Memory Forensics

Memory Forensics Definition & Process

`Memory forensics`, also known as volatile memory analysis, is a specialized branch of digital forensics that focuses on the examination and analysis of the volatile memory (RAM) of a computer or digital device. Unlike traditional digital forensics, which involves analyzing data stored on non-volatile storage media like hard drives or solid-state drives, memory forensics deals with the live state of a system at a particular moment in time.

Here are some types of data found in RAM that are valuable for incident investigation:

- `Network connections`
- `File handles and open Files`
- `Open registry keys`
- `Running processes on the system`
- `Loaded modules`
- `Loaded device drivers`
- `Command history and console sessions`
- `Kernel data structures`
- `User and credential information`
- `Malware artifacts`
- `System configuration`
- `Process memory regions`

As we discussed both in the previous section and in the `YARA & Sigma for SOC Analysts` module, when malware operates, it often leaves traces or footprints in a system's active memory. By analyzing this memory, investigators can uncover malicious processes, identify indicators of compromise, and reconstruct the malware's actions.

It should be noted that in some cases, important data or encryption keys may reside in memory. Memory forensics can help recover this data, which may be crucial for an investigation.

The following outlines a systematic approach to memory forensics, formulated to aid in in-memory investigations and drawing inspiration from SANS's six-step memory forensics methodology.

1. `Process Identification and Verification`: Let's begin by identifying all active processes. Malicious software often masquerades as legitimate processes, sometimes with subtle name variations to avoid detection. We need to:
    - `Enumerate all running processes.`
    - `Determine their origin within the operating system.`
    - `Cross-reference with known legitimate processes.`
    - `Highlight any discrepancies or suspicious naming conventions.`
2. `Deep Dive into Process Components`: Once we've flagged potentially rogue processes, our next step is to scrutinize the associated Dynamic Link Libraries (DLLs) and handles. Malware often exploits DLLs to conceal its activities. We should:
    - `Examine DLLs linked to the suspicious process.`
    - `Check for unauthorized or malicious DLLs.`
    - `Investigate any signs of DLL injection or hijacking.`
3. `Network Activity Analysis`: Many malware strains, especially those that operate in stages, necessitate internet connectivity. They might beacon to Command and Control (C2) servers or exfiltrate data. To uncover these:

? Go to Questions

Table of Contents

Introduction to Digital Forensics
Windows Forensics Overview
Evidence Acquisition Techniques & Tools

Evidence Examination & Analysis

Memory Forensics
Disk Forensics
Rapid Triage Examination & Analysis Tools
Practical Digital Forensics Scenario

Skills Assessment

Skills Assessment

    - Review active and passive network connections in the system's memory.
    - Identify and document external IP addresses and associated domains.
    - Determine the nature and purpose of the communication.
        - Validate the process' legitimacy.
        - Assess if the process typically requires network communication.
        - Trace back to the parent process.
        - Evaluate its behavior and necessity.
4. `Code Injection Detection`: Advanced adversaries often employ techniques like process hollowing or utilize unmapped memory sections. To counter this, we should:
    - Use memory analysis tools to detect anomalies or signs of these techniques.
    - Identify any processes that seem to occupy unusual memory spaces or exhibit unexpected behaviors.
5. `Rootkit Discovery`: Achieving stealth and persistence is a common goal for adversaries. Rootkits, which embed deep within the OS, grant threat actors continuous, often elevated, system access while evading detection. To tackle this:
    - Scan for signs of rootkit activity or deep OS alterations.
    - Identify any processes or drivers operating at unusually high privileges or exhibiting stealth behaviors.
6. `Extraction of Suspicious Elements`: After pinpointing suspicious processes, drivers, or executables, we need to isolate them for in-depth analysis. This involves:
    - Dumping the suspicious components from memory.
    - Storing them securely for subsequent examination using specialized forensic tools.

The Volatility Framework

The preferred tool for conducting memory forensics is Volatility. Volatility is a leading open-source memory forensics framework. At the heart of this framework lies the Volatility Python script. This script harnesses a plethora of plugins, enabling it to dissect memory images with precision. Given its Python foundation, we can execute Volatility on any platform that's Python-compatible. Moreover, our team can leverage Volatility to scrutinize memory image files from a broad spectrum of widely-used operating systems. This includes Windows, spanning from Windows XP to Windows Server 2016, macOS, and, of course, prevalent Linux distributions.

Volatility modules or plugins are extensions or add-ons that enhance the functionality of the Volatility Framework by extracting specific information or perform specific analysis tasks on memory images.

Some commonly used modules include:

- **`pslist`**: Lists the running processes.
- **`cmdline`**: Displays process command-line arguments
- **`netscan`**: Scans for network connections and open ports.
- **`malfind`**: Scans for potentially malicious code injected into processes.
- **`handles`**: Scans for open handles
- **`svcscan`**: Lists Windows services.
- **`dlllist`**: Lists loaded DLLs (Dynamic-link Libraries) in a process.
- **`hivelist`**: Lists the registry hives in memory.

Volatility offers extensive documentation. You can find modules and their associated documentation using the following links:

- **`Volatility v2`**: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference
- **`Volatility v3`**: https://volatility3.readthedocs.io/en/latest/index.html

A useful Volatility (v2 & v3) cheatsheet can be found here: https://blog.onfvp.com/post/volatility-cheatsheet/

Volatility v2 Fundamentals

Let's now navigate to the bottom of this section and click on "Click here to spawn the target system!". Then, let's SSH into the Target IP using the provided credentials. The vast majority of the actions/commands covered from this point up to end of this section can be replicated inside the target, offering a more comprehensive grasp of the topics presented.

Let's now see a demonstration of utilizing `Volatility v2` to analyze a memory dump saved as `Win7-2515534d.vmem` inside the `/home/htb-student/MemoryDumps` directory of this section's target.

Volatility's `help` section and `available plugins` can be seen as follows.

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py --help
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help            list all available options and their default values.
                        Default values may be set in the configuration file
                        (/etc/volatilityrc)
  --conf-file=/home/htb-student/.volatilityrc
                        User based configuration file
  -d, --debug           Debug volatility
  --plugins=PLUGINS     Additional plugin directories to use (colon separated)
  --info                Print information about all registered objects
  --cache-directory=/home/htb-student/.cache/volatility
                        Directory where cache files are stored
  --cache               Use caching
  --tz=TZ               Sets the (Olson) timezone for displaying timestamps
                        using pytz (if installed) or tzset
  -C 190000, --confsize=190000
                        Config data size
  -Y YARAOFFSET, --yaraoffset=YARAOFFSET
                        YARA start offset
  -f FILENAME, --filename=FILENAME
                        Filename to use when opening an image
  --profile=WinXPSP2x86
                        Name of the profile to load (use --info to see a list
                        of supported profiles)
  -l LOCATION, --location=LOCATION
                        A URN location from which to load an address space
  -w, --write           Enable write support
  --dtb=DTB             DTB Address
  --physical_shift=PHYSICAL_SHIFT
                        Linux kernel physical shift address
  --virtual_shift=VIRTUAL_SHIFT
                        Linux kernel virtual shift address
  --shift=SHIFT         Mac KASLR shift address
  --output=text         Output in this format (support is module specific, see
                        the Module Output Options below)
  --output-file=OUTPUT_FILE
                        Write output in this file
  -v, --verbose         Verbose information
  -g KDBG, --kdbg=KDBG  Specify a KDBG virtual address (Note: for 64-bit
                        Windows 8 and above this is the address of
                        KdCopyDataBlock)
  --force               Force utilization of suspect profile
  -k KPCR, --kpcr=KPCR  Specify a specific KPCR address
  --cookie=COOKIE       Specify the address of nt!ObHeaderCookie (valid for
                        Windows 10 only)

