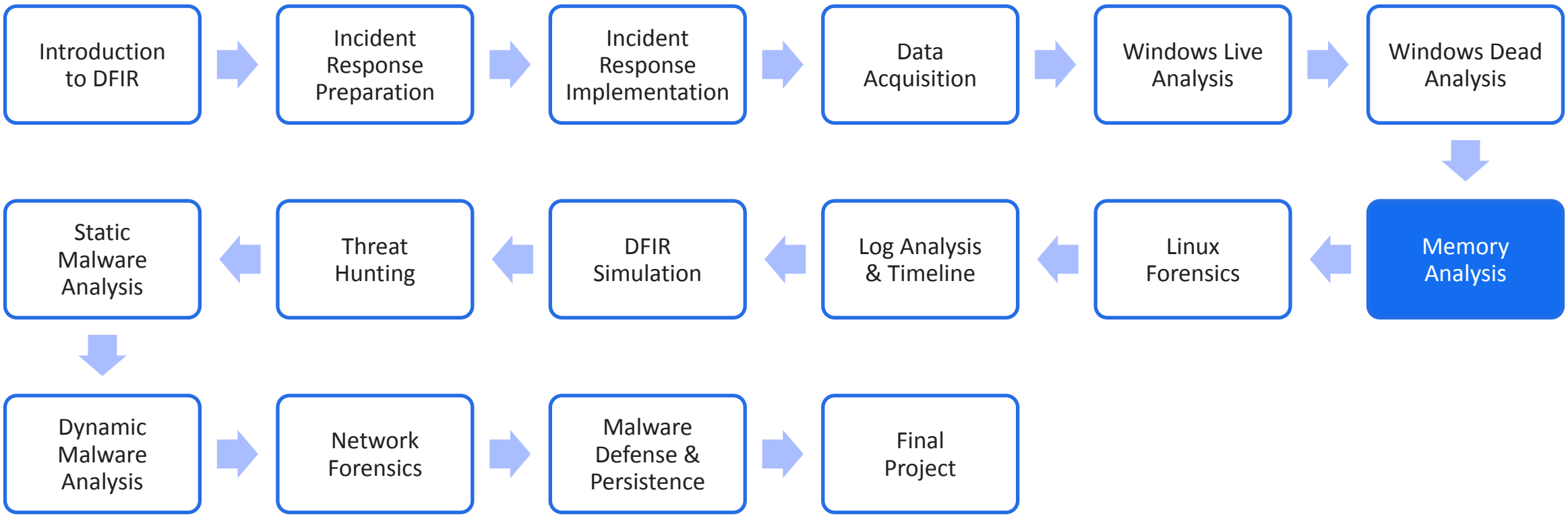
Cybersecurity Professional Program

Memory Analysis

Digital Forensics & Incident Response

Digital Forensics & Incident Response

Course Path



Objectives

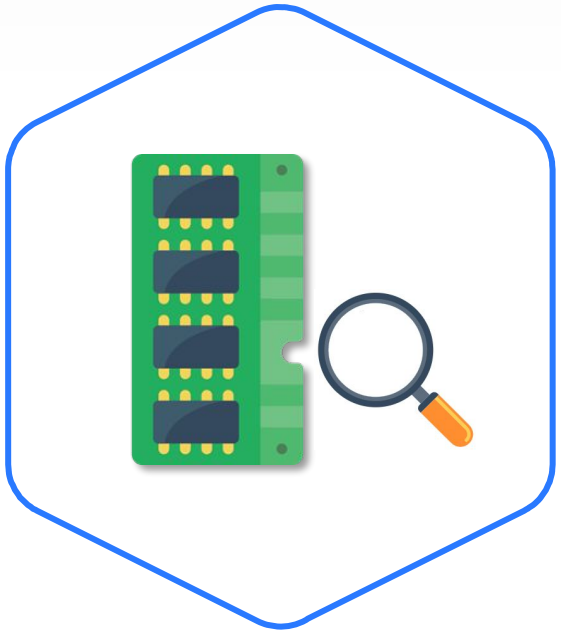
Acquire knowledge and skill of in-depth memory artifact analysis for malware memory investigation.

- Memory Analysis
- Memory Identification
- Process Investigation
- Network Investigation
- Code Injection Investigation
- File & Process Dumping
- System Investigation

Memory Analysis

Memory Analysis

Memory Analysis



- One of the most important sources of information.
- An entire area of expertise.
- Extremely useful in malware analysis.

