./L2L Memory Forensics

```
> $ cat PersonalInfo.txt
> Name: Ji Cheng
> Major: Information Security
> Year: Year 3
```

What is memory forensics?

- Analysis of volatile data in a computer’s memory dump
- Volatile data is the data stored in temporary memory on a computer while running
- Memory dump is a snapshot capture of a computer memory data from a specific instant
- Data can include browsing history, chat messages and clipboard contents

Volatility for volatile data

- Volatility is a framework to analyse memory dump and we will be primarily using this framework for the rest of this presentation
- Volatility comes in two version: Volatility 2 and Volatility 3
- Volatility 2 will be focused in this presentation
- So let us get started in the world of memory forensics!

```
> git clone https://github.com/volatilityfoundation/volatility.git

> cd [path_to_folder_download]
```

Extra Libraries (So that you don’t have to pull your hair)

- Some extra libraries will need to be downloaded. Since Volatility 2 is written using Python 2, these libraries will be downloaded using pip

```
> pip install pycrypto
```

```
> pip install distorm3
```

- If you have moved away from python 2, use this link to follow to download Python 2 and pip: https://linuxize.com/post/how-to-install-pip-on-ubuntu-20.04/
- For errors when installing using pip, follow this link https://stackoverflow.com/questions/26053982/setup-script-exited-with-error-command-x86-64-linux-gnu-gcc-failed-with-exit

Materials used

Memory Files Used:

- Govtech Stack the Flag CTF 2020 Memory File
- OtterCTF 2018 Memory File

References Used:

- Peter M Stewart Website: https://www.petermstewart.net/
- Andrea Fortuna Volatility Cheat Sheets

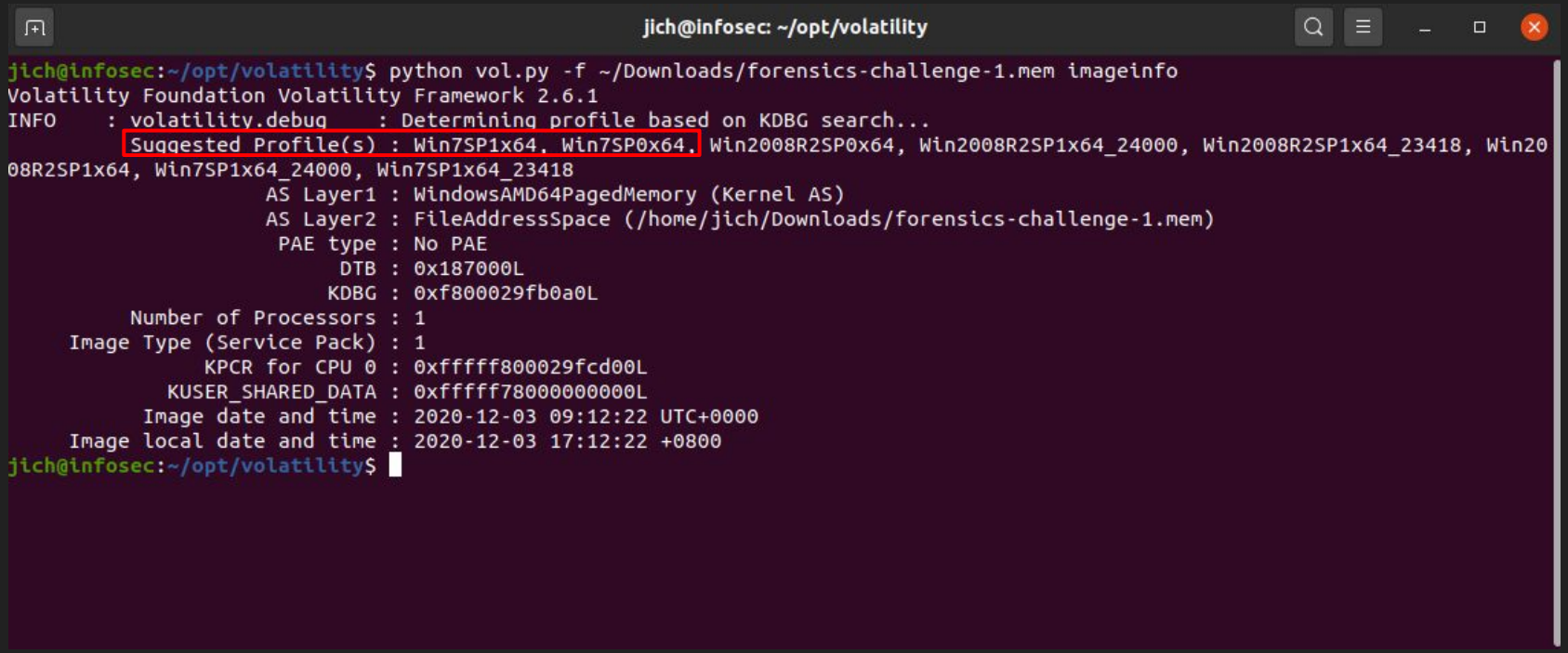
The beginning… Plugins!

- Before we can start running analytics on the memory file, we must first find out what kind of memory image we are working on
- This can be found using the imageinfo plugin

```
> python vol.py -f [Memory_Dump_File] imageinfo
```

- As seen, we can determine the profile of the memory dump. For subsequent scans, we will utilise the --profile=[Profile_Name] options to specify the options

```
> python vol.py -f [Memory_Dump_File] --profile=Win7SP1x64 [Plugin_Name]
```

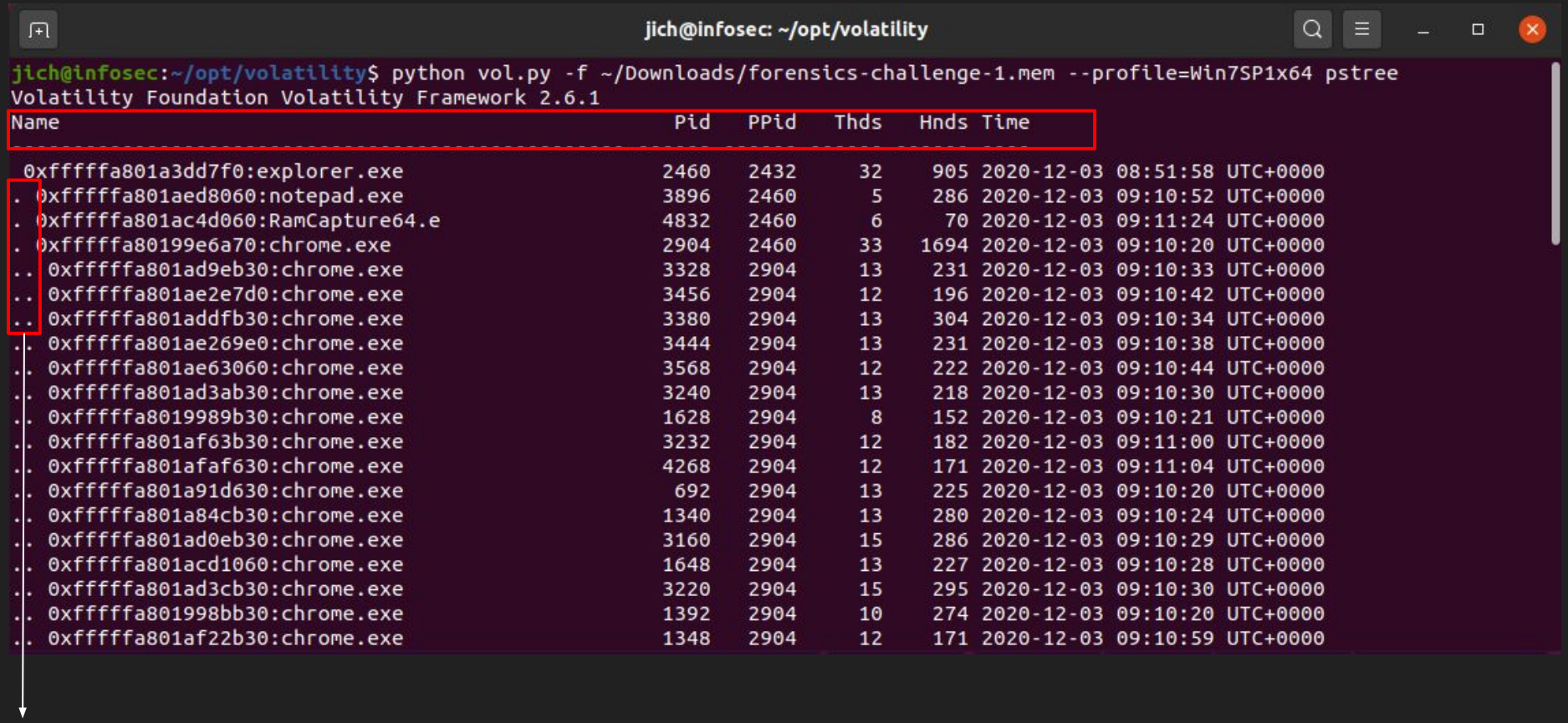

jich@infosec: ~/opt/volatility
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/forensics-challenge-1.mem imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_24000, Win7SP1x64_23418
AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
AS Layer2 : FileAddressSpace (/home/jich/Downloads/forensics-challenge-1.mem)
PAE type : No PAE
DTB : 0x187000L
KDBG : 0xf800029fb0a0L
Number of Processors : 1
Image Type (Service Pack) : 1
KPCR for CPU 0 : 0xfffff800029fcd00L
KUSER_SHARED_DATA : 0xfffff78000000000L
Image date and time : 2020-12-03 09:12:22 UTC+0000
Image local date and time : 2020-12-03 17:12:22 +0800
jich@infosec:~/opt/volatility$