        Supported Plugin Commands:

                agtidconfig     Parse the Agtid configuration
                amcache         Print AmCache information
                antianalysis
                apifinder
                apihooks        Detect API hooks in process and kernel memory
                apihooksdeep    Detect API hooks in process and kernel memory, with ssdeep for whit
                apt17scan       Detect processes infected with APT17 malware
                atoms           Print session and window station atom tables
                atomscan        Pool scanner for atom tables
```

```
attributeht     Find Hacking Team implants and attempt to attribute them using a wa
auditpol        Prints out the Audit Policies from HKLM\SECURITY\Policy\PolAdtEv
autoruns        Searches the registry and memory space for applications running at
bigpools        Dump the big page pools using BigPagePoolScanner
bioskbd         Reads the keyboard buffer from Real Mode memory
bitlocker       Extract Bitlocker FVEK. Supports Windows 7 - 10.
cachedump       Dumps cached domain hashes from memory
callbacks       Print system-wide notification routines
callstacks      this is the plugin class for callstacks
chromecookies   Scans for and parses potential Chrome cookie data
chromedownloadchains    Scans for and parses potential Chrome download chain record
chromedownloads Scans for and parses potential Chrome download records
chromehistory   Scans for and parses potential Chrome url history
chromesearchterms       Scans for and parses potential Chrome keyword search terms
chromevisits    Scans for and parses potential Chrome url visits data -- VERY SLOW,
clipboard       Extract the contents of the windows clipboard
cmdline         Display process command-line arguments
cmdscan         Extract command history by scanning for _COMMAND_HISTORY
connections     Print list of open connections [Windows XP and 2003 Only]
connscan        Pool scanner for tcp connections
consoles        Extract command history by scanning for _CONSOLE_INFORMATION
crashinfo       Dump crash-dump information
derusbiconfig   Parse the Derusbi configuration
deskscan        Poolscaner for tagDESKTOP (desktops)
devicetree      Show device tree
directoryenumerator     Enumerates all unique directories from FileScan
dlldump         Dump DLLs from a process address space
dlllist         Print list of loaded dlls for each process
driverbl        Scans memory for driver objects and compares the results with the b
driverirp       Driver IRP hook detection
driveritem
drivermodule    Associate driver objects to kernel modules
driverscan      Pool scanner for driver objects
dumpcerts       Dump RSA private and public SSL keys
dumpfiles       Extract memory mapped and cached files
dumpregistry    Dumps registry files out to disk
dyrescan        Extract Dyre Configuration from processes
editbox         Displays information about Edit controls. (Listbox experimental.)
envars          Display process environment variables
eventhooks      Print details on windows event hooks
evtlogs         Extract Windows Event Logs (XP/2003 only)
facebook        Retrieve facebook artifacts from a memory image
facebookcontacts        Finds possible Facebook contacts
facebookgrabinfo        Carves the memory dump for Owner's personal info JSON struc
facebookmessages        Carves the memory for every message exchanged between the O
fileitem
filescan        Pool scanner for file objects
firefoxcookies  Scans for and parses potential Firefox cookies (cookies.sqlite moz_
firefoxdownloads        Scans for and parses potential Firefox download records --
firefoxhistory  Scans for and parses potential Firefox url history (places.sqlite m
fwhooks         Enumerates modules which are using Firewall Hook Drivers on Windows
gahti           Dump the USER handle type information
gditimers       Print installed GDI timers and callbacks
gdt             Display Global Descriptor Table
getservicesids  Get the names of services in the Registry and return Calculated SID
getsids         Print the SIDs owning each process
ghostrat        Detects and decrypts Gh0stRat communication
handles         Print list of open handles for each process
hashdump        Dumps passwords hashes (LM/NTLM) from memory
hibinfo         Dump hibernation file information
hikitconfig     Parse the Hikit configuration
hivedump        Prints out a hive
hivelist        Print list of registry hives.
hivescan        Pool scanner for registry hives
hollowfind      Detects different types of Process Hollowing
hookitem
hpakextract     Extract physical memory from an HPAK file
hpakinfo        Info on an HPAK file
hpv_clipboard   Dump Virtual Machine Clipboard data
hpv_vmconnect   Virtual Machine Console data
hpv_vmwp        Display the Virtual Machine Process GUID for each running vm
idt             Display Interrupt Descriptor Table
idxparser       Scans for and parses Java IDX files
iehistory       Reconstruct Internet Explorer cache / history
imagecopy       Copies a physical address space out as a raw DD image
imageinfo       Identify information for the image
impscan         Scan for calls to imported functions
indx            Scans for and parses potential INDX entries
joblinks        Print process job link information
kdbgscan        Search for and dump potential KDBG values
kpcrscan        Search for and dump potential KPCR values
lastpass        Extract lastpass data from process.
ldrmodules      Detect unlinked DLLs
linuxgetprofile Scan to try to determine the Linux profile
```

```
logfile          Scans for and parses potential $Logfile entries
lsadump          Dump (decrypted) LSA secrets from the registry
machoinfo        Dump Mach-O file format information
malfind          Find hidden and injected code
malfinddeep      Find hidden and injected code, whitelist with ssdeep hashes
malfofind        Find indications of process hollowing/RunPE injections
malprocfind      Finds malicious processes based on discrepancies from observed, nor
malthfind        Find malicious threads by analyzing their callstack
mbrparser        Scans for and parses potential Master Boot Records (MBRs)
memdump          Dump the addressable memory for a process
memmap           Print the memory map
messagehooks     List desktop and thread window message hooks
mftparser        Scans for and parses potential MFT entries
mimikatz         mimikatz offline
moddump          Dump a kernel driver to an executable file sample
modscan          Pool scanner for kernel modules
modules          Print list of loaded modules
msdecompress     Carves and dumps Lznt1, Xpress and Xpress huffman Compressioned dat
multiscan        Scan for various objects at once
mutantscan       Pool scanner for mutex objects
ndispktscan      Extract the packets from memory
networkpackets   Carve and analyze ARP/IPv4 network packets from memory
notepad          List currently displayed notepad text
objtypescan      Scan for Windows object type objects
openioc_scan     Scan OpenIOC 1.1 based indicators
openvpn          Extract OpenVPN client credentials (username, password) cached in m
osint            Check Url/ip extracted from memory against opensource intelligence
patcher          Patches memory based on page scans
plugxconfig      Locate and parse the PlugX configuration
plugxscan        Detect processes infected with PlugX
poolpeek         Configurable pool scanner plugin
prefetchparser   Scans for and parses potential Prefetch files
printkey         Print a registry key, and its subkeys and values
privs            Display process privileges
procdump         Dump a process to an executable file sample
processbl        Scans memory for processes and loaded DLLs and compares the results
profilescan      Scan for executables to try to determine the underlying OS
psinfo           Displays process related information and suspicious memory regions
pslist           Print all running processes by following the EPROCESS lists
psscan           Pool scanner for process objects
pstotal          Combination of pslist,psscan & pstree --output=dot gives graphical
pstree           Print process list as a tree
psxview          Find hidden processes with various process listings
qemuinfo         Dump Qemu information
raw2dmp          Converts a physical memory sample to a windbg crash dump
registryitem
rsakey           Extract base64/PEM encoded private RSA keys from physical memory.
screenshot       Save a pseudo-screenshot based on GDI windows
servicebl        Scans memory for service objects and compares the results with the
servicediff      List Windows services (ala Plugx)
serviceitem
sessions         List details on _MM_SESSION_SPACE (user logon sessions)
shellbags        Prints ShellBags info
shimcache        Parses the Application Compatibility Shim Cache registry key
shimcachemem     Parses the Application Compatibility Shim Cache stored in kernel me
shutdowntime     Print ShutdownTime of machine from registry
sockets          Print list of open sockets
sockscan         Pool scanner for tcp socket objects
ssdeepscan       Scan process or kernel memory with SSDeep signatures
ssdt             Display SSDT entries
strings          Match physical offsets to virtual addresses (may take a while, VERY
svcscan          Scan for Windows services
symlinkscan      Pool scanner for symlink objects
systeminfo       Print common system details of machine from registry
thrdscan         Pool scanner for thread objects
threads          Investigate _ETHREAD and _KTHREADs
timeliner        Creates a timeline from various artifacts in memory
timers           Print kernel timers and associated module DPCs
truecryptmaster  Recover TrueCrypt 7.1a Master Keys
truecryptpassphrase     TrueCrypt Cached Passphrase Finder
truecryptsummary        TrueCrypt Summary
trustrecords     Extract MS Office TrustRecords from the Registry
twitter          Retrieve twitter artifacts from a memory image
uninstallinfo    Extract installed software info from Uninstall registry key
unloadedmodules  Print list of unloaded modules
usbstor          Parse USB Data from the Registry
userassist       Print userassist registry keys and information
userhandles      Dump the USER handle tables
usnjrnl          Scans for and parses potential USNJRNL entries
usnparser        Scans for and parses USN journal records
vaddump          Dumps out the vad sections to a file
vadinfo          Dump the VAD info
vadtree          Walk the VAD tree and display in tree format
vadwalk          Walk the VAD tree
vboxinfo         Dump virtualbox information
```