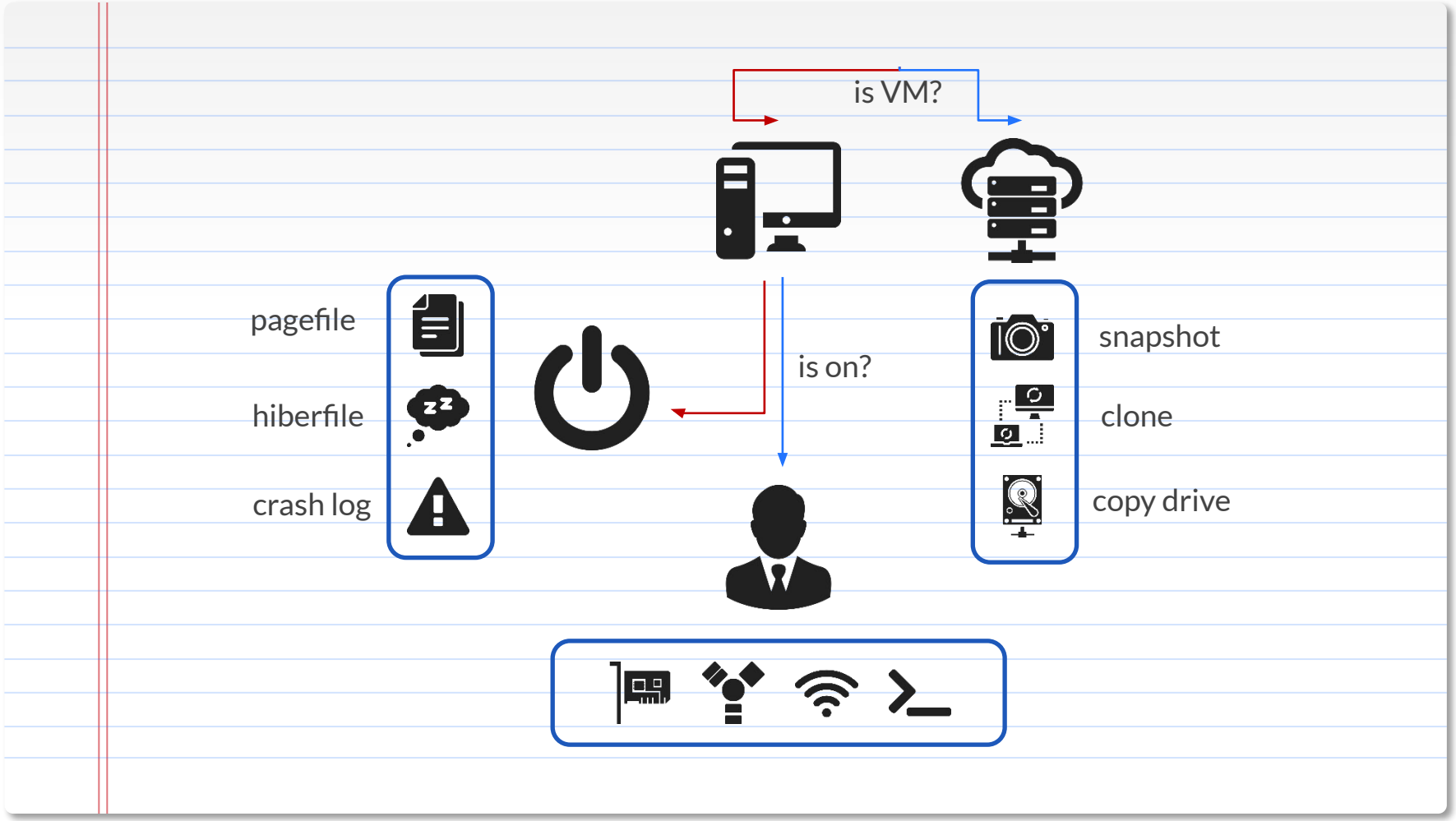
Memory Analysis Use Cases

Fileless Malware
Some malware can run in memory without creating files in the system.

Encrypted Systems
When loaded into memory, most information will be unencrypted.

Decision Tree - Recap

- Memory extraction is directly affected by the system's state.
- Memory dumping is different in bare-metal and virtual systems.
- For bare-metal, the acquisition depends on whether the system is on or off.



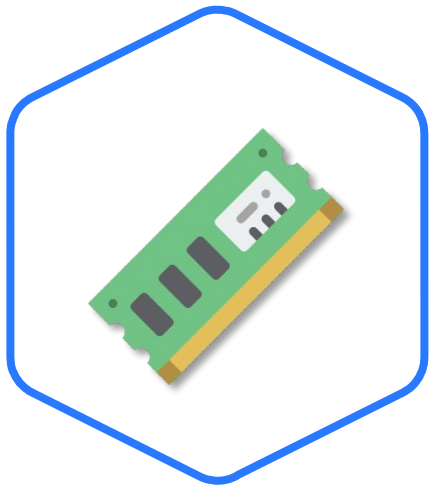
Dump Formats - Recap

Crash Dump

EWF

RAW

Hibernation



AFF4

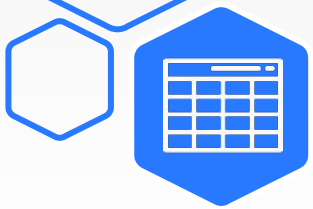
Analysis Frameworks

Volatility
A widely used framework for memory analysis and investigation.

Rekall
A framework developed by Google and an alternative to Volatility.

> Both frameworks feature similar functionality and commands.

Six Investigation Steps

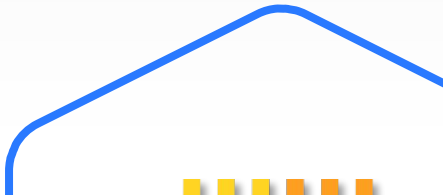


Step	Method
1 – Processes	Investigate rogue processes.
2 – DLL and Handles	Check DLLs used by various executables.
3 – Network	Check network activity and artifacts.
4 – Code Injection	Check for malware traces in memory.
5 – Rootkits	Check for signs of rootkits.
6 – Dump	Dump suspicious processes for in-depth analysis.

Memory Analysis

Memory Identification

KDBG for System Profiling



- Prior to analysis, the memory structure needs to be identified.
- Volatility uses the **_KDDEBUGGER_DATA64** structure to detect the Windows version.

Rekall uses **global symbol information**.

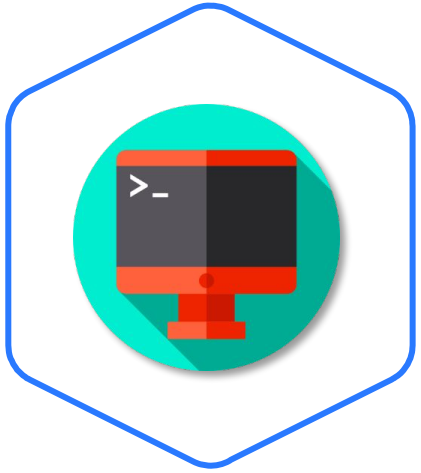
Profile Identification

- Detecting the correct profile is crucial for memory analysis.
- The locations of artifacts are different among OS's.
- Volatility uses the **imageinfo** command.

```
sansforensics@sift -> /tmp
$ vol.py -f cridex.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with
WinXPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/tmp/cridex.vmem)
                      PAE type : PAE
                           DTB : 0x2fe000L
                          KDBG : 0x80545ae0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2012-07-22 02:45:08 UTC+0000
     Image local date and time : 2012-07-21 22:45:08 -0400
sansforensics@sift -> /tmp
$
```

Memory Interaction

Plugins
The most common method is to analyze memory using Volatility.



Volshell
An advanced debug shell that interacts with memory.

Memory Identification

VM Metadata

- Memory obtained from VMs will have additional metadata.
- In a virtual block, this data is saved in a structure called **DBGFCOREDESCRIPTOR**.
- VMware metadata can be extracted using the **vmwareinfo** command.

```
sansforensics@sift -> /tmp
$ vol.py -f cridex.vmem --profile=WinXPSP2x86 volshell
Volatility Foundation Volatility Framework 2.6
Current context: System @ 0x823c89c8, pid=4, ppid=0 DTB=0x2fe000
Python 2.7.12 (default, Oct  8 2019, 14:14:10)
Type "copyright", "credits" or "license" for more information.

IPython 2.4.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: dt("DBGFCOREDESCRIPTOR")
 'DBGFCOREDESCRIPTOR' (24 bytes)
0x0    : u32Magic                     ['unsigned int']
0x4    : u32FmtVersion                ['unsigned int']
0x8    : cbSelf                       ['unsigned int']
0xc    : u32VBoxVersion               ['unsigned int']
0x10   : u32VBoxRevision              ['unsigned int']
0x14   : cCpus                        ['unsigned int']

In [2]:
```

Image Conversion

Crash dumps and hibernation files cannot be read as is.

They require conversion via the **imagecopy** command.

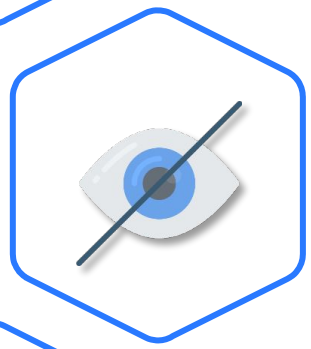
```
sansforensics@sift -> /tmp
$ vol.py imagecopy -f MEMORY.DMP -O crashdump.raw --profile=Win10x64_17134
Volatility Foundation Volatility Framework 2.6.1
Writing data (5.00 MB chunks):
|.....................................................................................
.....................................................................................
.........................|
sansforensics@sift -> /tmp
$ vol.py -f crashdump.raw imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win10x64_17134, Win10x64_14393, Win10x64_10586,
Win10x64_16299, Win2016x64_14393, Win10x64_17763, Win10x64_15063 (Instantiated with
Win10x64_15063)
                     AS Layer1 : SkipDuplicatesAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/tmp/crashdump.raw)
                      PAE type : No PAE
                           DTB : 0x1aa002L
                          KDBG : 0xf8017c599520L
          Number of Processors : 4
     Image Type (Service Pack) : 0
                KPCR for CPU 0 : 0xfffff8017b474000L
```

Memory Analysis

Process Investigation

Search Goals

Parental Structures - Legitimate processes have identifiable parent tree structures.