Trust the process

- After identifying the kind of memory dump profile, we will proceed to check the list of running processes in the memory dump
- Done by issuing the pstree plugin
- There are also a few other plugins related to processes

pslist	Print all running processes
psscan	Pool scanner for process objects. This can find processes previously terminated and processes that have been hidden
psxview	Find hidden processes with various process listing

```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] pstree
```

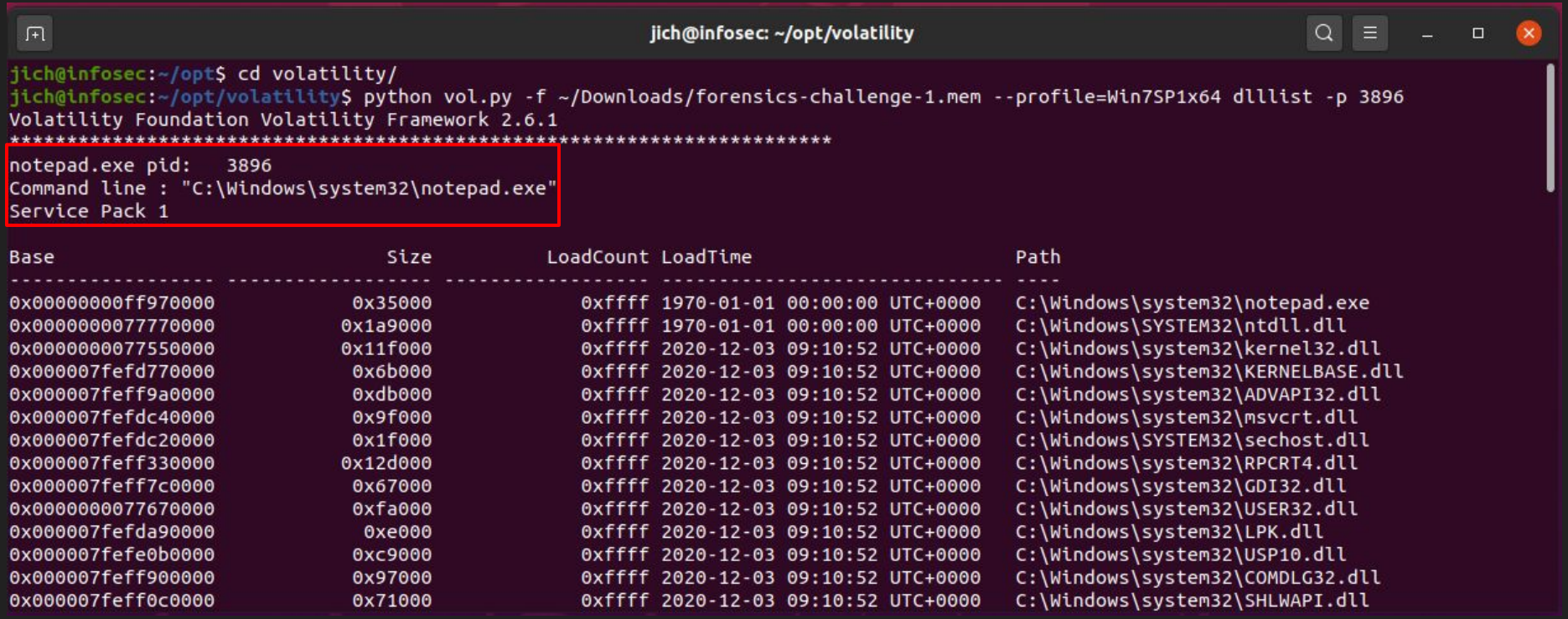


Parent-Child Processes

Depending on Windows (DLLs)

- DLLs stand for dynamic link library which are the libraries imported by processes
- Volatility plugin to use is dlllist. The option -p can be specified to show DLLs of a specific process

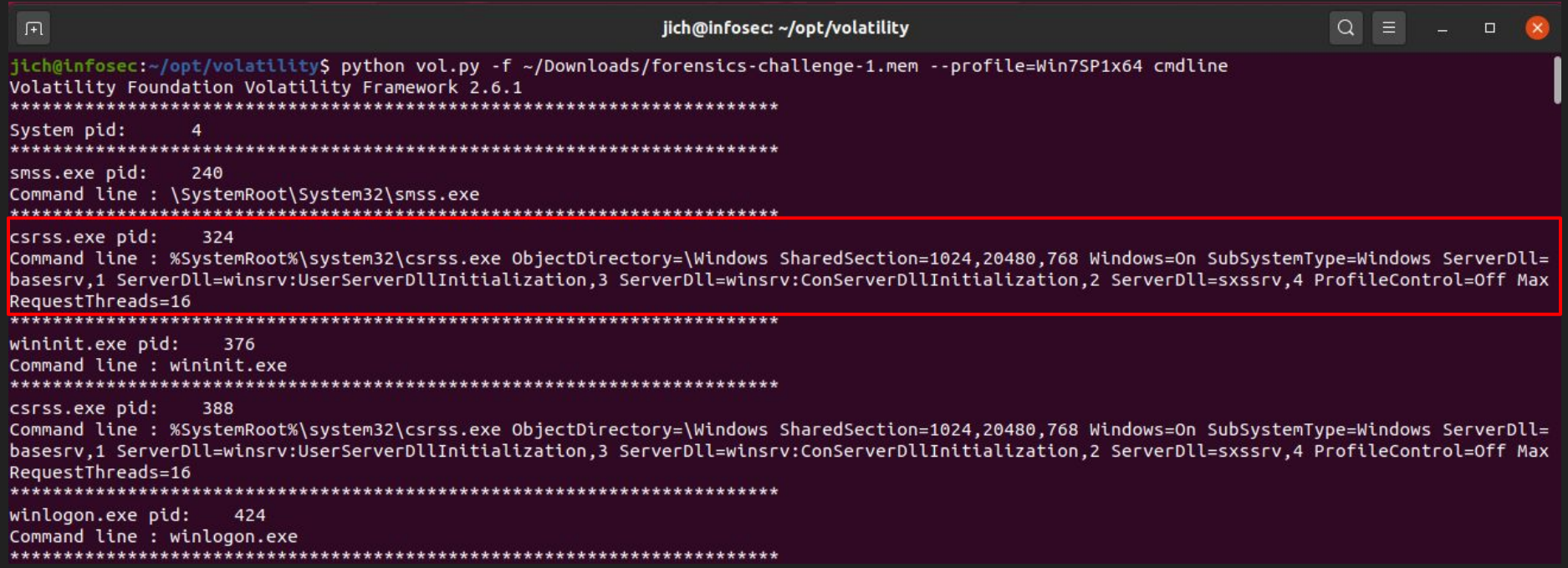
```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] dlllist -p [PID]
```


jich@infosec: ~/opt/volatility
jich@infosec:~/opt$ cd volatility/
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/forensics-challenge-1.mem --profile=Win7SP1x64 dlllist -p 3896
Volatility Foundation Volatility Framework 2.6.1
notepad.exe pid: 3896
Command line : "C:\Windows\system32\notepad.exe"
Service Pack 1
Base Size LoadCount LoadTime Path
0x00000000ff970000 0x35000 0xffff 1970-01-01 00:00:00 UTC+0000 C:\Windows\system32\notepad.exe
0x0000000077770000 0x1a9000 0xffff 1970-01-01 00:00:00 UTC+0000 C:\Windows\SYSTEM32\ntdll.dll
0x0000000077550000 0x11f000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\system32\kernel32.dll
0x000007fefd770000 0x6b000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\system32\KERNELBASE.dll
0x000007feff9a0000 0xdb000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\system32\ADVAPI32.dll
0x000007fefdc40000 0x9f000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\system32\msvcrt.dll
0x000007fefdc20000 0x1f000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\SYSTEM32\sechost.dll
0x000007feff330000 0x12d000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\system32\RPCRT4.dll
0x000007feff7c0000 0x67000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\system32\GDI32.dll
0x0000000077670000 0xfa000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\system32\USER32.dll
0x000007fefda90000 0xe000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\system32\LPK.dll
0x000007fefe0b0000 0xc9000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\system32\USP10.dll
0x000007feff900000 0x97000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\system32\COMDLG32.dll
0x000007feff0c0000 0x71000 0xffff 2020-12-03 09:10:52 UTC+0000 C:\Windows\system32\SHLWAPI.dll