```
vboxinfo        Dump virtualbox information
verinfo         Prints out the version information from PE images
vmwareinfo      Dump VMware VMSS/VMSN information
volshell        Shell in the memory image
windows         Print Desktop Windows (verbose details)
wintree         Print Z-Order Desktop Windows Tree
wndscan         Pool scanner for window stations
yarascan        Scan process or kernel memory with Yara signatures
```

Identifying the Profile

Profiles are essential for Volatility v2 to interpret the memory data correctly (profile identification has been enhanced in v3). To determine the profile that matches the operating system of the memory dump we can use the `imageinfo` plugin as follows.

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem imageinfo
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Wi
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/htb-student/MemoryDumps/Win7-2515534d.vmem
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf80002be9120L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff80002beb000L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2023-06-22 12:34:03 UTC+0000
     Image local date and time : 2023-06-22 18:04:03 +0530
```

Identifying Running Processes

Let's see if the suggested `Win7SP1x64` profile is correct by trying to list running process via the `pslist` plugin.

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
------------------ -------------------- ------ ------ ------ -------- ------ ------ --------------
0xfffffa8000ca8860 System                    4      0     97      446 ------      0 2023-06-22 12:0
0xfffffa8001a64920 smss.exe                264      4      2       29 ------      0 2023-06-22 12:0
0xfffffa80028a39a0 csrss.exe               352    344      8      626      0      0 2023-06-22 12:0
0xfffffa8002a51730 wininit.exe             404    344      3       76      0      0 2023-06-22 12:0
0xfffffa800291eb00 csrss.exe               416    396      9      307      1      0 2023-06-22 12:0
0xfffffa8002a86340 winlogon.exe            464    396      3      113      1      0 2023-06-22 12:0
0xfffffa8002ad8b00 services.exe            508    404      8      226      0      0 2023-06-22 12:0
0xfffffa8002adbb00 lsass.exe               516    404      6      585      0      0 2023-06-22 12:0
0xfffffa8002ae6b00 lsm.exe                 524    404      9      149      0      0 2023-06-22 12:0
0xfffffa8002b4f720 svchost.exe             628    508     10      366      0      0 2023-06-22 12:0
0xfffffa8002b7bb00 svchost.exe             696    508      7      288      0      0 2023-06-22 12:0
0xfffffa8002ba0b00 svchost.exe             744    508     18      455      0      0 2023-06-22 12:0
0xfffffa8002c00280 svchost.exe             868    508     19      443      0      0 2023-06-22 12:0
0xfffffa8002c52710 svchost.exe             920    508     17      599      0      0 2023-06-22 12:0
0xfffffa8002c5c680 svchost.exe             964    508     28      838      0      0 2023-06-22 12:0
0xfffffa80022679b0 svchost.exe            1000    508     13      365      0      0 2023-06-22 12:0
0xfffffa8002d15b00 spoolsv.exe            1120    508     13      273      0      0 2023-06-22 12:0
0xfffffa8002d4f9b0 svchost.exe            1156    508     18      308      0      0 2023-06-22 12:0
0xfffffa8002d2f060 svchost.exe            1268    508     11      165      0      0 2023-06-22 12:0
0xfffffa8002d2d060 svchost.exe            1348    508     15      258      0      0 2023-06-22 12:0
0xfffffa8000d78b00 VGAuthService.         1412    508      4       96      0      0 2023-06-22 12:0
0xfffffa8002db6b00 vm3dservice.ex         1440    508      4       61      0      0 2023-06-22 12:0
0xfffffa8002e2e9b0 vmtoolsd.exe           1468    508     13      299      0      0 2023-06-22 12:0
0xfffffa8002e45a70 vm3dservice.ex         1488   1440      2       45      1      0 2023-06-22 12:0
0xfffffa8002f58b00 svchost.exe            1724    508      6       92      0      0 2023-06-22 12:0
0xfffffa8002fa2b00 WmiPrvSE.exe           1908    628      9      197      0      0 2023-06-22 12:0
0xfffffa8002f8fb00 dllhost.exe            1968    508     13      190      0      0 2023-06-22 12:0
0xfffffa8003007b00 msdtc.exe              1960    508     12      145      0      0 2023-06-22 12:0
0xfffffa8001bfbb00 taskhost.exe           2432    508      9      241      1      0 2023-06-22 12:0
0xfffffa80027ca970 dwm.exe                2484    868      5      152      1      0 2023-06-22 12:0
0xfffffa8001d27b00 explorer.exe           2508   2472     24      843      1      0 2023-06-22 12:0
0xfffffa80123fc590 vmtoolsd.exe           2600   2508      8      182      1      0 2023-06-22 12:0
```