Hidden Processes - Some processes may attempt to hide their execution.

Suspicious Details - Malicious processes will often attempt to copy the names and PIDs of legitimate processes.

Process Investigation

Pslist and Pstree

Pslist is a basic plugin used to view the process list.

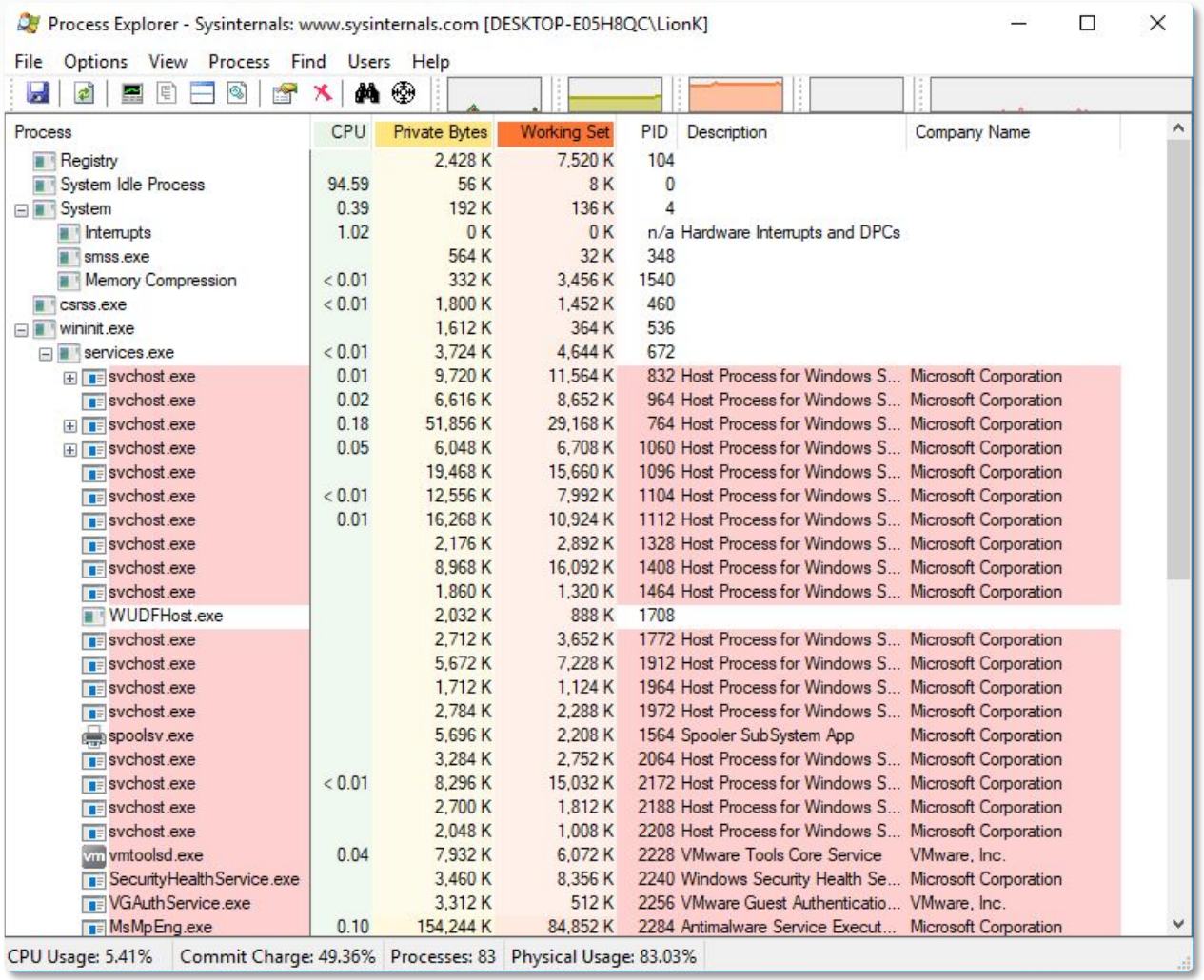
Pstree is more advanced and shows the process hierarchy as well.

Alternatively, processes can be corelated using the PID and PPID values.

```
sansforensics@siftworkstation -> /tmp
$ vol.py -f cridex.vmem --profile=WinXPSP2x86 pstree
Volatility Foundation Volatility Framework 2.6.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x823c89c8:System                                      4      0     53    240 1970-01-01 00:00:00
UTC+0000
. 0x822f1020:smss.exe                                 368      4      3     19 2012-07-22 02:42:31
UTC+0000
.. 0x82298700:winlogon.exe                            608    368     23    519 2012-07-22 02:42:32
UTC+0000
... 0x81e2ab28:services.exe                           652    608     16    243 2012-07-22 02:42:32
UTC+0000
.... 0x821dfda0:svchost.exe                          1056    652      5     60 2012-07-22 02:42:33
UTC+0000
.... 0x81eb17b8:spoolsv.exe                          1512    652     14    113 2012-07-22 02:42:36
UTC+0000
.... 0x81e29ab8:svchost.exe                           908    652      9    226 2012-07-22 02:42:33
UTC+0000
..... 0x8205bda0:wuauclt.exe                         1588   1004      5    132 2012-07-22 02:44:01
UTC+0000
..... 0x821fcda0:wuauclt.exe                         1136   1004      8    173 2012-07-22 02:43:46
UTC+0000
```

Psxview can also be used to tell if a process is trying to hide.

