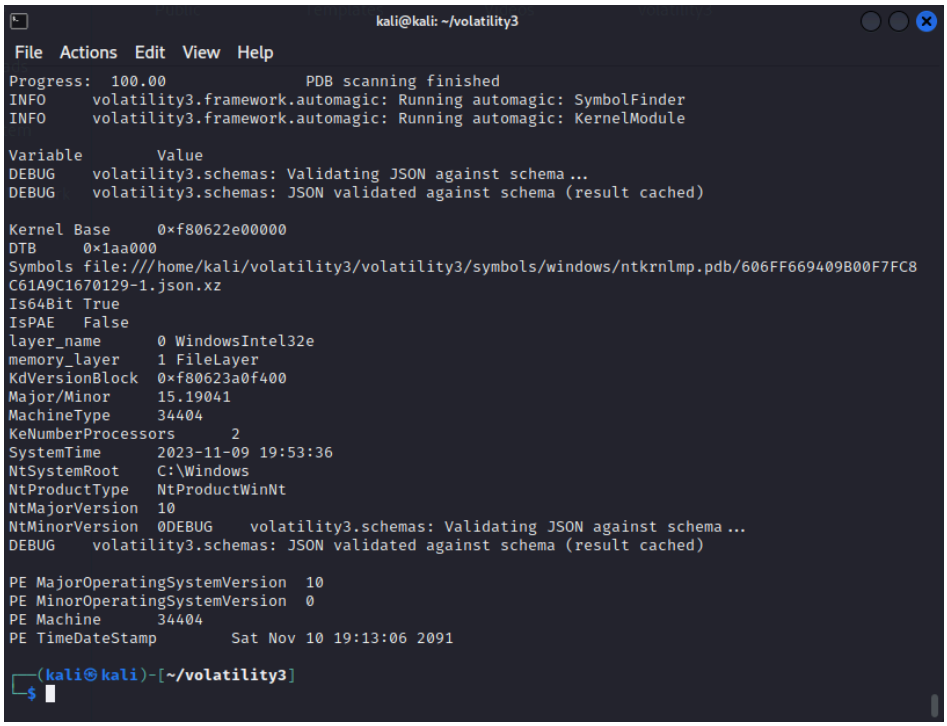
The above image shows that the memory acquisition is done successfully.

**Test 2: Identify OS Information**

**Summary of test experience**:

Here, I utilized the windows.info plugin of volatility3 to get the OS information from the memory dump.

**Screenshot of test results:**




**Provide a brief (1–2 sentences) explanation of the results answering the question: How does the image demonstrate that you completed the test?**

From the above image, we can see that volatility found a Device Tree Blob of a Kernel Base, the OS is 64 bit, it is not physically extensible, the kernel layer is WindowsIntel32e, and the root location is C:\Windows.

**Test 3: List the Running Processes**

**Summary of test experience**:

Here, I utilized the windows.pslist.PsList plugin of volatility3 to get the list of running processes from the memory dump.

**Screenshot of test results:**

```
kali@kali: ~/volatility3
File  Actions  Edit  View  Help
└─$ python3 vol.py -f ~/Desktop/memdump.raw windows.pslist.PsList
Volatility 3 Framework 2.5.0
Progress:  100.00               PDB scanning finished
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime
ExitTime        File output

4       0       System  0x ba0d77269080  115     -       N/A     False   2023-11-09 19:40:04.000000
        N/A     Disabled
92      4       Registry        0x ba0d773b9040  4       -       N/A     False   2023-11-09 19:40:0
2.000000        N/A     Disabled
336     4       smss.exe        0x ba0d77d7e040  2       -       N/A     False   2023-11-09 19:40:0
4.000000        N/A     Disabled
428     416     csrss.exe       0x ba0d7c285080  10      -       0       False   2023-11-09 19:40:0
9.000000        N/A     Disabled
504     416     wininit.exe     0x ba0d7ca37080  1       -       0       False   2023-11-09 19:40:0
9.000000        N/A     Disabled
512     496     csrss.exe       0x ba0d7ca3c140  12      -       1       False   2023-11-09 19:40:0
9.000000        N/A     Disabled
604     496     winlogon.exe    0x ba0d7ca77080  3       -       1       False   2023-11-09 19:40:1
0.000000        N/A     Disabled
620     504     services.exe    0x ba0d7ca7b080  6       -       0       False   2023-11-09 19:40:1
0.000000        N/A     Disabled
652     504     lsass.exe       0x ba0d7caa1080  9       -       0       False   2023-11-09 19:40:1
0.000000        N/A     Disabled
764     620     svchost.exe     0x ba0d7cb12240  12      -       0       False   2023-11-09 19:40:1
0.000000        N/A     Disabled
776     504     fontdrvhost.ex  0x ba0d7cb24140  5       -       0       False   2023-11-09 19:40:1
0.000000        N/A     Disabled
784     604     fontdrvhost.ex  0x ba0d7cb27140  5       -       1       False   2023-11-09 19:40:1
0.000000        N/A     Disabled
892     620     svchost.exe     0x ba0d7cb712c0  8       -       0       False   2023-11-09 19:40:1
0.000000        N/A     Disabled
940     620     svchost.exe     0x ba0d7cbb1240  5       -       0       False   2023-11-09 19:40:1
0.000000        N/A     Disabled
```

**Provide a brief (1–2 sentences) explanation of the results answering the question: How does the image demonstrate that you completed the test?**

The above screenshot is listing out all the current processes running in the memory dump.

