

Cybersecurity Professional Program

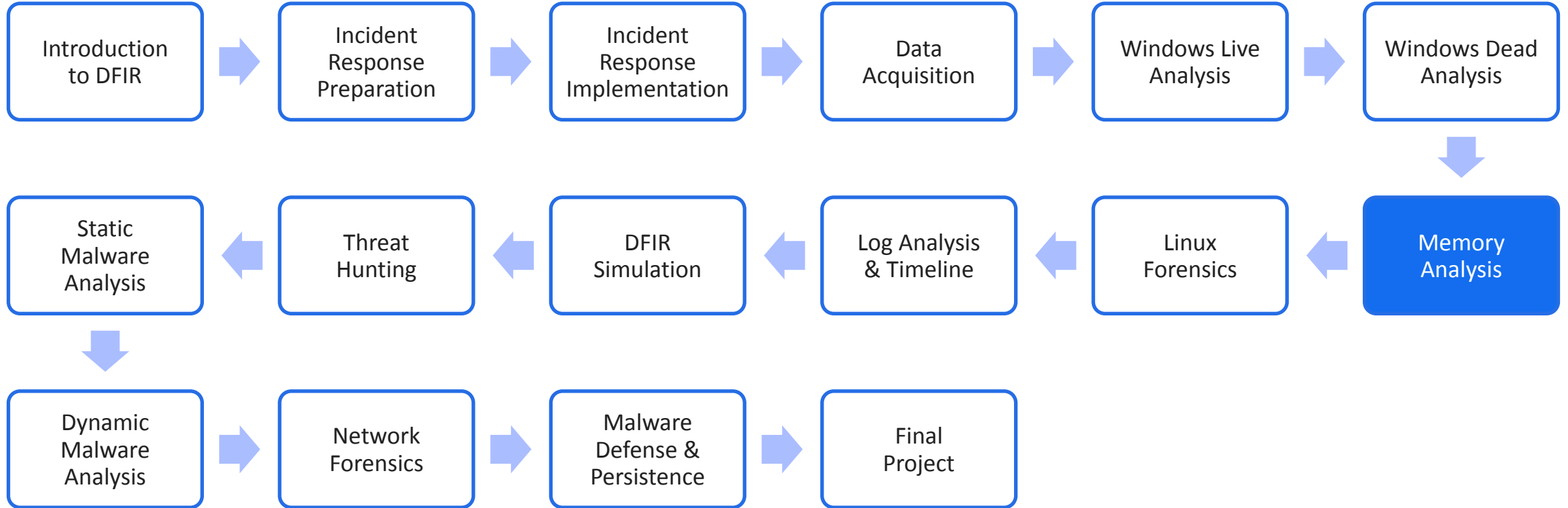

Memory Analysis


Digital Forensics & Incident Response



Digital Forensics & Incident Response

Course Path



The logo for 'Memory Analysis Objectives' features three blue hexagons on the left. The top hexagon is a wireframe, the middle one is solid blue, and the bottom one is solid blue with a white target icon. To the right of the hexagons, the text 'Memory Analysis' is in a small, grey, sans-serif font, and 'Objectives' is in a large, bold, blue, sans-serif font.

Memory Analysis Objectives

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

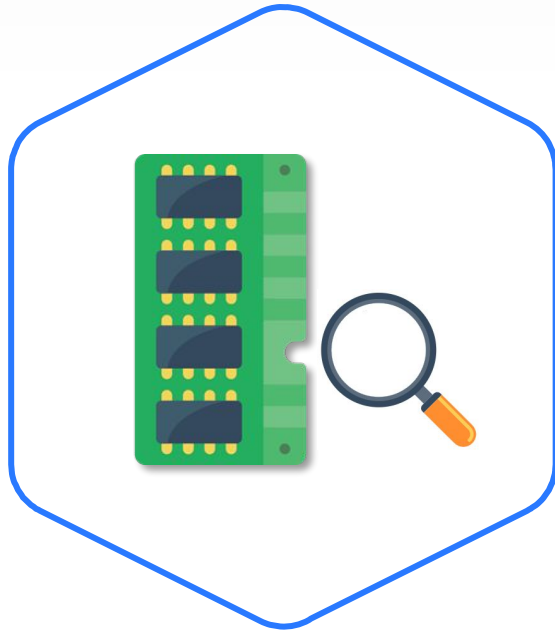
- Memory Analysis
- Memory Identification
- Process Investigation
- Network Investigation
- Code Injection Investigation
- File & Process Dumping
- System Investigation



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.