Legitimate Tree

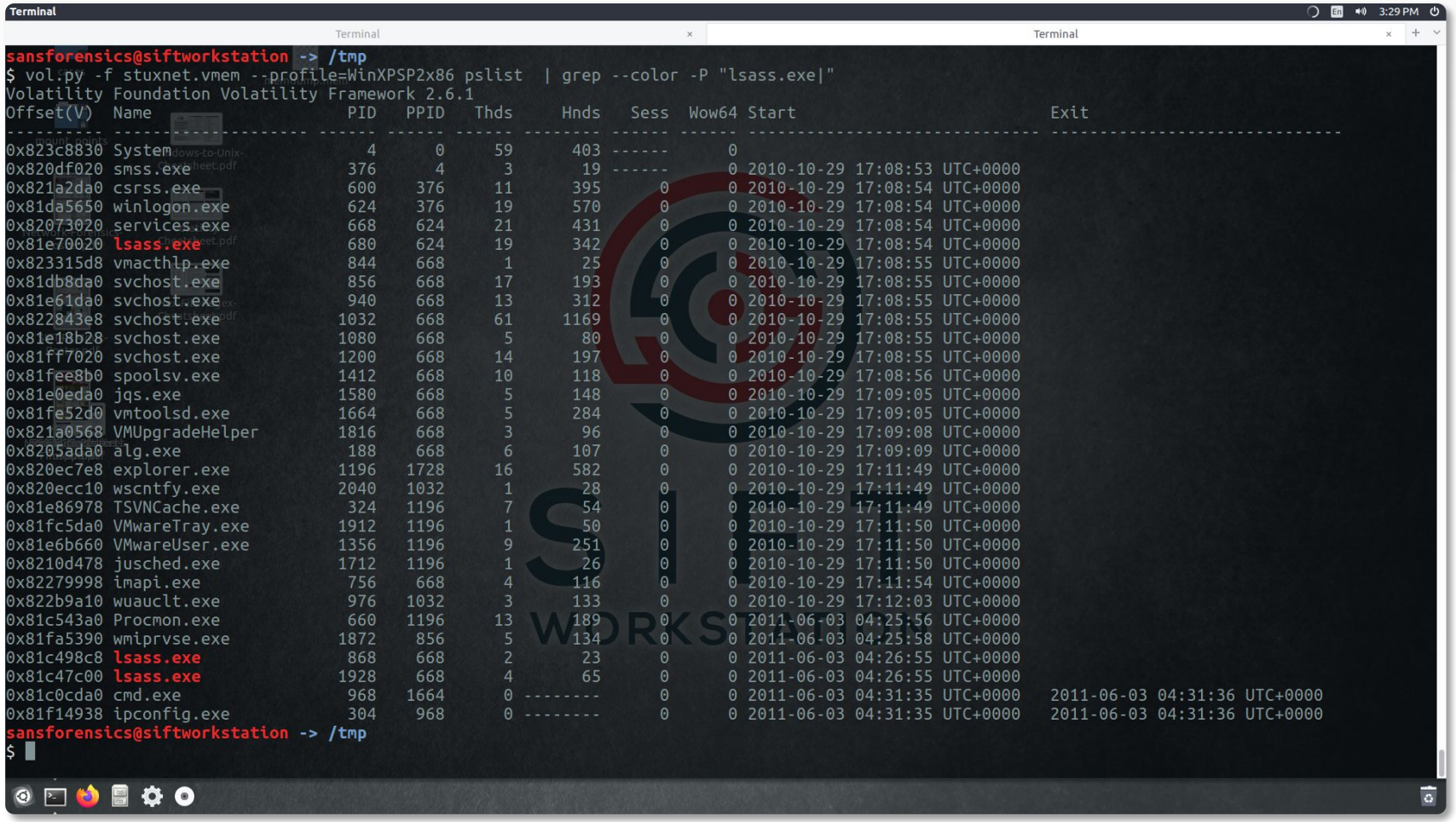


Windows maintains an organized process hierarchy.

Out-of-place processes are easily identifiable.

For example, svchost.exe must be executed under services.exe.

Suspicious Process Tree



In the case of Stuxnet, the malware created processes with illegitimate PPIDs.

Memory Analysis

Network Investigation

Network Connections

Connscan
Scans for identifiable TCP connections in older versions of Windows.

Sockets
Scans for all open sockets.

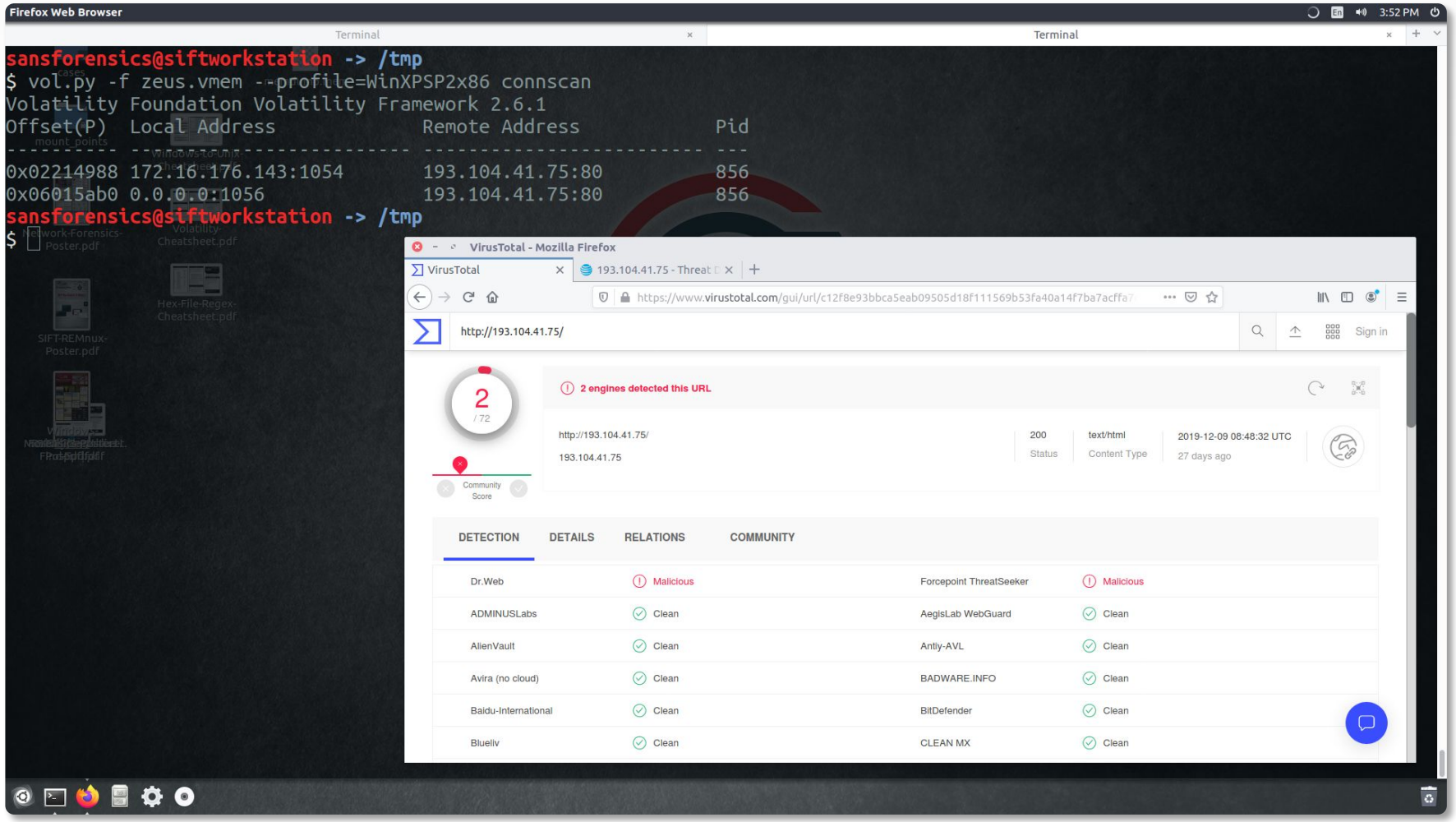
> **Netscan** can be used in more recent versions of Windows.