```
0xfffffa80027edb00 SearchIndexer.         2756    508     17      800      0      0 2023-06-22 12:0
0xfffffa80023e7750 cmd.exe                3040   2508      1       21      1      0 2023-06-22 12:0
0xfffffa8001d19060 conhost.exe            3048    416      2       53      1      0 2023-06-22 12:0
0xfffffa8002d95870 taskmgr.exe            2648    464      6      113      1      0 2023-06-22 12:0
0xfffffa8000e0fb00 ProcessHacker.          716   2508      9      476      1      0 2023-06-22 12:0
0xfffffa8000eee060 sppsvc.exe             1080    508      4      146      0      0 2023-06-22 12:0
0xfffffa8000ea6a00 svchost.exe             608    508     15      431      0      0 2023-06-22 12:0
0xfffffa8000e2e620 wmpnetwk.exe           2968    508     18      442      0      0 2023-06-22 12:0
0xfffffa80022af430 ida64.exe              2248   2508      7      340      1      0 2023-06-22 12:1
0xfffffa8001420300 x32dbg.exe             2820   2508     20      480      1      1 2023-06-22 12:2
0xfffffa8000ee96d0 Ransomware.wan         1512   2820     11      167      1      1 2023-06-22 12:2
0xfffffa8002ca4240 Ransomware.wan         2320    508    117      497      0      1 2023-06-22 12:3
0xfffffa8002ad9560 dllhost.exe            1876    628      4       79      1      0 2023-06-22 12:3
0xfffffa8001d0f8b0 tasksche.exe           2972   1512      0 --------      1      0 2023-06-22 12:3
0xfffffa8001d22b00 tasksche.exe           1792   1044      8       82      0      1 2023-06-22 12:3
0xfffffa8002fa3060 SearchProtocol          852   2756      8      289      0      0 2023-06-22 12:3
0xfffffa8002572060 @WanaDecryptor         1060   1792      2       71      0      1 2023-06-22 12:3
0xfffffa8001568060 taskhsvc.exe           3012   1060      4      101      0      1 2023-06-22 12:3
0xfffffa8001ddb060 conhost.exe            2348    352      1       32      0      0 2023-06-22 12:3
0xfffffa8000df81b0 VSSVC.exe               288    508      6      116      0      0 2023-06-22 12:3
0xfffffa800141e9a0 @WanaDecryptor         3252   3212      1       75      1      1 2023-06-22 12:3
0xfffffa80014e4a70 MpCmdRun.exe           3436   3412      5      116      0      0 2023-06-22 12:3
0xfffffa80014c12c0 SearchFilterHo         3904   2756      6      109      0      0 2023-06-22 12:3
0xfffffa8000f2f1c0 audiodg.exe            4048    744      6      128      0      0 2023-06-22 12:3
0xfffffa8000dbc5a0 cmd.exe                2080   1468      0 --------      0      0 2023-06-22 12:3
0xfffffa8000f90b00 conhost.exe            3292    352      0 --------      0      0 2023-06-22 12:3
0xfffffa8000f7b790 ipconfig.exe           2360   2080      0 --------      0      0 2023-06-22 12:3
```

It should be noted that even if we specify another profile from the suggested list Volatility may still provide us with the correct output.

Identifying Network Artifacts

The `netscan` plugin can be used to scan for network artifacts as follows.

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
Offset(P)          Proto    Local Address                  Foreign Address      State            Pi
0x1a15caa0         UDPv4    0.0.0.0:3702                   *:*                                   13
0x1a15caa0         UDPv6    :::3702                        *:*                                   13
0x1fd7cac0         TCPv4    0.0.0.0:49155                  0.0.0.0:0            LISTENING        50
0x1fd7cac0         TCPv6    :::49155                       :::0                 LISTENING        50
0x3da01a70         UDPv4    0.0.0.0:3702                   *:*                                   13
0x3da0b130         UDPv4    0.0.0.0:0                      *:*                                   10
0x3da0b130         UDPv6    :::0                           *:*                                   10
0x3dcf1010         UDPv4    0.0.0.0:62718                  *:*                                   13
0x3dcf15b0         UDPv4    0.0.0.0:62719                  *:*                                   13
0x3dcf15b0         UDPv6    :::62719                       *:*                                   13
0x3da15010         TCPv4    0.0.0.0:49156                  0.0.0.0:0            LISTENING        51
0x3da15010         TCPv6    :::49156                       :::0                 LISTENING        51
0x3dc69860         TCPv4    0.0.0.0:5357                   0.0.0.0:0            LISTENING        4
0x3dc69860         TCPv6    :::5357                        :::0                 LISTENING        4
0x3dca3ee0         TCPv4    0.0.0.0:49154                  0.0.0.0:0            LISTENING        96
0x3dca3ee0         TCPv6    :::49154                       :::0                 LISTENING        96
0x3dcf7280         TCPv4    0.0.0.0:49155                  0.0.0.0:0            LISTENING        50
0x3dd07540         TCPv4    0.0.0.0:445                    0.0.0.0:0            LISTENING        4
0x3dd07540         TCPv6    :::445                         :::0                 LISTENING        4
0x3e5f7cd0         UDPv4    0.0.0.0:3702                   *:*                                   13
0x3e5f7cd0         UDPv6    :::3702                        *:*                                   13
0x3deff8d0         TCPv4    0.0.0.0:10243                  0.0.0.0:0            LISTENING        4
0x3deff8d0         TCPv6    :::10243                       :::0                 LISTENING        4
0x3df01ba0         TCPv4    0.0.0.0:49154                  0.0.0.0:0            LISTENING        96
0x3e194410         TCPv4    0.0.0.0:135                    0.0.0.0:0            LISTENING        69
0x3e195840         TCPv4    0.0.0.0:135                    0.0.0.0:0            LISTENING        69
0x3e195840         TCPv6    :::135                         :::0                 LISTENING        69
0x3e1ab8f0         TCPv4    0.0.0.0:49152                  0.0.0.0:0            LISTENING        40
0x3e1fe300         TCPv4    0.0.0.0:49153                  0.0.0.0:0            LISTENING        74
0x3e1fe300         TCPv6    :::49153                       :::0                 LISTENING        74
0x3e1fecd0         TCPv4    0.0.0.0:49153                  0.0.0.0:0            LISTENING        74
0x3e963ad0         TCPv4    127.0.0.1:9050                 0.0.0.0:0            LISTENING        30
0x3ec4f620         TCPv4    0.0.0.0:49152                  0.0.0.0:0            LISTENING        40
0x3ec4f620         TCPv6    :::49152                       :::0                 LISTENING        40
0x3f1fd6f0         TCPv4    0.0.0.0:554                    0.0.0.0:0            LISTENING        29
0x3f1fd6f0         TCPv6    :::554                         :::0                 LISTENING        29
0x3ec2d010         TCPv4    127.0.0.1:50313                127.0.0.1:50314      ESTABLISHED      -1
```

```
0x3f3ced90         UDPv4    0.0.0.0:3702                   *:*                                    13
0x3f2284c0         TCPv4    0.0.0.0:49156                  0.0.0.0:0              LISTENING       51
0x3fcfd930         UDPv4    127.0.0.1:1900                 *:*                                    13
0x3fd1bbf0         UDPv6    ::1:61543                      *:*                                    13
0x3fd28310         UDPv4    127.0.0.1:61544                *:*                                    13
0x3fd2b420         UDPv6    ::1:1900                       *:*                                    13
0x3fd4a4a0         UDPv4    0.0.0.0:5004                   *:*                                    29
0x3fd4a4a0         UDPv6    :::5004                        *:*                                    29
0x3fd4aa90         UDPv4    0.0.0.0:5005                   *:*                                    29
0x3fd4adb0         UDPv4    0.0.0.0:5004                   *:*                                    29
0x3fd5fec0         UDPv4    0.0.0.0:5005                   *:*                                    29
0x3fd5fec0         UDPv6    :::5005                        *:*                                    29
0x3fc02ca0         TCPv4    0.0.0.0:554                    0.0.0.0:0              LISTENING       29
0x3fca6010         TCPv4    0.0.0.0:2869                   0.0.0.0:0              LISTENING       4
0x3fca6010         TCPv6    :::2869                        :::0                   LISTENING       4
0x3fc4f600         TCPv4    127.0.0.1:55206                127.0.0.1:9050         ESTABLISHED     -1
0x3fe604f0         TCPv4    127.0.0.1:9050                 127.0.0.1:55206        ESTABLISHED     -1
```

To find `_TCPT_OBJECT` structures using pool tag scanning, use the `connscan` command. This can find artifacts from previous connections that have since been terminated, in addition to the active ones.

Identifying Injected Code

The `malfind` plugin can be used to identify and extract injected code and malicious payloads from the memory of a running process as follows.