Memory Analysis

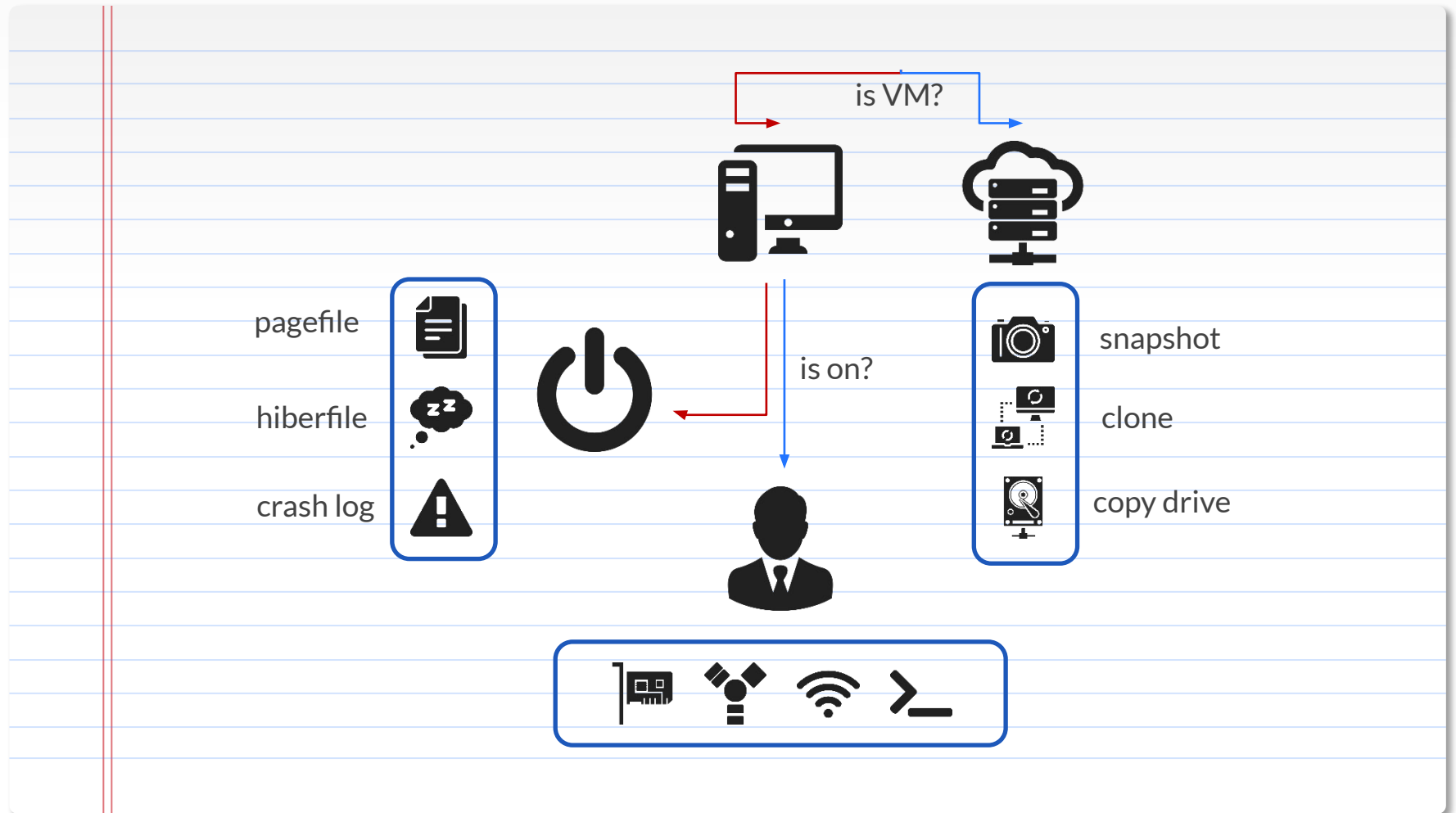
Decision Tree - Recap



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



RAW



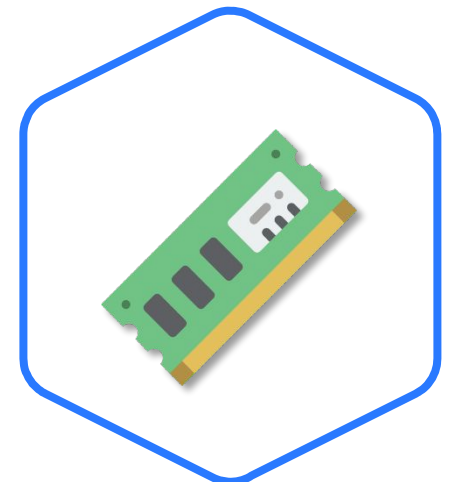
Crash Dump



Hibernation



EWF



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.