Your wish is my command

- Volatility can also search memory dumps of commands that users have entered in the console shell
- Main plugin to use is cmdline which displays the commands issued on the console
- For Linux images, the linux_bash is the plugin to be used

```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] cmdline
```


jich@infosec: ~/opt/volatility
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/forensics-challenge-1.mem --profile=Win7SP1x64 cmdline
Volatility Foundation Volatility Framework 2.6.1
****
System pid: 4
****
smss.exe pid: 240
Command line : \SystemRoot\System32\smss.exe
****
csrss.exe pid: 324
Command line : %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,20480,768 Windows=On SubSystemType=Windows ServerDll=basesrv,1 ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:ConServerDllInitialization,2 ServerDll=sxssrv,4 ProfileControl=Off MaxRequestThreads=16
****
wininit.exe pid: 376
Command line : wininit.exe
****
csrss.exe pid: 388
Command line : %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,20480,768 Windows=On SubSystemType=Windows ServerDll=basesrv,1 ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:ConServerDllInitialization,2 ServerDll=sxssrv,4 ProfileControl=Off MaxRequestThreads=16
****
winlogon.exe pid: 424
Command line : winlogon.exe
****

Copycat

- We can also view what is copied on the clipboard on the memory dump
- This is done using the clipboard plugin

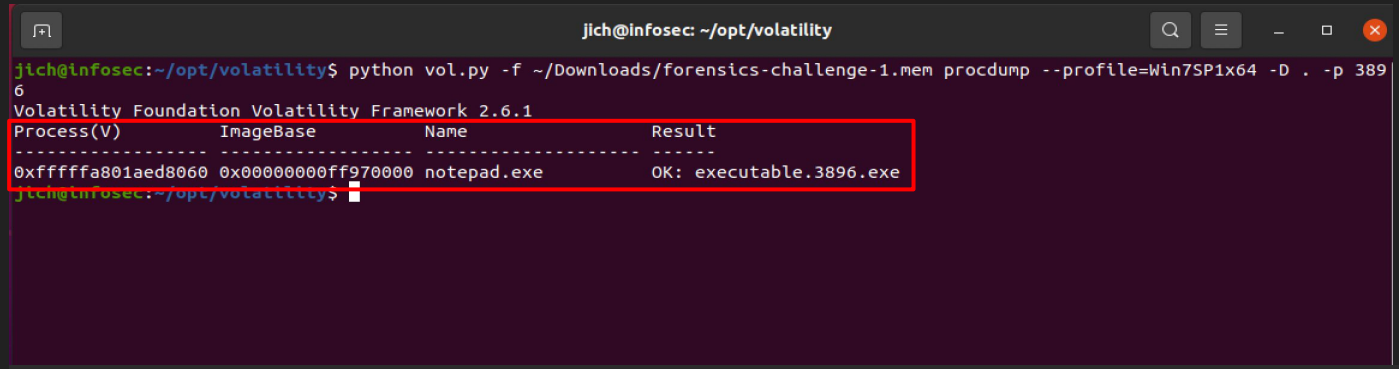
```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] clipboard
```

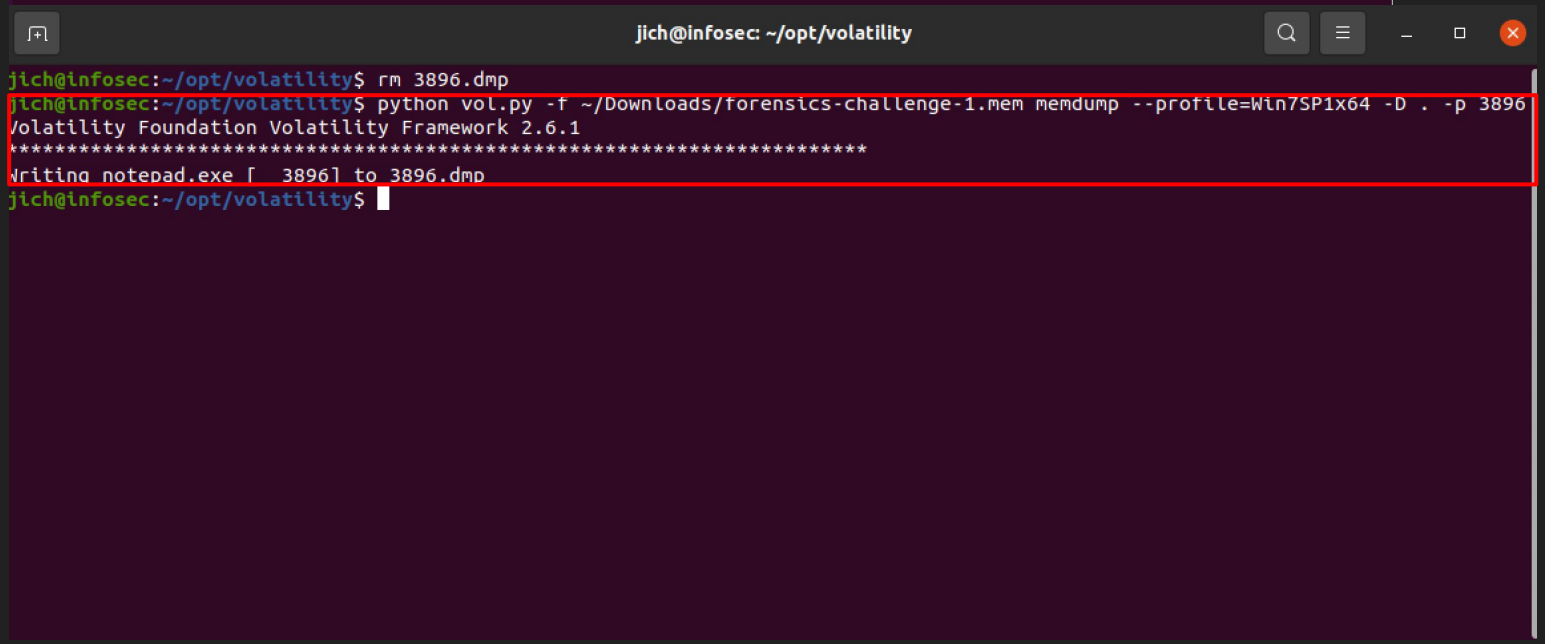
```
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64 clipboard
Volatility Foundation Volatility Framework 2.6.1
Session    WindowStation Format                         Handle Object             Data
---------- ------------- ------------------ ------------------ ------------------ ------------------------------------------------
         1 WinSta0       CF_UNICODETEXT               0x602e3 0xfffff900c1ad93f0 M@il_Pr0vid0rs
         1 WinSta0       CF_TEXT                         0x10 ------------------
         1 WinSta0       0x150133L              0x200000000000 ------------------
         1 WinSta0       CF_TEXT                          0x1 ------------------
         1 ------------- ------------------           0x150133 0xfffff900c1c1adc0
jich@infosec:~/opt/volatility$
```

No Dump Zone?