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
Process: svchost.exe Pid: 608 Address: 0x12350000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 128, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x0000000012350000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0000000012350010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0000000012350020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0000000012350030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

0x0000000012350000 0000            ADD [EAX], AL
0x0000000012350002 0000            ADD [EAX], AL
0x0000000012350004 0000            ADD [EAX], AL
0x0000000012350006 0000            ADD [EAX], AL
0x0000000012350008 0000            ADD [EAX], AL
0x000000001235000a 0000            ADD [EAX], AL
0x000000001235000c 0000            ADD [EAX], AL
0x000000001235000e 0000            ADD [EAX], AL
0x0000000012350010 0000            ADD [EAX], AL
0x0000000012350012 0000            ADD [EAX], AL
0x0000000012350014 0000            ADD [EAX], AL
0x0000000012350016 0000            ADD [EAX], AL
0x0000000012350018 0000            ADD [EAX], AL
0x000000001235001a 0000            ADD [EAX], AL
0x000000001235001c 0000            ADD [EAX], AL
0x000000001235001e 0000            ADD [EAX], AL
0x0000000012350020 0000            ADD [EAX], AL
0x0000000012350022 0000            ADD [EAX], AL
0x0000000012350024 0000            ADD [EAX], AL
0x0000000012350026 0000            ADD [EAX], AL
0x0000000012350028 0000            ADD [EAX], AL
0x000000001235002a 0000            ADD [EAX], AL
0x000000001235002c 0000            ADD [EAX], AL
0x000000001235002e 0000            ADD [EAX], AL
0x0000000012350030 0000            ADD [EAX], AL
0x0000000012350032 0000            ADD [EAX], AL
0x0000000012350034 0000            ADD [EAX], AL
0x0000000012350036 0000            ADD [EAX], AL
0x0000000012350038 0000            ADD [EAX], AL
0x000000001235003a 0000            ADD [EAX], AL
0x000000001235003c 0000            ADD [EAX], AL
0x000000001235003e 0000            ADD [EAX], AL
```

Identifying Handles

The `handles` plugin in Volatility is used for analyzing the handles (file and object references) held by a specific process

within a memory dump. Understanding the handles associated with a process can provide valuable insights during incident response and digital forensics investigations, as it reveals the resources and objects a process is interacting with. Here's how to use the handles plugin.

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
Offset(V)            Pid             Handle             Access Type            Details
------------------ ------ ---------------- ---------------- --------------- -------
0xfffff8a001628ee0   1512                0x4                0x9 Key             MACHINE\SOFTWARE\M
0xfffff8a00221e7e0   1512               0x14                0x9 Key             MACHINE\SOFTWARE\M
0xfffff8a0023b3490   1512               0x20            0x20019 Key             MACHINE\SYSTEM\CON
0xfffff8a001f1e300   1512               0x38            0xf003f Key             MACHINE
0xfffff8a001f3b410   1512               0x40                0x1 Key             MACHINE\SYSTEM\CON
0xfffff8a001f35280   1512               0x58                0x1 Key             MACHINE\SYSTEM\CON
0xfffff8a001f18440   1512               0x9c            0xf003f Key             USER\S-1-5-21-3232
0xfffff8a001d4e1f0   1512               0xa0            0x2001f Key             USER\S-1-5-21-3232
0xfffff8a00080e8a0   1512               0xc0            0xf003f Key             USER
0xfffff8a00237dc10   1512               0xe0                0x1 Key             USER\S-1-5-21-3232
0xfffff8a001f63a80   1512              0x120                0x1 Key             MACHINE\SOFTWARE\W
0xfffff8a00208b750   1512              0x124            0x20019 Key             MACHINE\SOFTWARE\P
0xfffff8a0022b6850   1512              0x128            0x20019 Key             USER\S-1-5-21-3232
0xfffff8a000d807b0   1512              0x12c            0x20019 Key             USER\S-1-5-21-3232
0xfffff8a0013b2920   1512              0x130            0x20019 Key             MACHINE\SOFTWARE\P
0xfffff8a001f7b610   1512              0x134            0x20019 Key             USER\S-1-5-21-3232
0xfffff8a0022f8ad0   1512              0x138            0x20019 Key             USER\S-1-5-21-3232
0xfffff8a0026778a0   1512              0x13c            0x20019 Key             MACHINE\SOFTWARE\W
0xfffff8a000f4fb00   1512              0x140            0x20019 Key             MACHINE\SOFTWARE\W
0xfffff8a001efb870   1512              0x154            0xf003f Key             MACHINE\SYSTEM\CON
0xfffff8a001f683c0   1512              0x15c            0xf003f Key             MACHINE\SYSTEM\CON
0xfffff8a001f17660   1512              0x164            0x20019 Key             USER\S-1-5-21-3232
0xfffff8a0012cbe90   1512              0x168            0x20019 Key             MACHINE\SOFTWARE\W
0xfffff8a00000c610   1512              0x1b8            0x2001f Key             USER\S-1-5-21-3232
0xfffff8a0025cf4c0   1512              0x1bc            0x20019 Key             MACHINE\SOFTWARE\W
0xfffff8a00125d610   1512              0x1d0                0xf Key             USER\S-1-5-21-3232
0xfffff8a0023dcdd0   1512              0x22c            0xf003f Key             MACHINE\SOFTWARE\C
```

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
Offset(V)            Pid             Handle             Access Type            Details
------------------ ------ ---------------- ---------------- --------------- -------
0xfffffa8001d162e0   1512               0x10           0x100020 File            \Device\HarddiskVo
0xfffffa800228adc0   1512               0x1c           0x100020 File            \Device\HarddiskVo
0xfffffa8000df8070   1512              0x110           0x12019f File            \Device\HarddiskVo
0xfffffa8002210cd0   1512              0x170           0x100080 File            \Device\Nsi
0xfffffa8000dedf20   1512              0x1e4           0x100001 File            \Device\KsecDD
0xfffffa8002f70700   1512              0x23c           0x120089 File            \Device\HarddiskVo
```

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
Offset(V)            Pid             Handle             Access Type            Details
------------------ ------ ---------------- ---------------- --------------- -------
0xfffffa8001d0f8b0   1512              0x29c           0x1fffff Process         tasksche.exe(2972)
```

Identifying Windows Services

The `svcscan` plugin in Volatility is used for listing and analyzing Windows services running on a system within a memory dump. Here's how to use the `svcscan` plugin.

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP
```

```
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
Offset: 0xb755a0
Order: 71
Start: SERVICE_AUTO_START
Process ID: 628
Service Name: DcomLaunch
Display Name: DCOM Server Process Launcher
Service Type: SERVICE_WIN32_SHARE_PROCESS
Service State: SERVICE_RUNNING
Binary Path: C:\Windows\system32\svchost.exe -k DcomLaunch

Offset: 0xb754b0
Order: 70
Start: SERVICE_DEMAND_START
Process ID: -
Service Name: dc21x4vm
Display Name: dc21x4vm
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_STOPPED
Binary Path: -

Offset: 0xb753c0
Order: 69
Start: SERVICE_AUTO_START
Process ID: 868
Service Name: CscService
Display Name: Offline Files
Service Type: SERVICE_WIN32_SHARE_PROCESS
Service State: SERVICE_RUNNING
Binary Path: C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted