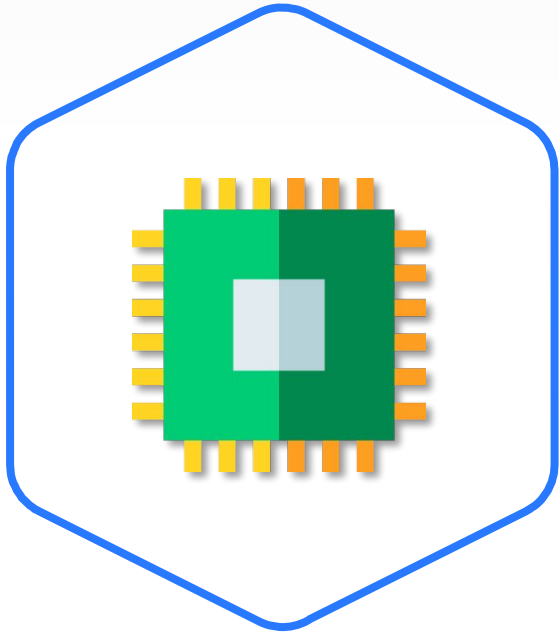




Memory Analysis

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**.



Memory Identification

Profile Identification



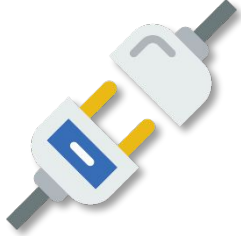
Detecting the correct profile is crucial for memory analysis.

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



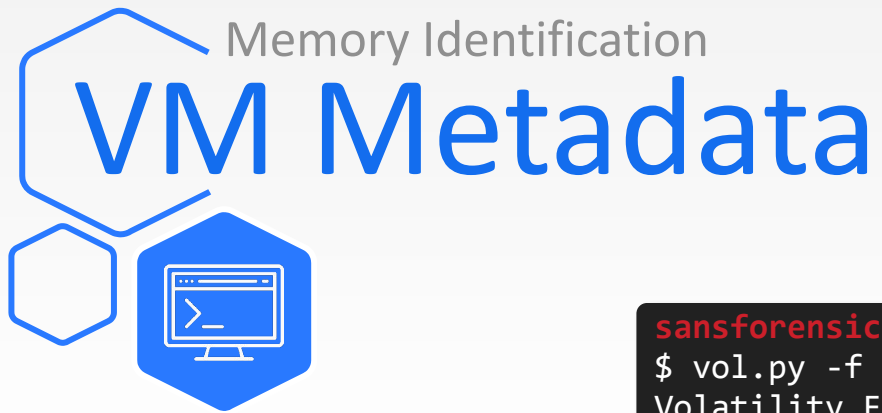
Plugins

The most common method is to analyze memory using Volatility.



Volshell

An advanced debug shell that interacts with memory.



Memory obtained from VMs will have additional metadata.

In a virtual block, this data is saved in a structure called **DBGFCOREDESCRIPT** OR.

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]:
```

Memory Identification

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 : 0xffffffff8017b474000L
```



Memory Analysis

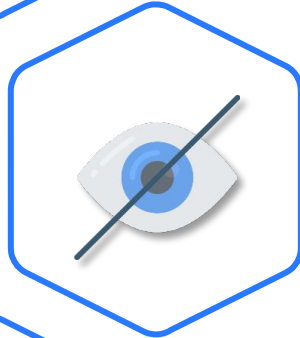
Process Investigation

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.



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 correlated using the PID and PPID values.