- After identifying the process ID, we might want to dump that process' executable
- This is done through using the procdump plugin
- To dump all memory resident address of a process, use memdump plugin instead

```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] procdump -D [Directory] -p [PID]
```


jich@infosec: ~/opt/volatility
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/forensics-challenge-1.mem procdump --profile=Win7SP1x64 -D . -p 389
6
Volatility Foundation Volatility Framework 2.6.1
Process(V) ImageBase Name Result
0xfffffa801aed8060 0x00000000ff970000 notepad.exe OK: executable.3896.exe
jich@infosec:~/opt/volatility$

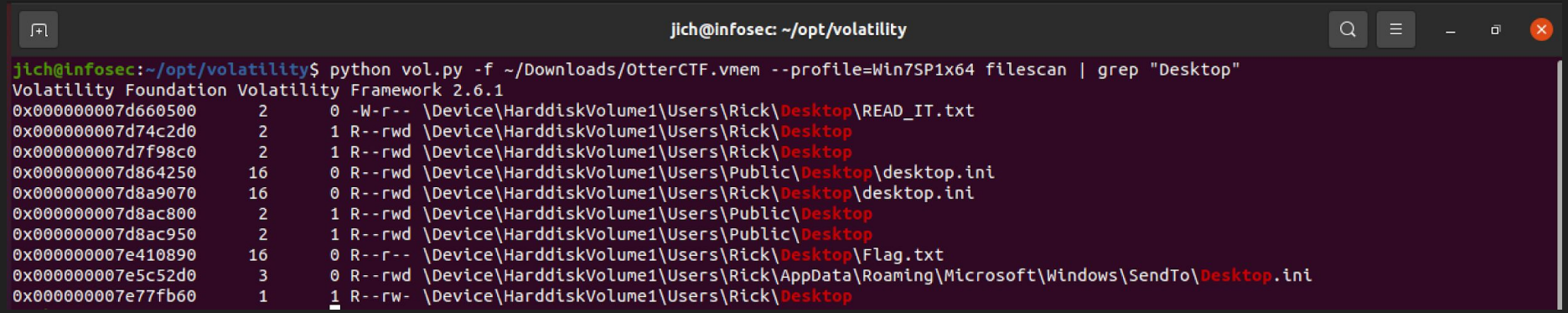

jich@infosec: ~/opt/volatility
jich@infosec:~/opt/volatility$ rm 3896.dmp
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/forensics-challenge-1.mem memdump --profile=Win7SP1x64 -D . -p 3896
Volatility Foundation Volatility Framework 2.6.1
****
Writing notepad.exe [3896] to 3896.dmp
jich@infosec:~/opt/volatility$

File me in - I

- Volatility also supports for the scanning of files on the image
- This is done through the filescan plugin
- Since Volatility searches through all files on the image, this command can be combined with the grep command to filter out files

```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] filescan
```

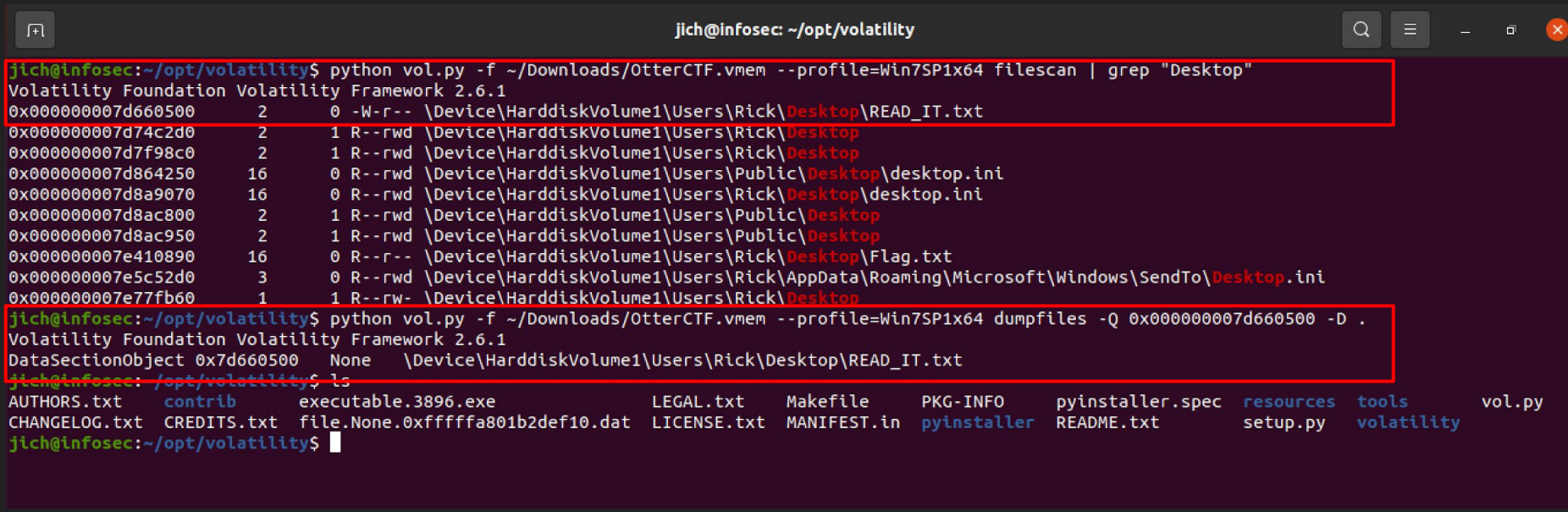
```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] filescan | grep “Desktop”
```


jich@infosec: ~/opt/volatility
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64 filescan | grep "Desktop"
Volatility Foundation Volatility Framework 2.6.1
0x000000007d660500 2 0 -W-r-- \Device\HarddiskVolume1\Users\Rick\Desktop\READ_IT.txt
0x000000007d74c2d0 2 1 R--rwd \Device\HarddiskVolume1\Users\Rick\Desktop
0x000000007d7f98c0 2 1 R--rwd \Device\HarddiskVolume1\Users\Rick\Desktop
0x000000007d864250 16 0 R--rwd \Device\HarddiskVolume1\Users\Public\Desktop\desktop.ini
0x000000007d8a9070 16 0 R--rwd \Device\HarddiskVolume1\Users\Rick\Desktop\desktop.ini
0x000000007d8ac800 2 1 R--rwd \Device\HarddiskVolume1\Users\Public\Desktop
0x000000007d8ac950 2 1 R--rwd \Device\HarddiskVolume1\Users\Public\Desktop
0x000000007e410890 16 0 R--r-- \Device\HarddiskVolume1\Users\Rick\Desktop\Flag.txt
0x000000007e5c52d0 3 0 R--rwd \Device\HarddiskVolume1\Users\Rick\AppData\Roaming\Microsoft\Windows\SendTo\Desktop.ini
0x000000007e77fb60 1 1 R--rw- \Device\HarddiskVolume1\Users\Rick\Desktop

File me in - II

- Afterwards, using the dumpfiles plugin and -Q flag to specify offset, we can extract the contents of the file

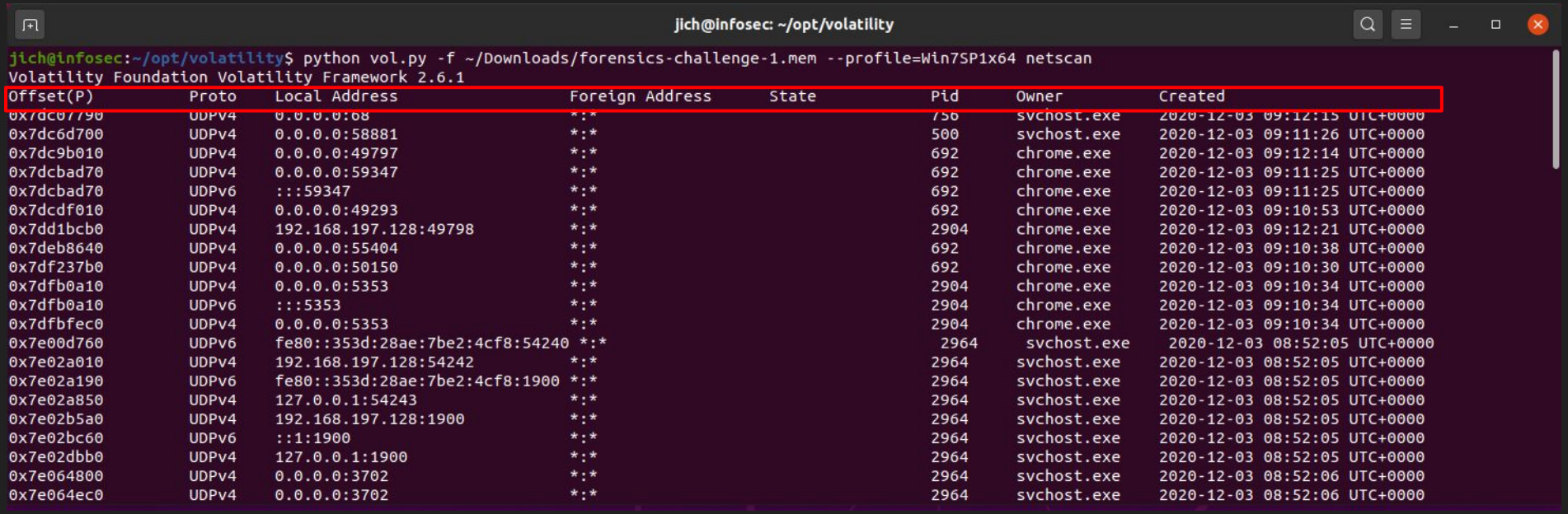
```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] dumpfiles -Q [Offset] -D [Output_Directory]
```