Offset: 0xb770d0
Order: 68
Start: SERVICE_SYSTEM_START
Process ID: -
Service Name: CSC
Display Name: Offline Files Driver
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_RUNNING
Binary Path: \Driver\CSC
---SNIP---
```

Identifying Loaded DLLs

The `dlllist` plugin in Volatility is used for listing the dynamic link libraries (DLLs) loaded into the address space of a specific process within a memory dump. Here's how to use the `dlllist` plugin.

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
************************************************************************
Ransomware.wan pid:   1512
Command line : "C:\Users\Analyst\Desktop\Samples\Ransomware.wannacry.exe"


Base                             Size          LoadCount LoadTime                       Path
------------------ ------------------ ------------------ ------------------------------ ----
0x0000000000400000           0x66b000             0xffff 1970-01-01 00:00:00 UTC+0000   C:\Users\An
0x00000000773f0000           0x19f000             0xffff 1970-01-01 00:00:00 UTC+0000   C:\Windows\
0x00000000739d0000            0x3f000                0x3 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000073970000            0x5c000                0x1 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000073960000             0x8000                0x1 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000000400000           0x66b000             0xffff 1970-01-01 00:00:00 UTC+0000   C:\Users\An
0x00000000775b0000           0x180000             0xffff 1970-01-01 00:00:00 UTC+0000   C:\Windows\
0x0000000075b50000           0x110000             0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x00000000770c0000            0x47000             0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000074d30000            0xa1000             0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000077110000            0xac000             0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000075b30000            0x19000             0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000074de0000            0xf0000             0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000074cd0000            0x60000             0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000074cc0000             0xc000             0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x00000000755f0000            0x35000             0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000074f70000             0x6000             0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x000000006bb70000            0x66000             0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
```

```
0x000000000738f0000            0x1c000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x00000000737c0000              0x7000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000075160000            0x437000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000076050000              0x4000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000075e60000            0x100000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000074ee0000             0x90000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x00000000770a0000              0xa000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x00000000750c0000              0x9d000           0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x00000000770b0000              0x4000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000075c60000              0x57000           0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000074f80000              0x4000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x00000000737b0000              0x9000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000075f60000              0x3000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000075630000              0x3000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000076210000            0x236000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000075fa0000              0x5000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x00000000756d0000              0x17000           0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000075fb0000              0xb000            0xffff 2023-06-22 12:23:42 UTC+0000   C:\Windows\
0x0000000075ad0000             0x60000               0x2 2023-06-22 12:28:02 UTC+0000   C:\Windows\
0x00000000760b0000             0xcd000               0x1 2023-06-22 12:28:02 UTC+0000   C:\Windows\
0x000000006cd90000              0x8000               0x2 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x0000000076450000            0xc4c000               0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x0000000075850000            0x15f000              0x22 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x000000006cc30000              0x4000               0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x00000000771c0000              0x4000               0x2 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x000000006cc10000             0x12000               0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x000000006cc00000              0xd000               0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x000000006cbc0000              0x3c000              0x4 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x000000006cbb0000              0x6000               0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x000000006cd80000              0x4000               0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x00000000737d0000             0x44000               0x2 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x00000000756f0000            0x150000               0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x0000000075a30000             0x91000               0x4 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x000000006cba0000              0x5000               0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x0000000075640000             0x83000               0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x000000006bb10000              0x5a000              0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x000000006bb00000              0x10000              0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x000000006baf0000              0x6000               0x1 2023-06-22 12:28:06 UTC+0000   C:\Windows\
0x0000000071ac0000             0x17000               0x1 2023-06-22 12:30:20 UTC+0000   C:\Windows\
0x000000006d420000             0x3b000               0x1 2023-06-22 12:30:20 UTC+0000   C:\Windows\
0x0000000071ab0000              0xe000               0x1 2023-06-22 12:30:20 UTC+0000   C:\Windows\
0x000000006bae0000              0x8000               0x1 2023-06-22 12:30:20 UTC+0000   C:\Windows\
0x000000006ced0000              0x4c000            0xffff 2023-06-22 12:31:13 UTC+0000   C:\Windows\
```

Identifying Hives

The `hivelist` plugin in Volatility is used for listing the hives (registry files) present in the memory dump of a Windows system. Here's how to use the hivelist plugin.

Memory Forensics

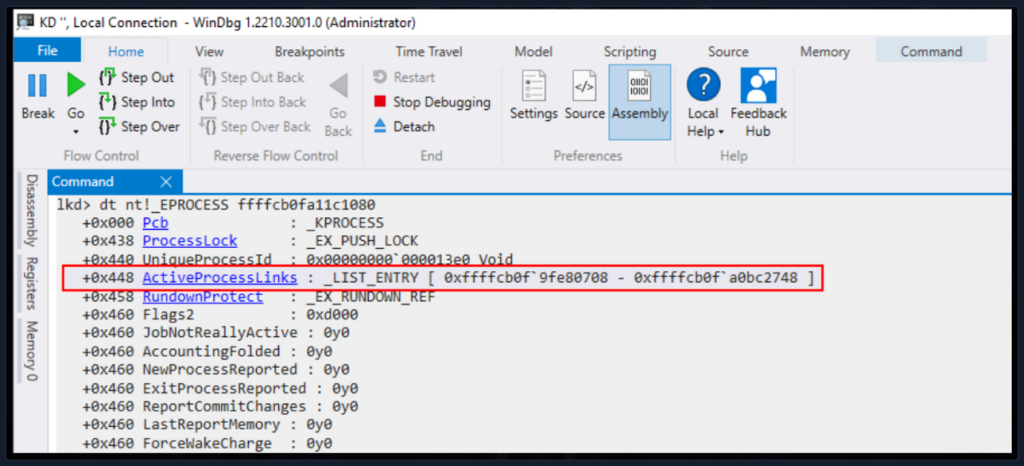
```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
Virtual            Physical           Name
------------------ ------------------ ----
0xfffff8a001710010 0x000000002c2e4010 \??\C:\Users\Analyst\AppData\Local\Microsoft\Windows\UsrClass
0xfffff8a001d4b410 0x000000001651f410 \??\C:\System Volume Information\Syscache.hve
0xfffff8a00000f010 0x0000000026de8010 [no name]
0xfffff8a000024010 0x00000000273f3010 \REGISTRY\MACHINE\SYSTEM
0xfffff8a000058010 0x0000000026727010 \REGISTRY\MACHINE\HARDWARE
0xfffff8a0000f7410 0x0000000019824410 \SystemRoot\System32\Config\DEFAULT
0xfffff8a000844010 0x000000001a979010 \Device\HarddiskVolume1\Boot\BCD
0xfffff8a0009d6010 0x000000001998d010 \SystemRoot\System32\Config\SOFTWARE
0xfffff8a000e0a010 0x0000000000724e010 \SystemRoot\System32\Config\SAM
0xfffff8a000e36010 0x0000000012f0e010 \SystemRoot\System32\Config\SECURITY
0xfffff8a000f7e010 0x0000000012f7b010 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0xfffff8a00100c410 0x0000000006de7410 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0xfffff8a0016a8010 0x000000002aecd010 \??\C:\Users\Analyst\ntuser.dat
```

Rootkit Analysis with Volatility v2

Let's now see a demonstration of utilizing `Volatility v2` to analyze a memory dump saved as `rootkit.vmem` inside the `/home/htb-student/MemoryDumps` directory of this section's target.

Understanding the EPROCESS Structure

EPROCESS is a data structure in the Windows kernel that represents a process. Each running process in the Windows operating system has a corresponding `EPROCESS` block in kernel memory. During memory analysis, the examination of `EPROCESS` structures is crucial for understanding the running processes on a system, identifying parent-child relationships, and determining which processes were active at the time of the memory capture.



FLINK and BLINK

A doubly-linked list is a fundamental data structure in computer science and programming. It is a type of linked list where each node (record) contains two references or pointers:

- **`Next Pointer`**: `This points to the next node in the list, allowing us to traverse the list in a forward direction.`
- **`Previous Pointer`**: `This points to the previous node in the list, allowing us to traverse the list in a backward direction.`

Within the `EPROCESS` structure, we have `ActiveProcessLinks` as the doubly-linked list which contains the `flink` field and the `blink` field.