Network Investigation

Connections Lookup

If a suspicious connection is found, its IP can be looked up.

The IP can also be correlated to a PID.



Network Investigation

Sockets

- Sockets displays only local open ports.
- It shows both TCP and UDP ports.
- Not all ports detected by Conscan will be displayed by Sockets.

```
sansforensics@siftworkstation -> /tmp
$ vol.py -f cridex.vmem --profile=WinXPSP2x86 sockets
Volatility Foundation Volatility Framework 2.6.1
Offset(V)       PID   Port  Proto Protocol        Address         Create Time
---------- -------- ------ ------ -------------- --------------- -----------
0x81ddb780      664    500     17 UDP            0.0.0.0         2012-07-22 02:42:53 UTC+0000
0x82240d08     1484   1038      6 TCP            0.0.0.0         2012-07-22 02:44:45 UTC+0000
0x81dd7618     1220   1900     17 UDP            172.16.112.128  2012-07-22 02:43:01 UTC+0000
0x82125610      788   1028      6 TCP            127.0.0.1       2012-07-22 02:43:01 UTC+0000
0x8219cc08        4    445      6 TCP            0.0.0.0         2012-07-22 02:42:31 UTC+0000
0x81ec23b0      908    135      6 TCP            0.0.0.0         2012-07-22 02:42:33 UTC+0000
0x82276878        4    139      6 TCP            172.16.112.128  2012-07-22 02:42:38 UTC+0000
0x82277460        4    137     17 UDP            172.16.112.128  2012-07-22 02:42:38 UTC+0000
0x81e76620     1004    123     17 UDP            127.0.0.1       2012-07-22 02:43:01 UTC+0000
0x82172808      664      0    255 Reserved       0.0.0.0         2012-07-22 02:42:53 UTC+0000
0x81e3f460        4    138     17 UDP            172.16.112.128  2012-07-22 02:42:38 UTC+0000
0x821f0630     1004    123     17 UDP            172.16.112.128  2012-07-22 02:43:01 UTC+0000
0x822cd2b0     1220   1900     17 UDP            127.0.0.1       2012-07-22 02:43:01 UTC+0000
0x82172c50      664   4500     17 UDP            0.0.0.0         2012-07-22 02:42:53 UTC+0000
```

- Conscan and Sockets should be used as complementary plugins.

Network Investigation

URL Identification

- URLs represent network-related information not identified in memory.
- There is no available utility that can be used to scan for URLs.
- URLs can, however, be retrieved via Strings or Bulk Extractor.

```
sansforensics@siftworkstation -> /tmp
$ strings cridex.vmem | egrep "^http://" | sort | uniq
http://
http://18
http://188.40.0.138:8080/zb/v_01_a/in/cp.php
http://*:2869/
http://ns.adobe.com/xap/1.0/
http://ocsp.verisign.com0
http://%s/%s
http://www.microsoft.com/provisioning/BaseEapConnectionPropertiesV1
http://www.microsoft.com/provisioning/BaseEapUserPropertiesV1
http://www.microsoft.com/provisioning/Register
http://www.microsoft.com/provisioning/SSID
http://www.microsoft.com/provisioning/WirelessProfile
http://www.microsoft.com/SoftwareDistribution
http://www.microsoft.com/SoftwareDistribution/Server/ClientWebService
http://www.microsoft.com/SoftwareDistribution/Server/IMonitorable
http://www.usertrust.com1
http://www.usertrust.com1+0)
http://www.usertrust.com1604
http://www.valicert.com/1 0
```

Network Investigation

Lab DFIR-07-L1

WannaCry

15 – 20 Min

Mission

Inspect the memory image and find IoCs that indicate the system was infected with WannaCry.

Steps

- Find IoCs that indicate the system was infected with WannaCry.
- Identify the domain WannaCry attempted to access.

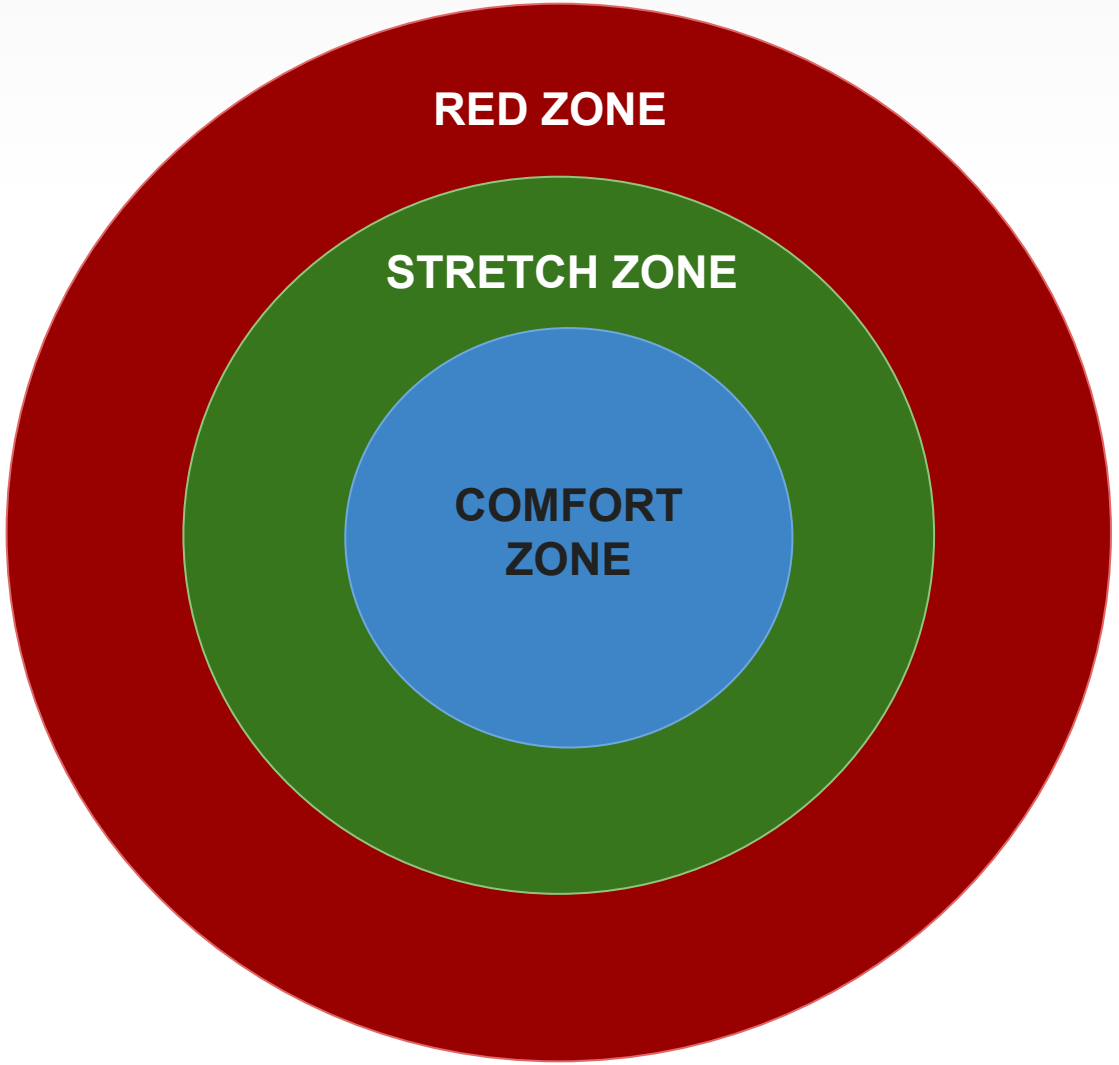
Environment & Tools

- VirtualBox
- SIFT Workstation

Related Files

- Lab document
- DFIR-07-L1 wcry.zip

Pulse Check



Memory Analysis

Code Injection Investigation

DLL List

In Windows, information can be obtained by inspecting DLLs.