**Test 4: List Network Connections**

**Summary of test experience**:

Here, I utilized the windows.netscan.NetScan plugin of volatility3 to get the list of network connections from the memory dump.

**Screenshot of test results:**

```
kali@kali: ~/volatility3
File  Actions  Edit  View  Help
Volatility 3 Framework 2.5.0
Progress:  100.00               PDB scanning finished
Offset  Proto   LocalAddr       LocalPort       ForeignAddr     ForeignPort     State   PID     Ow
ner     Created

0×ba0d772881b0  TCPv4   192.168.56.106  139     0.0.0.0 0       LISTENING       4       System  20
23-11-09 19:40:10.000000
0×ba0d772885d0  TCPv4   0.0.0.0 49668   0.0.0.0 0       LISTENING       2244    spoolsv.exe     20
23-11-09 19:40:10.000000
0×ba0d77288730  TCPv4   0.0.0.0 49668   0.0.0.0 0       LISTENING       2244    spoolsv.exe     20
23-11-09 19:40:10.000000
0×ba0d77288730  TCPv6   ::      49668   ::      0       LISTENING       2244    spoolsv.exe     20
23-11-09 19:40:10.000000
0×ba0d7769b5d0  TCPv4   0.0.0.0 49665   0.0.0.0 0       LISTENING       504     wininit.exe     20
23-11-09 19:40:10.000000
0×ba0d7769b5d0  TCPv6   ::      49665   ::      0       LISTENING       504     wininit.exe     20
23-11-09 19:40:10.000000
0×ba0d77ae8050  TCPv4   0.0.0.0 5040    0.0.0.0 0       LISTENING       3884    svchost.exe     20
23-11-09 19:40:13.000000
0×ba0d786df050  TCPv4   0.0.0.0 49669   0.0.0.0 0       LISTENING       620     services.exe    20
23-11-09 19:40:11.000000
0×ba0d786df1b0  TCPv4   0.0.0.0 49664   0.0.0.0 0       LISTENING       652     lsass.exe       20
23-11-09 19:40:10.000000
0×ba0d786df890  TCPv4   0.0.0.0 49664   0.0.0.0 0       LISTENING       652     lsass.exe       20
23-11-09 19:40:10.000000
0×ba0d786df890  TCPv6   ::      49664   ::      0       LISTENING       652     lsass.exe       20
23-11-09 19:40:10.000000
0×ba0d786dfb50  TCPv4   0.0.0.0 135     0.0.0.0 0       LISTENING       892     svchost.exe     20
23-11-09 19:40:10.000000
0×ba0d786dfb50  TCPv6   ::      135     ::      0       LISTENING       892     svchost.exe     20
23-11-09 19:40:10.000000
0×ba0d786dfcb0  TCPv4   0.0.0.0 49666   0.0.0.0 0       LISTENING       1068    svchost.exe     20
23-11-09 19:40:10.000000
0×ba0d786e0230  TCPv4   0.0.0.0 49667   0.0.0.0 0       LISTENING       1284    svchost.exe     20
```

**Provide a brief (1–2 sentences) explanation of the results answering the question: How does the image demonstrate that you completed the test?**

The above screenshot is listing out all the network connections with IP address and port number.

**Test 5: Establish Reverse Shell on the Windows VM**

**Summary of test experience**:

In Test 5, I have to establish a reverse shell on the windows machine VM. So, first I opened a port to listen to everything on that port. Later I ran the reverse shell script connecting to the Kali. Finally, I got Windows shell access from Kali.