jich@infosec: ~/opt/volatility
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64 filescan | grep "Desktop"
Volatility Foundation Volatility Framework 2.6.1
0x000000007d660500 2 0 -W-r-- \Device\HarddiskVolume1\Users\Rick\Desktop\READ_IT.txt
0x000000007d74c2d0 2 1 R--rwd \Device\HarddiskVolume1\Users\Rick\Desktop
0x000000007d7f98c0 2 1 R--rwd \Device\HarddiskVolume1\Users\Rick\Desktop
0x000000007d864250 16 0 R--rwd \Device\HarddiskVolume1\Users\Public\Desktop\desktop.ini
0x000000007d8a9070 16 0 R--rwd \Device\HarddiskVolume1\Users\Rick\Desktop\desktop.ini
0x000000007d8ac800 2 1 R--rwd \Device\HarddiskVolume1\Users\Public\Desktop
0x000000007d8ac950 2 1 R--rwd \Device\HarddiskVolume1\Users\Public\Desktop
0x000000007e410890 16 0 R--r-- \Device\HarddiskVolume1\Users\Rick\Desktop\Flag.txt
0x000000007e5c52d0 3 0 R--rwd \Device\HarddiskVolume1\Users\Rick\AppData\Roaming\Microsoft\Windows\SendTo\Desktop.ini
0x000000007e77fb60 1 1 R--rw- \Device\HarddiskVolume1\Users\Rick\Desktop
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64 dumpfiles -Q 0x000000007d660500 -D .
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x7d660500 None \Device\HarddiskVolume1\Users\Rick\Desktop\READ_IT.txt
jich@infosec:~/opt/volatility$ ls
AUTHORS.txt contrib executable.3896.exe LEGAL.txt Makefile PKG-INFO pyinstaller.spec resources tools vol.py
CHANGELOG.txt CREDITS.txt file.None.0xfffffa801b2def10.dat LICENSE.txt MANIFEST.in pyinstaller README.txt setup.py volatility
jich@infosec:~/opt/volatility$

Caught in the Network

- We can also find out the network artefacts through the netscan plugin
- There are also some other plugins for network retrieving, however, most of the other plugins only work for x86 and x64 Windows XP and Windows 2003 Server

```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] netscan
```


jich@infosec: ~/opt/volatility
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/forensics-challenge-1.mem --profile=Win7SP1x64 netscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P) Proto Local Address Foreign Address State Pid Owner Created
0x7dc6d700 UDPv4 0.0.0.0:58881 *:* 500 svchost.exe 2020-12-03 09:11:26 UTC+0000
0x7dc9b010 UDPv4 0.0.0.0:49797 *:* 692 chrome.exe 2020-12-03 09:12:14 UTC+0000
0x7dcbad70 UDPv4 0.0.0.0:59347 *:* 692 chrome.exe 2020-12-03 09:11:25 UTC+0000
0x7dcbad70 UDPv6 :::59347 *:* 692 chrome.exe 2020-12-03 09:11:25 UTC+0000
0x7dcdf010 UDPv4 0.0.0.0:49293 *:* 692 chrome.exe 2020-12-03 09:10:53 UTC+0000
0x7dd1bcb0 UDPv4 192.168.197.128:49798 *:* 2904 chrome.exe 2020-12-03 09:12:21 UTC+0000
0x7deb8640 UDPv4 0.0.0.0:55404 *:* 692 chrome.exe 2020-12-03 09:10:38 UTC+0000
0x7df237b0 UDPv4 0.0.0.0:50150 *:* 692 chrome.exe 2020-12-03 09:10:30 UTC+0000
0x7dfb0a10 UDPv4 0.0.0.0:5353 *:* 2904 chrome.exe 2020-12-03 09:10:34 UTC+0000
0x7dfb0a10 UDPv6 :::5353 *:* 2904 chrome.exe 2020-12-03 09:10:34 UTC+0000
0x7dfbfec0 UDPv4 0.0.0.0:5353 *:* 2904 chrome.exe 2020-12-03 09:10:34 UTC+0000
0x7e00d760 UDPv6 fe80::353d:28ae:7be2:4cf8:54240 *:* 2964 svchost.exe 2020-12-03 08:52:05 UTC+0000
0x7e02a010 UDPv4 192.168.197.128:54242 *:* 2964 svchost.exe 2020-12-03 08:52:05 UTC+0000
0x7e02a190 UDPv6 fe80::353d:28ae:7be2:4cf8:1900 *:* 2964 svchost.exe 2020-12-03 08:52:05 UTC+0000
0x7e02a850 UDPv4 127.0.0.1:54243 *:* 2964 svchost.exe 2020-12-03 08:52:05 UTC+0000
0x7e02b5a0 UDPv4 192.168.197.128:1900 *:* 2964 svchost.exe 2020-12-03 08:52:05 UTC+0000
0x7e02bc60 UDPv6 ::1:1900 *:* 2964 svchost.exe 2020-12-03 08:52:05 UTC+0000
0x7e02dbb0 UDPv4 127.0.0.1:1900 *:* 2964 svchost.exe 2020-12-03 08:52:05 UTC+0000
0x7e064800 UDPv4 0.0.0.0:3702 *:* 2964 svchost.exe 2020-12-03 08:52:06 UTC+0000
0x7e064ec0 UDPv4 0.0.0.0:3702 *:* 2964 svchost.exe 2020-12-03 08:52:06 UTC+0000

Hash me outside - I

- The next sections that we will be focusing on will be sections on the Windows Registry and hashes. So we will have a few definitions before we move into it
- Windows Registry: Collection of databases of configuration settings for Microsoft Windows Operating Systems
- Registry Hives: Section of the registry that contains “registry keys”, “registry subkeys” and “registry values”
- SAM: Security Account Manager - Database file in Windows that stores password
- NTLM Hashes: Suite of Microsoft security protocols

Hash me outside - II

- The plugin to retrieve the hashes of the memory dump is hashdump
- After retrieving the hashes, they can be cracked by other tools such as John the Ripper, Hashcat or sometimes a simple search of the hash online

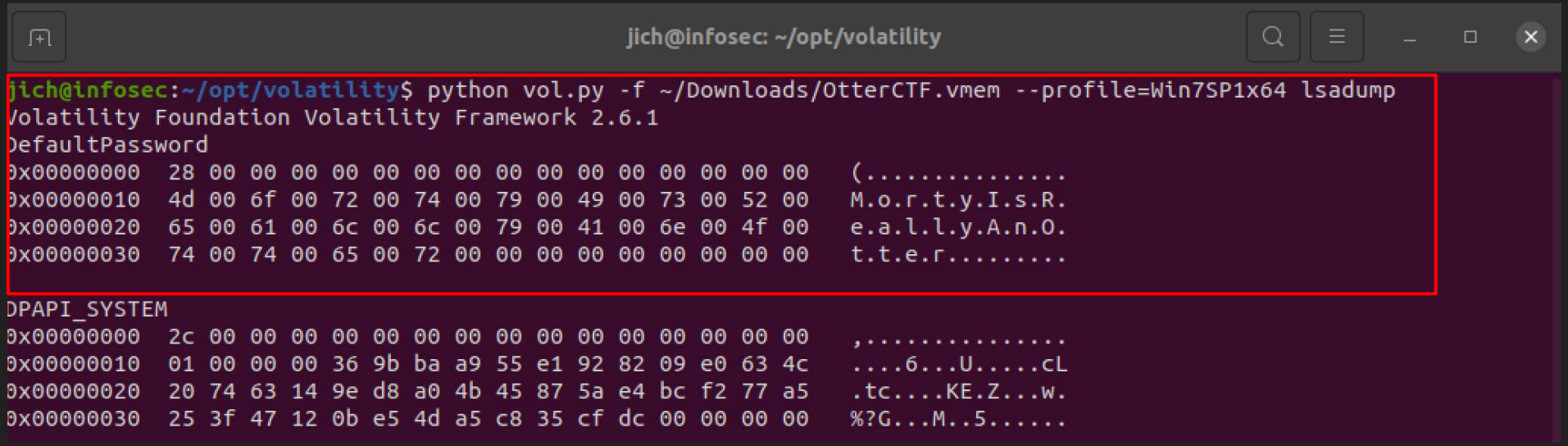
```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] hashdump
```

```
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64 hashdump
Volatility Foundation Volatility Framework 2.6.1
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Rick:1000:aad3b435b51404eeaad3b435b51404ee:518172d012f97d3a8fcc089615283940:::
```