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

```
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 UTC+0000	4	0	53	240	1970-01-01 00:00:00
. 0x822f1020:smss.exe UTC+0000	368	4	3	19	2012-07-22 02:42:31
.. 0x82298700:winlogon.exe UTC+0000	608	368	23	519	2012-07-22 02:42:32
... 0x81e2ab28:services.exe UTC+0000	652	608	16	243	2012-07-22 02:42:32
.... 0x821dfda0:svchost.exe UTC+0000	1056	652	5	60	2012-07-22 02:42:33
.... 0x81eb17b8:spoolsv.exe UTC+0000	1512	652	14	113	2012-07-22 02:42:36
.... 0x81e29ab8:svchost.exe UTC+0000	908	652	9	226	2012-07-22 02:42:33
..... 0x8205bda0:wuauc1t.exe UTC+0000	1588	1004	5	132	2012-07-22 02:44:01
..... 0x821fcda0:wuauc1t.exe UTC+0000	1136	1004	8	173	2012-07-22 02:43:46
..... 0x82311360:svchost.exe UTC+0000	824	652	20	184	2012-07-22 02:42:33

Process Investigation

Legitimate Tree

Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-E05H8QC\LionK]

File Options View Process Find Users Help

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Registry		2,428 K	7,520 K	104		
System Idle Process	94.59	56 K	8 K	0		
System	0.39	192 K	136 K	4		
Interrupts	1.02	0 K	0 K	n/a	Hardware Interrupts and DPCs	
smss.exe		564 K	32 K	348		
Memory Compression	< 0.01	332 K	3,456 K	1540		
csrss.exe	< 0.01	1,800 K	1,452 K	460		
wininit.exe		1,612 K	364 K	536		
services.exe	< 0.01	3,724 K	4,644 K	672		
svchost.exe	0.01	9,720 K	11,564 K	832	Host Process for Windows S...	Microsoft Corporation
svchost.exe	0.02	6,616 K	8,652 K	964	Host Process for Windows S...	Microsoft Corporation
svchost.exe	0.18	51,856 K	29,168 K	764	Host Process for Windows S...	Microsoft Corporation
svchost.exe	0.05	6,048 K	6,708 K	1060	Host Process for Windows S...	Microsoft Corporation
svchost.exe		19,468 K	15,660 K	1096	Host Process for Windows S...	Microsoft Corporation
svchost.exe	< 0.01	12,556 K	7,992 K	1104	Host Process for Windows S...	Microsoft Corporation
svchost.exe	0.01	16,268 K	10,924 K	1112	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,176 K	2,892 K	1328	Host Process for Windows S...	Microsoft Corporation
svchost.exe		8,968 K	16,092 K	1408	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,860 K	1,320 K	1464	Host Process for Windows S...	Microsoft Corporation
WUDFHost.exe		2,032 K	888 K	1708		
svchost.exe		2,712 K	3,652 K	1772	Host Process for Windows S...	Microsoft Corporation
svchost.exe		5,672 K	7,228 K	1912	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,712 K	1,124 K	1964	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,784 K	2,288 K	1972	Host Process for Windows S...	Microsoft Corporation
spoolsv.exe		5,696 K	2,208 K	1564	Spooler SubSystem App	Microsoft Corporation
svchost.exe		3,284 K	2,752 K	2064	Host Process for Windows S...	Microsoft Corporation
svchost.exe	< 0.01	8,296 K	15,032 K	2172	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,700 K	1,812 K	2188	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,048 K	1,008 K	2208	Host Process for Windows S...	Microsoft Corporation
vmtoolsd.exe	0.04	7,932 K	6,072 K	2228	VMware Tools Core Service	VMware, Inc.
SecurityHealthService.exe		3,460 K	8,356 K	2240	Windows Security Health Se...	Microsoft Corporation
VGAuthService.exe		3,312 K	512 K	2256	VMware Guest Authentica...	VMware, Inc.
MsMpEng.exe	0.10	154,244 K	84,852 K	2284	Antimalware Service Execut...	Microsoft Corporation

CPU Usage: 5.41% Commit Charge: 49.36% Processes: 83 Physical Usage: 83.03%

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