- **`flink`**: `Is a forward pointer that points to the ActiveProcessLinks list entry of the _next_ EPROCESS structure in the list of active processes.`
- **`blink`**: `Is a backward pointer within the EPROCESS structure that points to the ActiveProcessLinks list entry of the _previous_ EPROCESS structure in the list of active processes.`

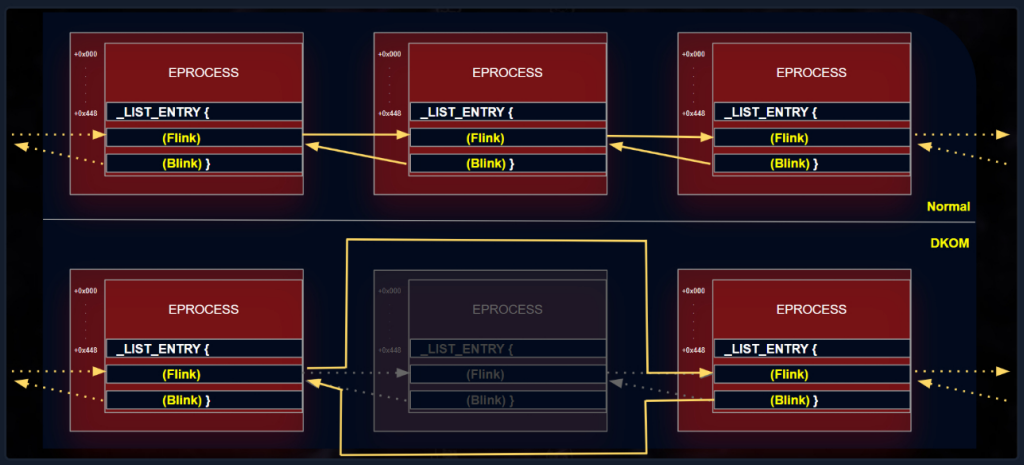
These linked lists of EPROCESS structures are used by the Windows kernel to quickly iterate through all running processes on the system. The below diagram shows how this linked list looks like.

Identifying Rootkit Signs

`Direct Kernel Object Manipulation (DKOM)` is a sophisticated technique used by rootkits and advanced malware to manipulate the Windows operating system's kernel data structures in order to hide malicious processes, drivers, files, and other artifacts from detection by security tools and utilities running in userland (i.e., in user mode).

If, for example, a monitoring tool is dependent on the `EPROCESS` structure for the enumeration of the running processes, and there's a rootkit running on the system which manipulates the `EPROCESS` structure directly in kernel memory by altering the `EPROCESS` structure or unlinking a process from lists, the monitoring tool will not be able to get the hidden process in the currently running processes list.

The below screenshot shows a graphical representation of how this unlinking actually works.



The `psscan` plugin is used to enumerate running processes. It scans the memory pool tags associated with each process's `EPROCESS` structure. This technique can help identify processes that may have been hidden or unlinked by rootkits, as well as processes that have been terminated but have not been removed from memory yet. This plugin can be used as follows.

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/rootkit.vmem psscan
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
Offset(P)          Name                PID   PPID PDB        Time created                   Time ex
------------------ ---------------- ------ ------ ---------- ------------------------------ -------
0x0000000001a404b8 ipconfig.exe       2988   2980 0x091403c0 2023-06-24 07:31:16 UTC+0000   2023-06
0x0000000001a63138 cmd.exe            2980   2004 0x091401c0 2023-06-24 07:31:16 UTC+0000   2023-06
0x0000000001b24888 explorer.exe       1444    624 0x09140320 2023-06-23 16:34:38 UTC+0000
0x0000000001bc62a8 taskschе.exe       1084   1684 0x091403e0 2023-06-24 07:28:16 UTC+0000
0x0000000001c3d2d8 @WanaDecryptor@    2248   1084 0x091403a0 2023-06-24 07:29:20 UTC+0000
0x0000000001c4e020 cmd.exe            1932   1444 0x09140380 2023-06-24 07:27:16 UTC+0000
0x0000000001c54da0 cmd.exe            2396   2264 0x091401c0 2023-06-24 07:29:30 UTC+0000   2023-06
0x0000000001c8a020 @WanaDecryptor@    2324   2284 0x09140440 2023-06-24 07:29:20 UTC+0000
0x0000000001cb7628 test.exe           1344    668 0x09140360 2023-06-24 07:28:15 UTC+0000
0x0000000002063ab8 svchost.exe        1220    668 0x09140160 2023-06-23 16:14:54 UTC+0000
0x0000000002093020 services.exe        668    624 0x09140080 2023-06-23 16:14:53 UTC+0000
0x0000000002094da0 ctfmon.exe          564    232 0x09140240 2023-06-23 16:15:09 UTC+0000
0x0000000002095020 csrss.exe           600    368 0x09140040 2023-06-23 16:14:51 UTC+0000
0x000000000209fa78 vmtoolsd.exe       2004    668 0x091402a0 2023-06-23 16:15:24 UTC+0000
0x00000000020a2a90 spoolsv.exe        1556    668 0x091401a0 2023-06-23 16:14:59 UTC+0000
0x00000000020ceb40 alg.exe            1520    668 0x091402c0 2023-06-23 16:15:26 UTC+0000
0x00000000020ff870 wmiprvse.exe        560    880 0x09140300 2023-06-23 16:15:26 UTC+0000
0x000000000216a650 taskhsvc.exe       2340   2248 0x09140340 2023-06-24 07:29:22 UTC+0000
0x0000000002172da0 winlogon.exe        624    368 0x09140060 2023-06-23 16:14:52 UTC+0000
0x00000000021adda0 msmsgs.exe          548    232 0x09140220 2023-06-23 16:15:09 UTC+0000
0x000000000224b128 svchost.exe         992    668 0x09140100 2023-06-23 16:14:53 UTC+0000
0x000000000225cda0 VGAuthService.e    1832    668 0x09140280 2023-06-23 16:15:16 UTC+0000
0x0000000002269490 vmacthlp.exe        848    668 0x091400c0 2023-06-23 16:14:53 UTC+0000
0x0000000002288770 wmic.exe           2416   2396 0x09140400 2023-06-24 07:29:30 UTC+0000   2023-06
0x00000000022ee020 cmd.exe            1628   1444 0x091402e0 2023-06-24 07:25:01 UTC+0000
0x0000000002346990 svchost.exe         880    668 0x091400e0 2023-06-23 16:14:53 UTC+0000
0x00000000023c7618 taskmgr.exe         260   1444 0x091401e0 2023-06-24 07:27:55 UTC+0000
0x0000000002419850 svchost.exe        1136    668 0x09140120 2023-06-23 16:14:53 UTC+0000
0x000000000248c020 smss.exe            368      4 0x09140020 2023-06-23 16:14:49 UTC+0000
0x000000000248f020 svchost.exe        1176    668 0x09140140 2023-06-23 16:14:53 UTC+0000
0x000000000249fda0 vmtoolsd.exe        540    232 0x09140180 2023-06-23 16:15:09 UTC+0000
0x00000000024a57a8 lsass.exe           680    624 0x091400a0 2023-06-23 16:14:53 UTC+0000
0x00000000024cb928 svchost.exe        1708    668 0x09140260 2023-06-23 16:15:16 UTC+0000
0x000000000250e020 rundll32.exe        532    232 0x09140200 2023-06-23 16:15:09 UTC+0000
0x00000000025c8830 System                4      0 0x0031c000
```

In the output below, we can see that the `pslist` plugin could not find `test.exe` which was hidden by a rootkit, but the `psscan` plugin was able to find it.

Memory Forensics