Hash me outside - III

- The passwords of the hashes can also be retrieved in another way, by dumping the LSA secrets of the image
- LSA (Local Security Authority) Secrets file store private data regarding user logins and authentication of users
- The plugin to use is lsadump

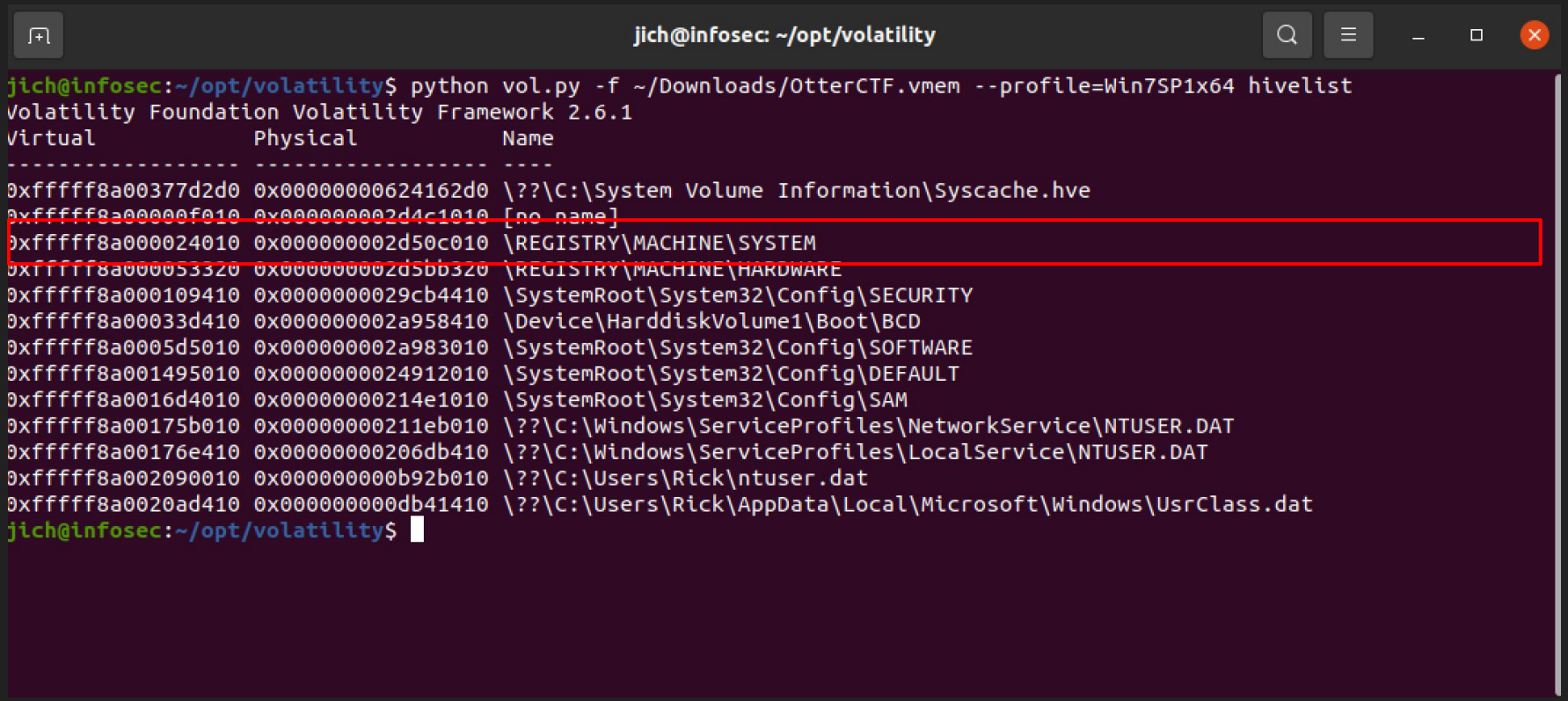
```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] lsadump
```


jich@infosec: ~/opt/volatility
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64 lsadump
Volatility Foundation Volatility Framework 2.6.1
DefaultPassword
0x00000000 28 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   (...............
0x00000010 4d 00 6f 00 72 00 74 00 79 00 49 00 73 00 52 00   M.o.r.t.y.I.s.R.
0x00000020 65 00 61 00 6c 00 6c 00 79 00 41 00 6e 00 4f 00   e.a.l.l.y.A.n.O.
0x00000030 74 00 74 00 65 00 72 00 00 00 00 00 00 00 00 00   t.t.e.r.........
DPAPI_SYSTEM
0x00000000 2c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ,...............
0x00000010 01 00 00 00 36 9b ba a9 55 e1 92 82 09 e0 63 4c   ....6...U.....cL
0x00000020 20 74 63 14 9e d8 a0 4b 45 87 5a e4 bc f2 77 a5   .tc....KE.Z...w.
0x00000030 25 3f 47 12 0b e5 4d a5 c8 35 cf dc 00 00 00 00   %?G...M..5......

Hive Five - I

- Registry hives can contain a wealth of information which includes SYSTEM Registry information such as hostname
- First step into getting to the information is to query the hive using the hivelist plugin

```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] lsadump
```

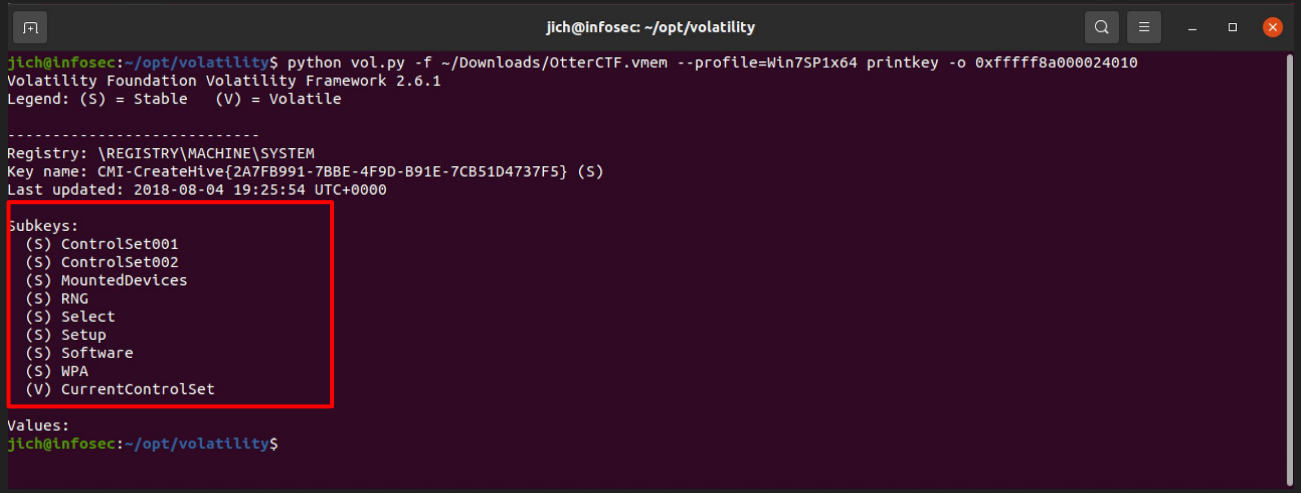

jich@infosec: ~/opt/volatility
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64 hivelist
Volatility Foundation Volatility Framework 2.6.1
Virtual Physical Name
0xfffff8a00377d2d0 0x00000000624162d0 \??\C:\System Volume Information\Syscache.hve
0xfffff8a000024010 0x000000002d50c010 \REGISTRY\MACHINE\SYSTEM
0xfffff8a000109410 0x0000000029cb4410 \SystemRoot\System32\Config\SECURITY
0xfffff8a00033d410 0x000000002a958410 \Device\HarddiskVolume1\Boot\BCD
0xfffff8a0005d5010 0x000000002a983010 \SystemRoot\System32\Config\SOFTWARE
0xfffff8a001495010 0x0000000024912010 \SystemRoot\System32\Config\DEFAULT
0xfffff8a0016d4010 0x00000000214e1010 \SystemRoot\System32\Config\SAM
0xfffff8a00175b010 0x00000000211eb010 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0xfffff8a00176e410 0x00000000206db410 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0xfffff8a002090010 0x000000000b92b010 \??\C:\Users\Rick\ntuser.dat
0xfffff8a0020ad410 0x000000000db41410 \??\C:\Users\Rick\AppData\Local\Microsoft\Windows\UsrClass.dat
jich@infosec:~/opt/volatility$