DLLs can indicate network activity or access to special system APIs.

DLLs for specific processes can be viewed using **Dlllist –p**.

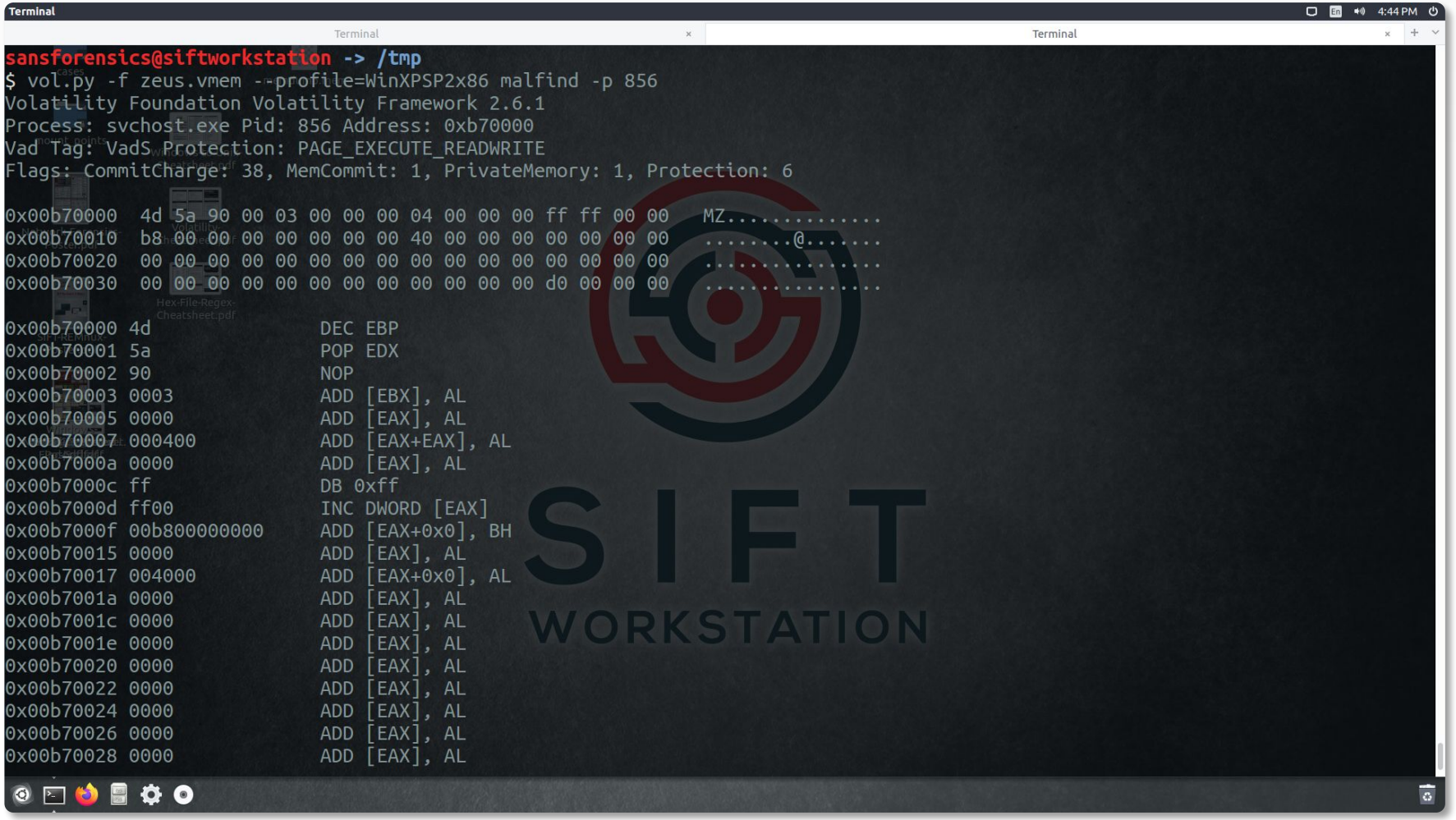
```
sansforensics@siftworkstation -> /tmp
$ vol.py -f zeus.vmem --profile=WinXPSP2x86 dlllist -p 856
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
svchost.exe pid:    856
Command line : C:\WINDOWS\system32\svchost -k DcomLaunch
Service Pack 2

Base             Size  LoadCount LoadTime                       Path
---------- ---------- ---------- ------------------------------ ----
0x01000000     0x6000     0xffff
C:\WINDOWS\system32\svchost.exe
0x7c900000    0xb0000     0xffff                                C:\WINDOWS\system32\ntdll.dll
0x7c800000    0xf4000     0xffff
C:\WINDOWS\system32\kernel32.dll
0x77dd0000    0x9b000     0xffff
C:\WINDOWS\system32\ADVAPI32.dll
0x77e70000    0x91000     0xffff                                C:\WINDOWS\system32\RPCRT4.dll
0x5cb70000    0x26000        0x1
C:\WINDOWS\system32\ShimEng.dll
0x6f880000    0x1ca000        0x1
```

Dlllist displays all DLLs loaded for a process in the system.

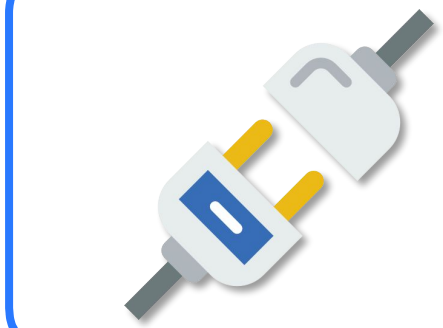
Code Injection Investigation

Malfind

- Volatility includes the Malfind plugin that detects possible code injection.
- Malfind detects executable sections in memory.



Third-Party Plugins



- Volatility supports third-party plugins that can be used in more complex investigations.
- The plugins are helpful when analyzing sophisticated attacks.

> Hollowfind is a plugin used to detect process hollowing attacks.