```
MisaelMacias@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/rootkit.vmem pslist
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cry
  from cryptography.hazmat.backends.openssl import backend
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          
---------- -------------------- ------ ------ ------ -------- ------ ------ ----------------------
0x823c8830 System                    4      0     58      476 ------      0                               
0x8228c020 smss.exe                368      4      3       19 ------      0 2023-06-23 16:14:49 UTC
0x81e95020 csrss.exe               600    368     14      544      0      0 2023-06-23 16:14:51 UTC
0x81f72da0 winlogon.exe            624    368     19      514      0      0 2023-06-23 16:14:52 UTC
0x81e93020 services.exe            668    624     16      277      0      0 2023-06-23 16:14:53 UTC
0x822a57a8 lsass.exe               680    624     23      358      0      0 2023-06-23 16:14:53 UTC
0x82069490 vmacthlp.exe            848    668      1       25      0      0 2023-06-23 16:14:53 UTC
0x82146990 svchost.exe             880    668     18      202      0      0 2023-06-23 16:14:53 UTC
0x8204b128 svchost.exe             992    668     11      272      0      0 2023-06-23 16:14:53 UTC
0x82219850 svchost.exe            1136    668     84     1614      0      0 2023-06-23 16:14:53 UTC
0x8228f020 svchost.exe            1176    668      5       77      0      0 2023-06-23 16:14:53 UTC
0x81e63ab8 svchost.exe            1220    668     15      218      0      0 2023-06-23 16:14:54 UTC
0x81ea2a90 spoolsv.exe            1556    668     11      129      0      0 2023-06-23 16:14:59 UTC
0x8230e020 rundll32.exe            532    232      4       78      0      0 2023-06-23 16:15:09 UTC
0x8229fda0 vmtoolsd.exe            540    232      6      247      0      0 2023-06-23 16:15:09 UTC
0x81fadda0 msmsgs.exe              548    232      2      190      0      0 2023-06-23 16:15:09 UTC
0x81e94da0 ctfmon.exe              564    232      1       75      0      0 2023-06-23 16:15:09 UTC
0x822cb928 svchost.exe            1708    668      5       87      0      0 2023-06-23 16:15:16 UTC
0x8205cda0 VGAuthService.e        1832    668      2       60      0      0 2023-06-23 16:15:16 UTC
0x81e9fa78 vmtoolsd.exe           2004    668      7      278      0      0 2023-06-23 16:15:24 UTC
0x81eff870 wmiprvse.exe            560    880     12      236      0      0 2023-06-23 16:15:26 UTC
0x81eceb40 alg.exe                1520    668      6      107      0      0 2023-06-23 16:15:26 UTC
0x81924888 explorer.exe           1444    624     17      524      0      0 2023-06-23 16:34:38 UTC
0x821c7618 taskmgr.exe             260   1444      3       75      0      0 2023-06-24 07:27:55 UTC
0x81a3d2d8 @WanaDecryptor@        2248   1084      3       57      0      0 2023-06-24 07:29:20 UTC
0x81a8a020 @WanaDecryptor@        2324   2284      2       56      0      0 2023-06-24 07:29:20 UTC
0x81f6a650 taskhsvc.exe           2340   2248      2       60      0      0 2023-06-24 07:29:22 UTC
0x81863138 cmd.exe                2980   2004      0 --------      0      0 2023-06-24 07:31:16 UTC
0x818404b8 ipconfig.exe           2988   2980      0 --------      0      0 2023-06-24 07:31:16 UTC
```

Memory Analysis Using Strings

Analyzing strings in memory dumps is a valuable technique in memory forensics and incident response. Strings often contain human-readable information, such as text messages, file paths, IP addresses, and even passwords.

We can either use the Strings tool from the Sysinternals suite if our system is Windows-based, or the `strings` command from `Binutils`, if our system is Linux-based.

Let's see some examples against a memory dump named `Win7-2515534d.vmem` that resides in the `/home/htb-student/MemoryDumps` directory of this section's target.

Identifying IPv4 Addresses

Memory Forensics

```
MisaelMacias@htb[/htb]$ strings /home/htb-student/MemoryDumps/Win7-2515534d.vmem | grep -E "\b([0-9
---SNIP---
127.192.0.0/10
212.83.154.33
directory server at 10.10.10.1:52860
127.192.0.0/10
0.0.0.0
192.168.182.254
---SNIP---
```

Identifying Email Addresses

Memory Forensics

```
MisaelMacias@htb[/htb]$ strings /home/htb-student/MemoryDumps/Win7-2515534d.vmem | grep -oE "\b[A-Z
CPS-requests@verisign.com
silver-certs@saunalahti.fi
joe@freebsd.org
info@netlock.net
UtV@UtV.UtT
acrse@economia.gob
acrse@economia.gob
```

```
CPS-requests@verisign.com
dl@comres.dll
info@globaltrust.info
info@globaltrust.info
info@globaltrust.info
ll@tzres.dll
am@tzres.dll
sy@tzres.dll
d@tzres.dll
5@tzres.dll
ic@tzres.dll
ll@tzres.dll
oo@tzres.dll
N@tzres.dll
1@tzres.dll
Mi@tzres.dll
le@tzres.dll
UtV@UtV.UtT
ssupport@hex-rays.com
ssupport@hex-rays.com
CPS-requests@verisign.com
info@aadobetech.com
CPS-requests@verisign.com
noreply@vmware.com
noreply@vmware.com
noreply@vmware.com
appro@openssl.org
appro@openssl.org
support@hex-rays.com
WanaDecryptor@.exe.lnk
1istrstream@stayer.exe
iVq0xhg@p.yRg
appro@openssl.org
appro@openssl.org
support@hex-rays.com
T@..AA
appro@openssl.org
em@provsvc.dll
support@hex-rays.com
support@hex-rays.com
support@hex-rays.com
WanaDecryptor@.exe.lnk
appro@openssl.org
now@bitwi.se
Bram@vim.org
support@hex-rays.com
support@hex-rays.com
---SNIP---
```

Identifying Command Prompt or PowerShell Artifacts

Memory Forensics

```
MisaelMacias@htb[/htb]$ strings /home/htb-student/MemoryDumps/Win7-2515534d.vmem | grep -E "(cmd|po
---SNIP---
ComSpec=C:\WINDOWS\system32\cmd.exe
ComSpec=C:\WINDOWS\system32\cmd.exe
cmd.exe
cmd.exe
cmd.exe
cmd.exe
C:\WINDOWS\system32\cmd.exe
cmd.exe /c "C:\Intel\ueqzlhmlwuxdg271\tasksche.exe"
ComSpec=C:\WINDOWS\system32\cmd.exe
cmd.exe /c "%s"
cmd.exe /c start /b @WanaDecryptor@.exe vs
cmd /c ""C:\Program Files\VMware\VMware Tools\suspend-vm-default.bat""
---SNIP---
```

These are just a few examples of common string searches during memory forensics and incident response. You can adapt and customize these searches based on your specific investigation's needs and the types of information you are looking for in the memory dump. Regular expressions can be powerful tools for pattern matching and data extraction during forensic analysis.

> **VPN Servers**

⚠ **Warning:** Each time you "Switch", your connection keys are regenerated and you must re-download your VPN connection file.

All VM instances associated with the old VPN Server will be terminated when switching to a new VPN server.

Existing PwnBox instances will automatically switch to the new VPN server.

US Academy 3 — Medium Load

PROTOCOL

UDP 1337 | TCP 443

DOWNLOAD VPN CONNECTION FILE

Connect to Pwnbox

Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK — 139ms

Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Download VPN Connection File

Target(s): Click here to spawn the target system!

SSH to with user "htb-student" and password "HTB_@cademy_stdnt!"

+ 1 Examine the file "/home/htb-student/MemoryDumps/Win7-2515534d.vmem" with Volatility. Enter the parent process name for @WanaDecryptor (Pid 1060) as your answer. Answer format: _.exe

tasksche.exe

Submit

SSH to with user "htb-student" and password "HTB_@cademy_stdnt!"

+ 1 Examine the file "/home/htb-student/MemoryDumps/Win7-2515534d.vmem" with Volatility. tasksche.exe (Pid 1792) has multiple file handles open. Enter the name of the suspicious-looking file that ends with .WNCRYT as your answer. Answer format: _.WNCRYT

hibsys.WNCRYT

Submit

SSH to with user "htb-student" and password "HTB_@cademy_stdnt!"

+ 1 Examine the file "/home/htb-student/MemoryDumps/Win7-2515534d.vmem" with Volatility. Enter the Pid of the process that loaded zlib1.dll as your answer.

3012

Submit

Previous Next

Mark Complete & Next