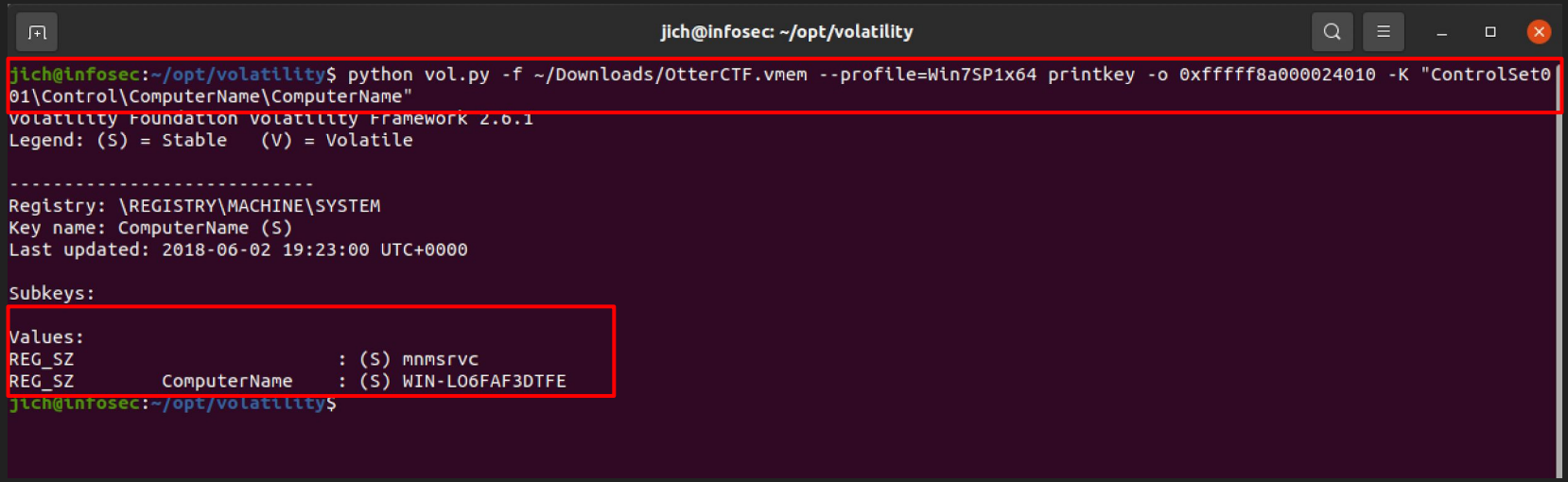
Hive Five - II

- To access each registry, we will have to use the printkey plugin and -o flag to specify the offset of the registry
- Registries might contain a few subkeys, therefore, we can use -K flag to specify which subkeys to access

```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] printkey -o [Virtual_Address]
```

```
> python vol.py -f [Memory_Dump_File] --profile=[Profile] printkey -o [Virtual_Address] -K [Subkey_Path]
```


jich@infosec: ~/opt/volatility
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64 printkey -o 0xfffff8a000024010
Volatility Foundation Volatility Framework 2.6.1
Legend: (S) = Stable (V) = Volatile
Registry: \REGISTRY\MACHINE\SYSTEM
Key name: CMI-CreateHive{2A7FB991-7BBE-4F9D-B91E-7CB51D4737F5} (S)
Last updated: 2018-08-04 19:25:54 UTC+0000
Subkeys:
(S) ControlSet001
(S) ControlSet002
(S) MountedDevices
(S) RNG
(S) Select
(S) Setup
(S) Software
(S) WPA
(V) CurrentControlSet
Values:
jich@infosec:~/opt/volatility$


jich@infosec: ~/opt/volatility
jich@infosec:~/opt/volatility$ python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64 printkey -o 0xfffff8a000024010 -K "ControlSet001\Control\ComputerName\ComputerName"
Volatility Foundation Volatility Framework 2.6.1
Legend: (S) = Stable (V) = Volatile
Registry: \REGISTRY\MACHINE\SYSTEM
Key name: ComputerName (S)
Last updated: 2018-06-02 19:23:00 UTC+0000
Subkeys:
Values:
REG_SZ : (S) mnmsrvc
REG_SZ ComputerName : (S) WIN-LO6FAF3DTFE
jich@infosec:~/opt/volatility$

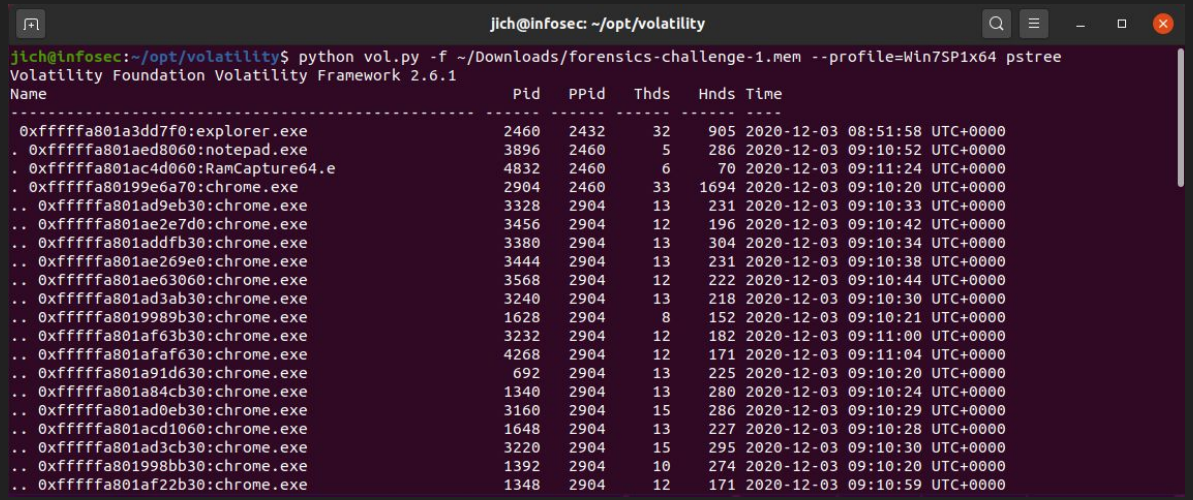
External Help - I

- Volatility also supports plugins written externally to help with forensics
- Plugins can be supplied using the --plugins=[Path_To_plugin] and also specifying the name of the plugin
- For this we will have to first download the plugin from GitHub
- Plugin has to be downloaded in the same directory as the memory dump

```
> git clone https://github.com/superponible/volatility-plugins.git
```

External Help - II

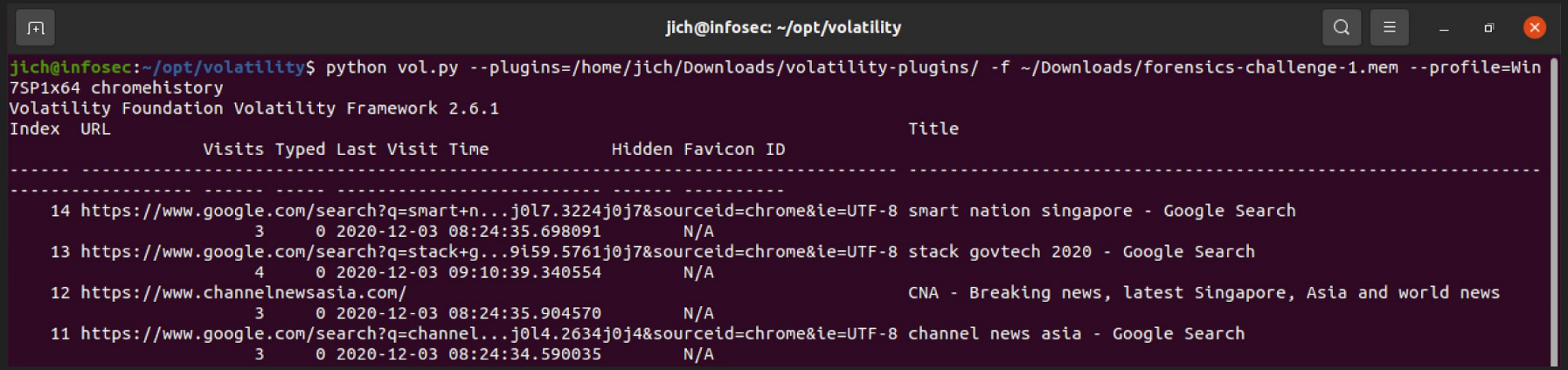
- We will utilise the plugin to dump the Chrome History of a memory dump
- After analysing the process tree, it can be seen that there are multiple instances of “Chrome.exe”