Memory Analysis

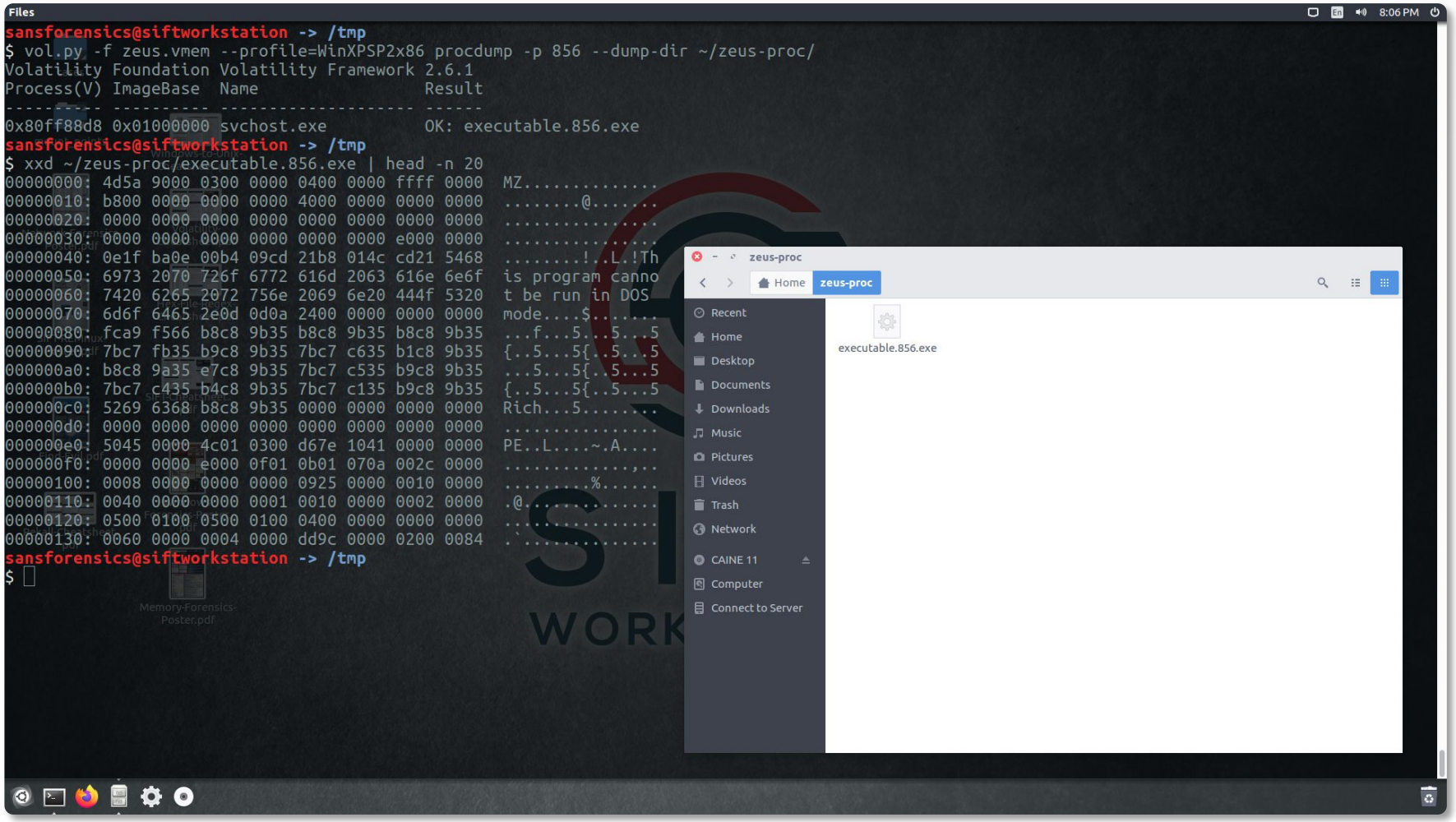
File & Process Dumping

Process Dumping

This is an important technique in any investigation.

Procdump allows dumping a process as an executable.

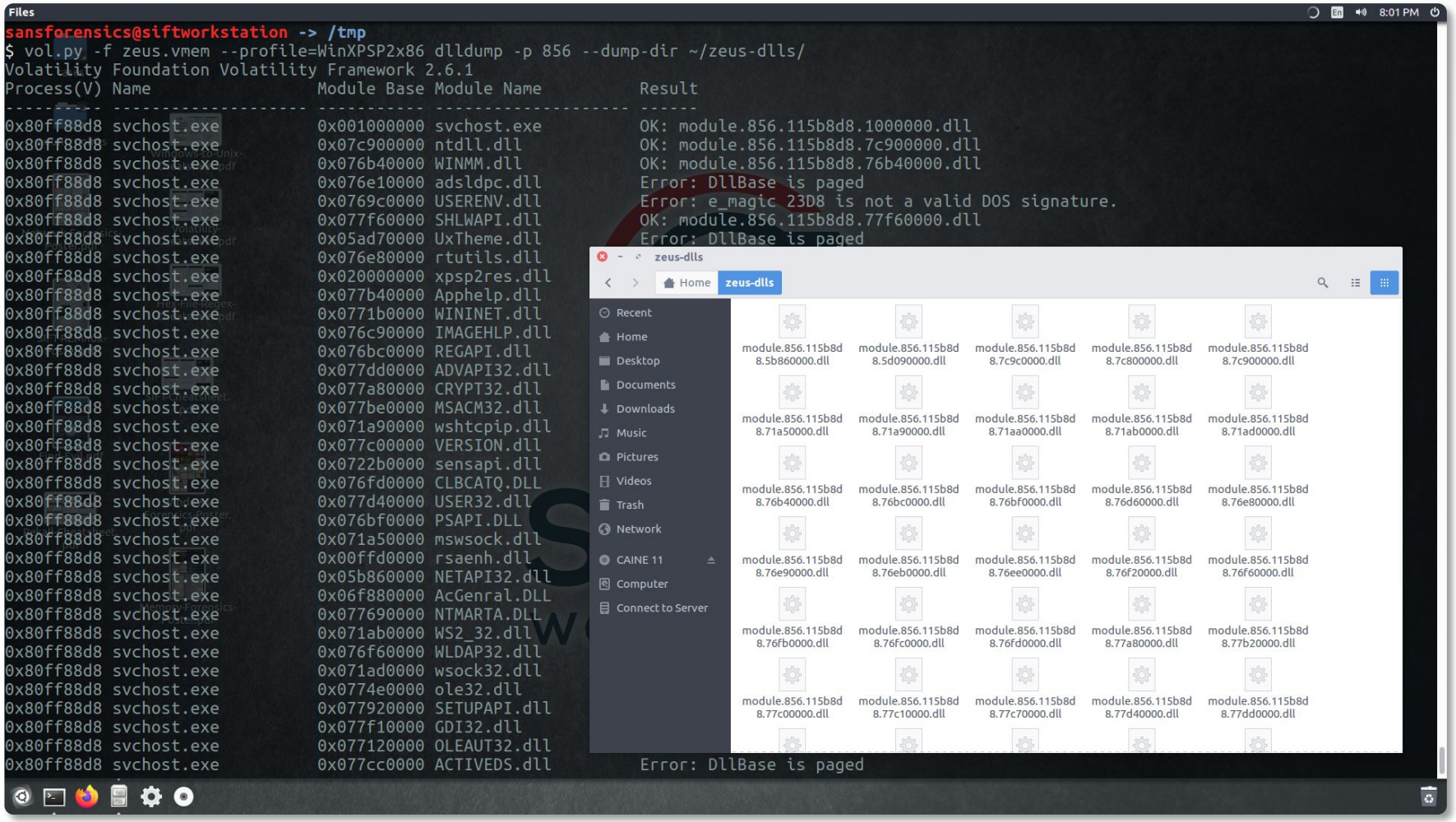
The executable can then be further inspected and analyzed.