```
Terminal
Terminal
Terminal
sansforensics@siftworkstation -> /tmp
$ vol.py -f stuxnet.vmem --profile=WinXPSP2x86 pslist | grep --color -P "lsass.exe|"
Volatility Foundation Volatility Framework 2.6.1
Offset(V)  Name  PID  PPID  Thds  Hnds  Sess  Wow64  Start  Exit
-----
0x823c8830 System 4 0 59 403 ----- 0 2010-10-29 17:08:53 UTC+0000
0x820df020 smss.exe 376 4 3 19 ----- 0 2010-10-29 17:08:54 UTC+0000
0x821a2da0 csrss.exe 600 376 11 395 0 0 2010-10-29 17:08:54 UTC+0000
0x81da5650 winlogon.exe 624 376 19 570 0 0 2010-10-29 17:08:54 UTC+0000
0x82073020 services.exe 668 624 21 431 0 0 2010-10-29 17:08:54 UTC+0000
0x81e70020 lsass.exe 680 624 19 342 0 0 2010-10-29 17:08:54 UTC+0000
0x823315d8 vmacthlp.exe 844 668 1 25 0 0 2010-10-29 17:08:55 UTC+0000
0x81db8da0 svchost.exe 856 668 17 193 0 0 2010-10-29 17:08:55 UTC+0000
0x81e61da0 svchost.exe 940 668 13 312 0 0 2010-10-29 17:08:55 UTC+0000
0x822843e8 svchost.exe 1032 668 61 1169 0 0 2010-10-29 17:08:55 UTC+0000
0x81e18b28 svchost.exe 1080 668 5 80 0 0 2010-10-29 17:08:55 UTC+0000
0x81ff7020 svchost.exe 1200 668 14 197 0 0 2010-10-29 17:08:55 UTC+0000
0x81fee8b0 spoolsv.exe 1412 668 10 118 0 0 2010-10-29 17:08:56 UTC+0000
0x81e0eda0 jqs.exe 1580 668 5 148 0 0 2010-10-29 17:09:05 UTC+0000
0x81fe52d0 vmttoolsd.exe 1664 668 5 284 0 0 2010-10-29 17:09:05 UTC+0000
0x821a0568 VMUpgradeHelper 1816 668 3 96 0 0 2010-10-29 17:09:08 UTC+0000
0x8205ada0 alg.exe 188 668 6 107 0 0 2010-10-29 17:09:09 UTC+0000
0x820ec7e8 explorer.exe 1196 1728 16 582 0 0 2010-10-29 17:11:49 UTC+0000
0x820ecc10 wscntfy.exe 2040 1032 1 28 0 0 2010-10-29 17:11:49 UTC+0000
0x81e86978 TSVNCache.exe 324 1196 7 54 0 0 2010-10-29 17:11:49 UTC+0000
0x81fc5da0 VMwareTray.exe 1912 1196 1 50 0 0 2010-10-29 17:11:50 UTC+0000
0x81e6b660 VMwareUser.exe 1356 1196 9 251 0 0 2010-10-29 17:11:50 UTC+0000
0x8210d478 jusched.exe 1712 1196 1 26 0 0 2010-10-29 17:11:50 UTC+0000
0x82279998 imapi.exe 756 668 4 116 0 0 2010-10-29 17:11:54 UTC+0000
0x822b9a10 wuauclt.exe 976 1032 3 133 0 0 2010-10-29 17:12:03 UTC+0000
0x81c543a0 Procmon.exe 660 1196 13 189 0 0 2011-06-03 04:25:56 UTC+0000
0x81fa5390 wmiprvse.exe 1872 856 5 134 0 0 2011-06-03 04:25:58 UTC+0000
0x81c498c8 lsass.exe 868 668 2 23 0 0 2011-06-03 04:26:55 UTC+0000
0x81c47c00 lsass.exe 1928 668 4 65 0 0 2011-06-03 04:26:55 UTC+0000
0x81c0cda0 cmd.exe 968 1664 0 ----- 0 0 2011-06-03 04:31:35 UTC+0000 2011-06-03 04:31:36 UTC+0000
0x81f14938 ipconfig.exe 304 968 0 ----- 0 0 2011-06-03 04:31:35 UTC+0000 2011-06-03 04:31:36 UTC+0000
sansforensics@siftworkstation -> /tmp
$
```

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.

The screenshot displays a Kali Linux desktop environment. In the background, a terminal window shows the execution of the `vol.py` command to scan a memory dump for connections. The output lists local and remote addresses along with their corresponding PIDs.

```
sansforensics@siftworkstation -> /tmp
$ vol.py -f zeus.vmem --profile=WinXPSP2x86 connschan
Volatility Foundation Volatility Framework 2.6.1
Offset(P) Local Address Remote Address Pid
-----
0x02214988 172.16.176.143:1054 193.104.41.75:80 856
0x06015ab0 0.0.0.0:1056 193.104.41.75:80 856
sansforensics@siftworkstation -> /tmp
$
```

In the foreground, a Mozilla Firefox browser window shows the VirusTotal analysis page for the URL `http://193.104.41.75/`. The page indicates that 2 engines detected this URL as malicious. The detection results are as follows:

DETECTION	DETAILS	RELATIONS	COMMUNITY
Dr.Web	Malicious	Forcepoint ThreatSeeker	Malicious
ADMINUSLabs	Clean	AegisLab WebGuard	Clean
AlienVault	Clean	Antiy-AVL	Clean
Avira (no cloud)	Clean	BADWARE.INFO	Clean
Baidu-International	Clean	BitDefender	Clean
Blueliv	Clean	CLEAN MX	Clean



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
```


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.