External Help - III

- We will utilise the plugin we downloaded to extract the Chrome history

```
> python vol.py --plugins=[Path_To_Plugin] -f [Memory_Dump_File]
--profile=[Profile] [Plugin_Name]
```



Let’s solve some challenges

OtterCTF 2018 - Chall 1

“We have found out that the memory dump is infected with a ransomware malware. Find out the bitcoin address of this malware”

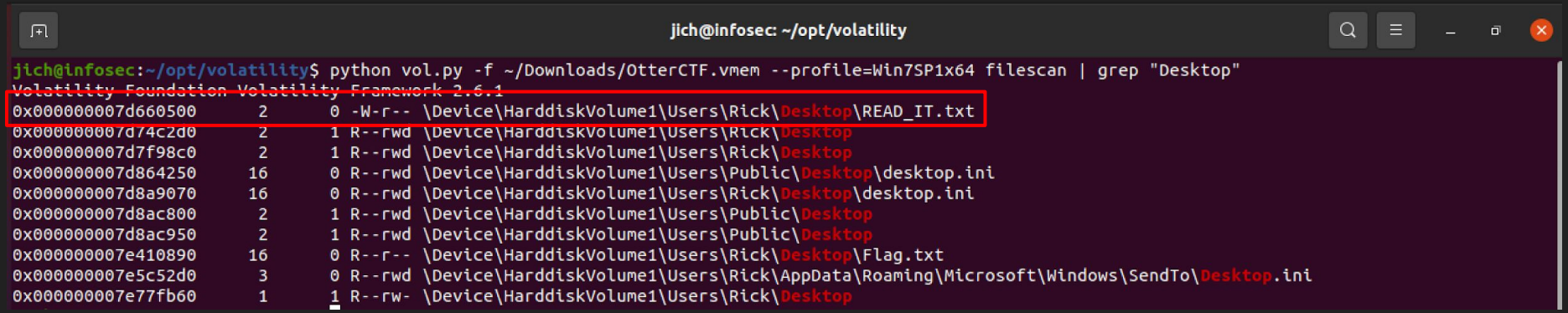
Background Knowledge:

- Ransomware tends to drop messages on the Desktop of the user, therefore, ransom note might be a file on the Desktop
- Using imageinfo, this Memory Dump is identified to have profile of Win7SP1x64

Step 1: File Scan

- Scan the memory dump for the files, filtering out the Desktop Files

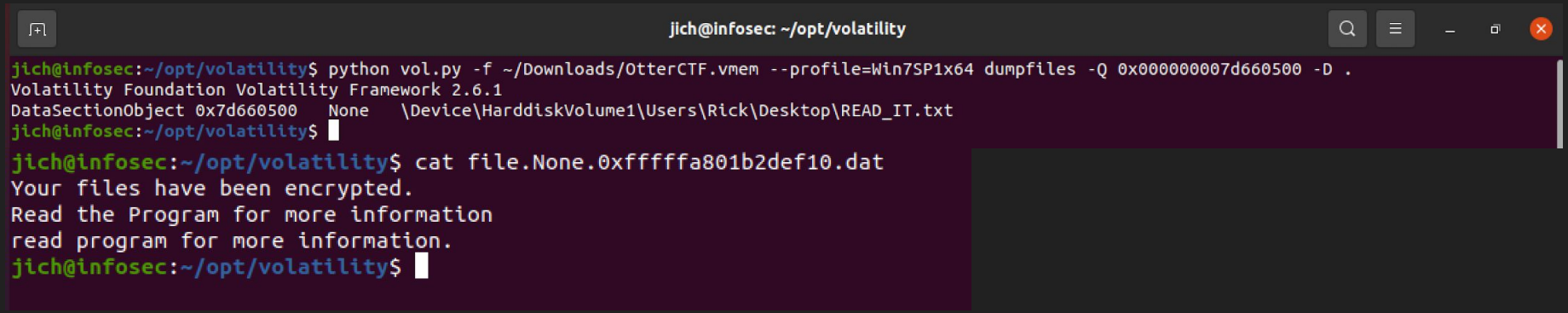
```
> python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64
filescan | grep “Desktop”
```



Step 2: Dump the files

- Dump the READ_IT.txt file using dumpfiles and -Q flag

```
> python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64
dumpfiles -Q
```



Step 3: Memory Dump

- Since the above file did not give us bitcoin address, we have to dump out the process that is suspected to be the malware. During the CTF, in previous questions, the malware is suspected to be named **vmware-tray.exe** with process ID **3720**
- memdump plugin is used to dump the memory of the process

```
> python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64 memdump -p 3720 -D .
```

Step 4: Strings utility

- After dumping the memory, we can use the following command to get the ransomware address

```
> strings -e l 3720.dmp | grep -i -A 20 “Ransom”
```

```
This is Ransomware. It locks your files until you pay for them. Before you ask, Yes we will
give you your files back once you pay and our server confrim that you pay.
Send 0.16 to the address below.
e al
I paid, Now give me back my files.
1MmpEmebJkqXG8nQv4cjJSmxZQFVmFo63M
he program you want to use to open this file:
Flag.txt.WINDOWS
MSCTFIME UI
Cancel
OleMainThreadWndName
Rick And Morty season 1 download.exe - Add New Torrent
hortcut
&Medium icons
cons
arge icons
View &Help
Expand all gro&ups
o grid
&Auto arrange
le names
```

OtterCTF 2018 - Chall 2

“There is something fishy with the malware’s graphics”

Background Knowledge:

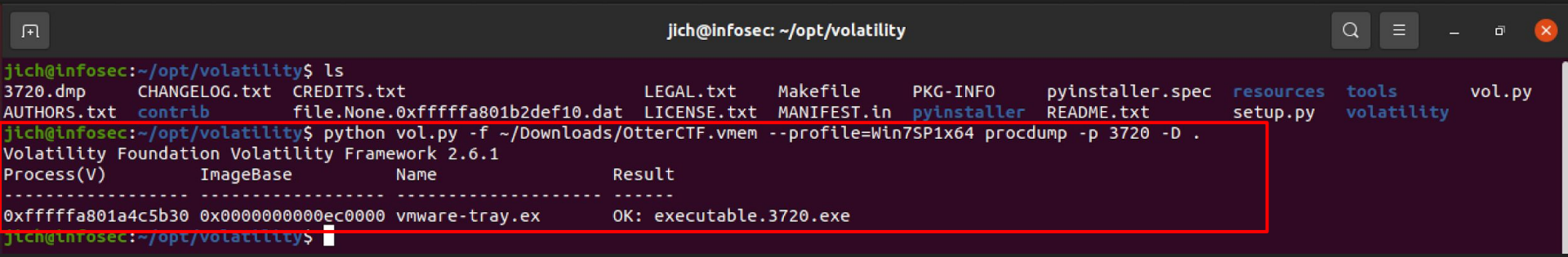
- Knowing that process ID 3720 is the malware process
- Have to check something related to image

Idea:

- Dump out the executable of the process and proceed to carve the executable for any image
- File Carving is another section of forensics, and for now, we will just learn the two main tools for file carving. (Maybe for future L2L…)

Step 1: Dumping Process

```
> python vol.py -f ~/Downloads/OtterCTF.vmem --profile=Win7SP1x64
procdump -p 3720 -D .
```



Step 2: File Carving

Here are some simple commands for file carving:

> binwalk -e [Executable]

> foremost -t [Type] [Executable]

> foremost -t png executable.3720.exe

Future Learning

- This set of slides only provides the basic understanding on how to use Volatility as a tool for extraction of information
- There are still plenty of commands that we have yet to touch. These include more commands to identify the processes in a memory dump image as well as many more commands to extract registry
- In this set of slides, we primarily focus on Windows Images, there are still commands used for Linux as well as Mac OS
- As Volatility 2 is due to have its end of life in August 2021, we will have to learn Volatility 3
- I am also still a beginner at this, so let us learn together!

Slides and Memory Dumps

- Slides in PDF format can be found on GitHub page: https://github.com/jichngan/l2lmemforensics
- Memory Dump is provided in Google Drive Folder
- Forensics-challenge-1.zip is GovTech Stack the Flag 2020 Memory Image
- OtterCTF.zip is OtterCTF 2018 Memory Image
- Thank you!