File & Process Dumping

DLL Dump

- Some DLLs loaded by processes can be dumped.
- DLLs should be dumped per process.
- A memory capture can contain thousands of DLLs.



File Dumping

Sometimes files are found in memory.

Files in memory can be dumped using **Dumpfiles**.

For Dumpfiles to work, the offset must first be found.

```
sansforensics@siftworkstation -> /tmp
$ $ vol.py -f zeus.vmem --profile=WinXPSP2x86 filescan | head -n 40
Volatility Foundation Volatility Framework 2.6.1

Offset(P)            #Ptr   #Hnd Access Name
------------------ ------ ------ ------ ----
0x0000000000353ad0      1      0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\crypt32.dll
0x0000000000353cb8      1      0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\apphelp.dll
0x00000000003f34f8      3      0 RWD--- \Device\HarddiskVolume1\$Directory
0x00000000003f3f08      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\ipconf.tsp
0x0000000000471170      1      0 R--r-- \Device\HarddiskVolume1\WINDOWS\system32\wzcdlg.dll
0x00000000004715c0      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\cnbjmon.dll
0x00000000004a06a0      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\urlmon.dll
0x00000000004a09c8      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\localspl.dll
0x00000000004aa4a0      3      0 RWD--- \Device\HarddiskVolume1\$Directory
0x00000000004c94a0      1      0 R--rwd \Device\HarddiskVolume1\WINDOWS\WindowsShell.Manifest
0x00000000004c9e48      3      0 RWD--- \Device\HarddiskVolume1\$Directory
0x00000000007c08e8      1      1 ------ \Device\Afd\Endpoint
...
```

Files loaded in memory are typically DLLs.

File & Process Dumping

Lab DFIR-07-L2

Stuxnet

30 – 40 Min

Mission

Investigate the provided sample and locate the malware within it.

Steps

- Identify the suspicious process.
- Identify suspicious network activity.
- Search for code injection.
- Test for virus signatures.

Environment & Tools

- VirtualBox
- SIFT Workstation

Related Files

- Lab document
- DFIR-07-L2 stuxnet.zip

Memory Analysis

System Investigation

Environmental Variables

Environmental variables are always saved for each process, and may indicate where a process was executed from.

```
sansforensics@sift -> /tmp
$ vol.py -f zeus.vmem --profile=WinXPSP2x86 envars -p 856
Volatility Foundation Volatility Framework 2.6.1
Pid      Process        Block              Variable                       Value
--- --------------- ---------- ------------------------------ -----
856 svchost.exe      0x00010000 ALLUSERSPROFILE                C:\Documents and Settings\Users
856 svchost.exe      0x00010000 CommonProgramFiles             C:\Program Files\Common Files
856 svchost.exe      0x00010000 COMPUTERNAME                   BILLY-DB5B96DD3
856 svchost.exe      0x00010000 ComSpec                        C:\WINDOWS\system32\cmd.exe
856 svchost.exe      0x00010000 FP_NO_HOST_CHECK               NO
856 svchost.exe      0x00010000 NUMBER_OF_PROCESSORS           1
856 svchost.exe      0x00010000 OS                             Windows_NT
856 svchost.exe      0x00010000 PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;
856 svchost.exe      0x00010000 PROCESSOR_ARCHITECTURE         x86
856 svchost.exe      0x00010000 PROCESSOR_IDENTIFIER           x86 Family 6 Model 23 Stepping 10,
856 svchost.exe      0x00010000 PROCESSOR_LEVEL                6
856 svchost.exe      0x00010000 PROCESSOR_REVISION             170a
856 svchost.exe      0x00010000 ProgramFiles                   C:\Program Files
856 svchost.exe      0x00010000 SystemDrive                    C:
856 svchost.exe      0x00010000 SystemRoot                     C:\WINDOWS
856 svchost.exe      0x00010000 TEMP                           C:\WINDOWS\TEMP
856 svchost.exe      0x00010000 TMP                            C:\WINDOWS\TEMP
856 svchost.exe      0x00010000 USERPROFILE                    C:\WINDOWS\system32\config
856 svchost.exe      0x00010000 windir                         C:\WINDOWS
```

System Investigation

Registry Keys

- Some registry keys can be extracted from memory.
- One of the most common keys to check is winlogon.

```
sansforensics@sift -> /tmp
$ vol.py -f zeus.vmem --profile=WinXPSP2x86 dumpkey printkey -K "Microsoft\Windows
NT\CurrentVersion\Winlogon"
Volatility Foundation Volatility Framework 2.6.1
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\software
Key name: Winlogon (S)
Last updated: 2010-08-15 19:17:23 UTC+0000

...

Values:
REG_DWORD     AutoRestartShell : (S) 1
REG_SZ        DefaultDomainName : (S) BILLY-DB5B96DD3
REG_SZ        DefaultUserName : (S) Administrator
REG_SZ        LegalNoticeCaption : (S)
REG_SZ        LegalNoticeText : (S)
REG_SZ        PowerdownAfterShutdown : (S) 0
REG_SZ        ReportBootOk    : (S) 1
REG_SZ        Shell           : (S) Explorer.exe
REG_SZ        ShutdownWithoutLogon : (S) 0
REG_SZ        System          : (S)
REG_SZ        Userinit        : (S) C:\WINDOWS\system32\userinit.exe,C:\WINDOWS\system32\sdra64.exe,
REG_SZ        VmApplet        : (S) rundll32 shell32,Control_RunDLL "sysdm.cpl"
REG_DWORD     SfcQuota        : (S) 4294967295
```

Command Line Data

Cmdscan
Scans for a command line history buffer.

Consoles
Scans for available console information.

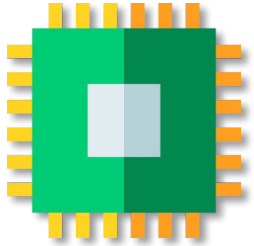
> Consoles is not generally used in Windows.

Additional Plugins

evtlogs

getsids

iehistory



modscan

System Investigation

Lab DFIR-07-L3

Zeus

20 – 30 Min

Mission

Investigate the memory sample and locate the malicious program, without guidelines.

Steps

- Discover malicious activity.
- Conduct a memory investigation, without guidelines.

Environment & Tools

- VirtualBox
- SIFT Workstation

Related Files

- Lab document
- DFIR-07-L3 zeus.zip

Thank You

Questions?