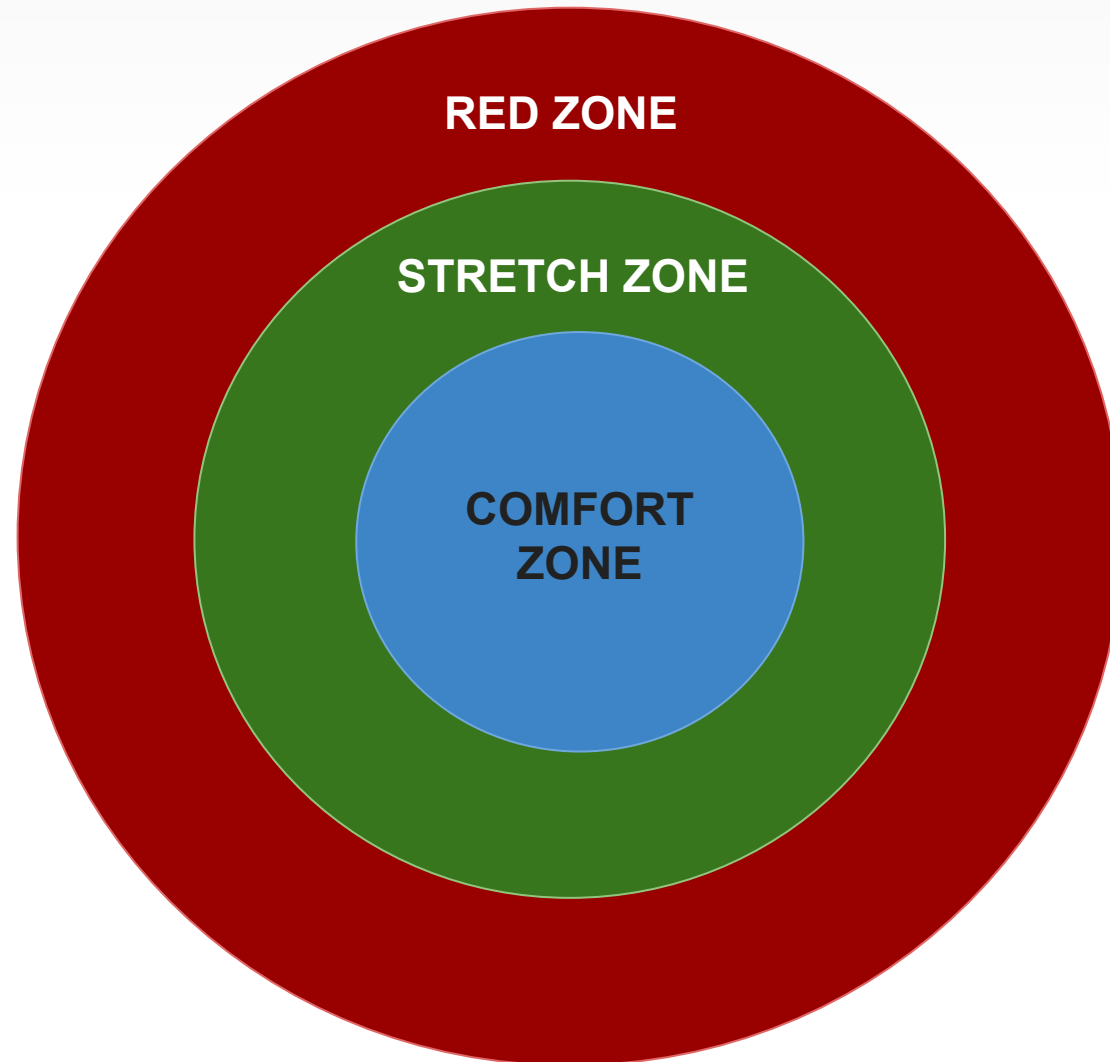
Environment & Tools

- 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**.

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

```
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    0xb000    0xffff                C:\WINDOWS\system32\ntdll.dll
0x7c800000    0xf400    0xffff
C:\WINDOWS\system32\kernel32.dll
0x77dd0000    0x9b00    0xffff
C:\WINDOWS\system32\ADVAPI32.dll
0x77e70000    0x9100    0xffff                C:\WINDOWS\system32\RPCRT4.dll
0x5cb70000    0x2600    0x1
C:\WINDOWS\system32\ShimEng.dll
0x6f880000    0x1ca00    0x1
```



Volatility includes the Malfind plugin that detects possible code injection.