**Screenshot of test results:**

```
kali@kali: ~
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ nc -lvp 1234
listening on [any] 1234 ...
10.0.2.4: inverse host lookup failed: Unknown host
connect to [10.0.2.5] from (UNKNOWN) [10.0.2.4] 49864
> dir
0409 AdvancedInstallers am-et AppLocker appraiser AppV ar-SA bg-BG Boot Bthprops CatRoot catroot2
 CodeIntegrity Com config Configuration ContainerSettingsProviders cs-CZ da-DK DDFs de-DE DiagSvc
s Dism downlevel drivers DriverState DriverStore dsc el-GR en en-GB en-US es-ES es-MX et-EE F12 f
f-Adlm-SN fi-FI fr-CA fr-FR FxsTmp GroupPolicy GroupPolicyUsers he-IL hr-HR hu-HU Hydrogen ias ic
sxml IME inetsrv InputMethod Ipmi it-IT ja-jp Keywords ko-KR Licenses LogFiles Logs lt-LT lv-LV M
ailContactsCalendarSync Microsoft migration migwiz MRT MSDRM MsDtc MUI my-mm nb-NO NDF networklis
t nl-NL Nui oobe OpenSSH osa-Osge-001 PerceptionSimulation pl-PL PointOfService Printing_Admin_Sc
ripts ProximityToast pt-BR pt-PT ras RasToast Recovery restore ro-RO ru-RU SecureBootUpdates setu
p Sgrm ShellExperiences si-lk sk-SK sl-SI SleepStudy slmgr SMI Speech Speech_OneCore spool spp sp
pui sr-Latn-RS sru sv-SE Sysprep SystemResetPlatform ta-in ta-lk Tasks th-TH ti-et tr-TR uk-UA UN
P wbem WCN WDI WinBioDatabase WinBioPlugIns WindowsPowerShell winevt WinMetadata winrm zh-CN zh-T
W 69fe178f-26e7-43a9-aa7d-2b616b672dde_eventlogservice.dll 6bea57fb-8dfb-4177-9ae8-42e8b3529933_R
untimeDeviceInstall.dll @AdvancedKeySettingsNotification.png @AppHelpToast.png @AudioToastIcon.pn
g @BackgroundAccessToastIcon.png @bitlockertoastimage.png @edptoastimage.png @EnrollmentToastIcon
.png @language_notification_icon.png @optionalfeatures.png @StorageSenseToastIcon.png @VpnToastIc
on.png @windows-hello-V4.1.gif @WindowsHelloFaceToastIcon.png @WindowsUpdateToastIcon.contrast-bl
ack.png @WindowsUpdateToastIcon.contrast-white.png @WindowsUpdateToastIcon.png @WirelessDisplayTo
ast.png @WLOGO_48×48.png aadauthhelper.dll aadcloudap.dll aadjcsp.dll aadtb.dll aadWamExtension.d
ll AarSvc.dll AboutSettingsHandlers.dll AboveLockAppHost.dll accessibilitycpl.dll accountaccessor
.dll AccountsRt.dll AcGenral.dll AcLayers.dll acledit.dll aclui.dll acmigration.dll ACPBackground
ManagerPolicy.dll acppage.dll acproxy.dll AcSpecfc.dll ActionCenter.dll ActionCenterCPL.dll Actio
nQueue.dll ActivationClient.dll ActivationManager.dll activeds.dll activeds.tlb ActiveHours.png A
ctiveSyncCsp.dll ActiveSyncProvider.dll actxprxy.dll AcWinRT.dll AcXtrnal.dll AdaptiveCards.dll A
ddressParser.dll adhapi.dll adhsvc.dll AdmTmpl.dll adprovider.dll adrclient.dll adsldp.dll adsldp
c.dll adsmsext.dll adsnt.dll adtschema.dll AdvancedEmojiDS.dll advapi32.dll advapi32res.dll advpa
ck.dll aeevts.dll aeinv.dll aemarebackup.dll aepic.dll agentactivationruntime.dll agentactivation
runtimestarter.exe agentactivationruntimewindows.dll AgentService.exe aitstatic.exe AJRouter.dll
alg.exe altspace.dll amcompat.tlb amsi.dll amsiproxy.dll amstream.dll Analog.Shell.Broker.dll Ana
```

**Provide a brief (1–2 sentences) explanation of the results answering the question: How does the image demonstrate that you completed the test?**

From the above image, it can be seen that on Kali's terminal, I ran 'dir' windows shell command to list out the files and folders under the current directory.

### Test 6: Find Abnormal Network Connections

**Summary of test experience**:

First, I dumped all the network connections to a text file using the windows.netscan.NetScan plugin of volatility 3. Then from the file, I collected the connection issued by the browser which is msedge.exe. Finally, from the previous result, I looked for the connections issued by the native service host.

**Screenshot of test results:**

```
0×bf8c1384aa20  TCPv4   192.168.56.106  49870   13.107.246.254  443     ESTABLISHED     3588    S
earchApp.exe    2023-11-09 20:30:12.000000
0×bf8c13e65a20  TCPv4   192.168.56.106  49863   20.62.149.92    443     CLOSED  2676    MsMpEng.e
xe      2023-11-09 20:28:26.000000
0×bf8c13f31010  TCPv4   192.168.56.106  49864   10.0.2.5        1234    ESTABLISHED     8548    p
owershell.exe   2023-11-09 20:28:27.000000
0×bf8c13f5f010  TCPv4   192.168.56.106  49867   20.50.80.214    443     ESTABLISHED     3588    S
earchApp.exe    2023-11-09 20:30:07.000000
0×bf8c1412b4e0  TCPv4   192.168.56.106  49869   72.21.81.200    443     ESTABLISHED     3588    S
earchApp.exe    2023-11-09 20:30:12.000000
```

**Provide a brief (1–2 sentences) explanation of the results answering the question: How does the image demonstrate that you completed the test?**

And I found that a powershell is executed from a network connection establishment, which is an unusual thing.