Malfind detects executable sections in memory.

```
Terminal
Terminal
Terminal
sansforensics@siftworkstation -> /tmp
$ vol.py -f zeus.vmem --profile=WinXPSP2x86 malfind -p 856
Volatility Foundation Volatility Framework 2.6.1
Process: svchost.exe Pid: 856 Address: 0xb70000
Vad Tag: VadS.Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 38, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00b70000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ.....
0x00b70010 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
0x00b70020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00b70030 00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00 .....
0x00b70000 4d DEC EBP
0x00b70001 5a POP EDX
0x00b70002 90 NOP
0x00b70003 0003 ADD [EBX], AL
0x00b70005 0000 ADD [EAX], AL
0x00b70007 000400 ADD [EAX+EAX], AL
0x00b7000a 0000 ADD [EAX], AL
0x00b7000c ff DB 0xff
0x00b7000d ff00 INC DWORD [EAX]
0x00b7000f 00b800000000 ADD [EAX+0x0], BH
0x00b70015 0000 ADD [EAX], AL
0x00b70017 004000 ADD [EAX+0x0], AL
0x00b7001a 0000 ADD [EAX], AL
0x00b7001c 0000 ADD [EAX], AL
0x00b7001e 0000 ADD [EAX], AL
0x00b70020 0000 ADD [EAX], AL
0x00b70022 0000 ADD [EAX], AL
0x00b70024 0000 ADD [EAX], AL
0x00b70026 0000 ADD [EAX], AL
0x00b70028 0000 ADD [EAX], AL
```


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.

```
Files
sansforensics@siftworkstation -> /tmp
$ vol.py -f zeus.vmem --profile=WinXPSP2x86 procdump -p 856 --dump-dir ~/zeus-proc/
Volatility Foundation Volatility Framework 2.6.1
Process(V) ImageBase Name Result
-----
0x80ff88d8 0x01000000 svchost.exe OK: executable.856.exe
sansforensics@siftworkstation -> /tmp
$ xxd ~/zeus-proc/executable.856.exe | head -n 20
00000000: 4d5a 9000 0300 0000 0400 0000 ffff 0000  MZ.....
00000010: b800 0000 0000 0000 4000 0000 0000 0000  .....@.....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000030: 0000 0000 0000 0000 0000 0000 e000 0000  .....
00000040: 0e1f ba0e 00b4 09cd 21b8 014c cd21 5468  .....!..L..!Th
00000050: 6973 2070 726f 6772 616d 2063 616e 6e6f  is program canno
00000060: 7420 6265 2072 756e 2069 6e20 444f 5320  t be run in DOS
00000070: 6d6f 6465 2e0d 0d0a 2400 0000 0000 0000  mode...$.
00000080: fca9 f566 b8c8 9b35 b8c8 9b35 b8c8 9b35  ...f...5...5...5
00000090: 7bc7 fb35 b9c8 9b35 7bc7 c635 b1c8 9b35  {...5...5{...5...5
000000a0: b8c8 9a35 e7c8 9b35 7bc7 c535 b9c8 9b35  ...5...5{...5...5
000000b0: 7bc7 c435 b4c8 9b35 7bc7 c135 b9c8 9b35  {...5...5{...5...5
000000c0: 5269 6368 b8c8 9b35 0000 0000 0000 0000  Rich...5.....
000000d0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000000e0: 5045 0000 4c01 0300 d67e 1041 0000 0000  PE..L...~.A...
000000f0: 0000 0000 e000 0f01 0b01 070a 002c 0000  .....
00000100: 0008 0000 0000 0000 0925 0000 0010 0000  .....%.....
00000110: 0040 0000 0000 0001 0010 0000 0002 0000  ..@.....
00000120: 0500 0100 0500 0100 0400 0000 0000 0000  .....
00000130: 0060 0000 0004 0000 dd9c 0000 0200 0084  .....
sansforensics@siftworkstation -> /tmp
$
```


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.

```
Files
sansforensics@siftworkstation -> /tmp
$ vol.py -f zeus.vmem --profile=WinXPSP2x86 dlldump -p 856 --dump-dir ~/zeus-dlls/
Volatility Foundation Volatility Framework 2.6.1
Process(V) Name Module Base Module Name Result
-----
0x80ff88d8 svchost.exe 0x001000000 svchost.exe OK: module.856.115b8d8.1000000.dll
0x80ff88d8 svchost.exe 0x07c900000 ntdll.dll OK: module.856.115b8d8.7c900000.dll
0x80ff88d8 svchost.exe 0x076b40000 WINMM.dll OK: module.856.115b8d8.76b40000.dll
0x80ff88d8 svchost.exe 0x076e10000 adsldpc.dll Error: DllBase is paged
0x80ff88d8 svchost.exe 0x0769c0000 USERENV.dll Error: e_magic 23D8 is not a valid DOS signature.
0x80ff88d8 svchost.exe 0x077f60000 SHLWAPI.dll OK: module.856.115b8d8.77f60000.dll
0x80ff88d8 svchost.exe 0x05ad70000 UxTheme.dll Error: DllBase is paged
0x80ff88d8 svchost.exe 0x076e80000 rtutils.dll
0x80ff88d8 svchost.exe 0x020000000 xpsp2res.dll
0x80ff88d8 svchost.exe 0x077b40000 Apphelp.dll
0x80ff88d8 svchost.exe 0x0771b0000 WININET.dll
0x80ff88d8 svchost.exe 0x076c90000 IMAGEHLP.dll
0x80ff88d8 svchost.exe 0x076bc0000 REGAPI.dll
0x80ff88d8 svchost.exe 0x077dd0000 ADVAPI32.dll
0x80ff88d8 svchost.exe 0x077a80000 CRYPT32.dll
0x80ff88d8 svchost.exe 0x077be0000 MSACM32.dll
0x80ff88d8 svchost.exe 0x071a90000 wshtcpip.dll
0x80ff88d8 svchost.exe 0x077c00000 VERSION.dll
0x80ff88d8 svchost.exe 0x0722b0000 sensapi.dll
0x80ff88d8 svchost.exe 0x076fd0000 CLBCATQ.DLL
0x80ff88d8 svchost.exe 0x077d40000 USER32.dll
0x80ff88d8 svchost.exe 0x076bf0000 PSAPI.DLL
0x80ff88d8 svchost.exe 0x071a50000 mswsock.dll
0x80ff88d8 svchost.exe 0x00ff00000 rsaenh.dll
0x80ff88d8 svchost.exe 0x05b860000 NETAPI32.dll
0x80ff88d8 svchost.exe 0x06f880000 AcGenral.DLL
0x80ff88d8 svchost.exe 0x077690000 NTMARTA.DLL
0x80ff88d8 svchost.exe 0x071ab0000 WS2_32.dll
0x80ff88d8 svchost.exe 0x076f60000 WLDAP32.dll
0x80ff88d8 svchost.exe 0x071ad0000 wsock32.dll
0x80ff88d8 svchost.exe 0x0774e0000 ole32.dll
0x80ff88d8 svchost.exe 0x077920000 SETUPAPI.dll
0x80ff88d8 svchost.exe 0x077f10000 GDI32.dll
0x80ff88d8 svchost.exe 0x077120000 OLEAUT32.dll
0x80ff88d8 svchost.exe 0x077cc0000 ACTIVEDS.dll
```


zeus-dlls

module.856.115b8d8.85b860000.dll
module.856.115b8d8.85d090000.dll
module.856.115b8d8.87c9c0000.dll
module.856.115b8d8.87c800000.dll
module.856.115b8d8.87c900000.dll
module.856.115b8d8.871a50000.dll
module.856.115b8d8.871a90000.dll
module.856.115b8d8.871aa0000.dll
module.856.115b8d8.871ab0000.dll
module.856.115b8d8.871ad0000.dll
module.856.115b8d8.876b40000.dll
module.856.115b8d8.876bc0000.dll
module.856.115b8d8.876bf0000.dll
module.856.115b8d8.876d60000.dll
module.856.115b8d8.876e80000.dll
module.856.115b8d8.876e90000.dll
module.856.115b8d8.876eb0000.dll
module.856.115b8d8.876ee0000.dll
module.856.115b8d8.876f20000.dll
module.856.115b8d8.876f60000.dll
module.856.115b8d8.876fb0000.dll
module.856.115b8d8.876fc0000.dll
module.856.115b8d8.876fd0000.dll
module.856.115b8d8.877a80000.dll
module.856.115b8d8.877b20000.dll
module.856.115b8d8.877c00000.dll
module.856.115b8d8.877c10000.dll
module.856.115b8d8.877c70000.dll
module.856.115b8d8.877d40000.dll
module.856.115b8d8.877dd0000.dll

Error: DllBase is paged

File & Process Dumping

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.

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 envvars -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.

System Investigation

Additional Plugins



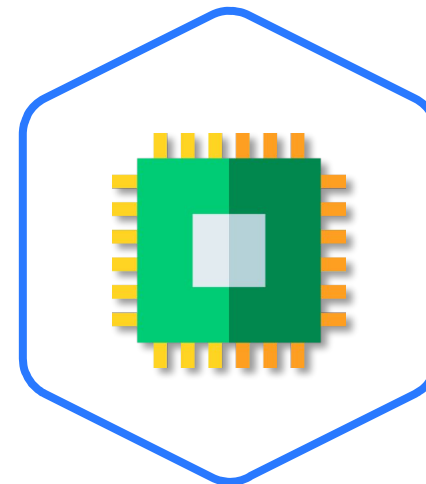
evtlogs



getsids



iehistory



modscan

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?