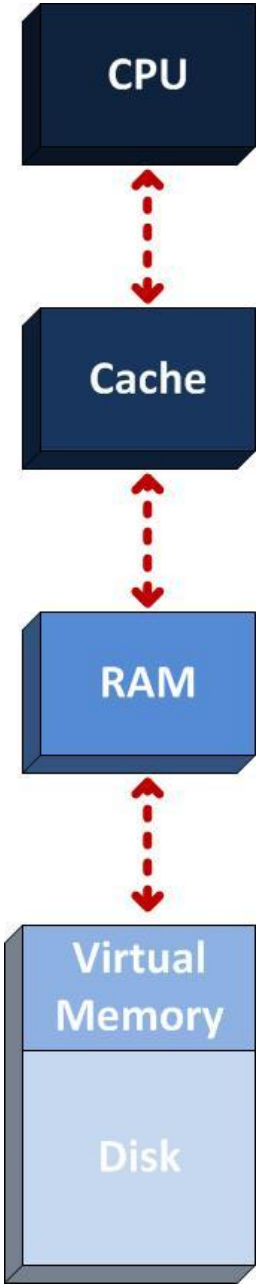
Detecting Malware With Memory Forensics

Hal Pomeranz

SANS Institute



Why Memory Forensics?

Everything in the OS traverses RAM

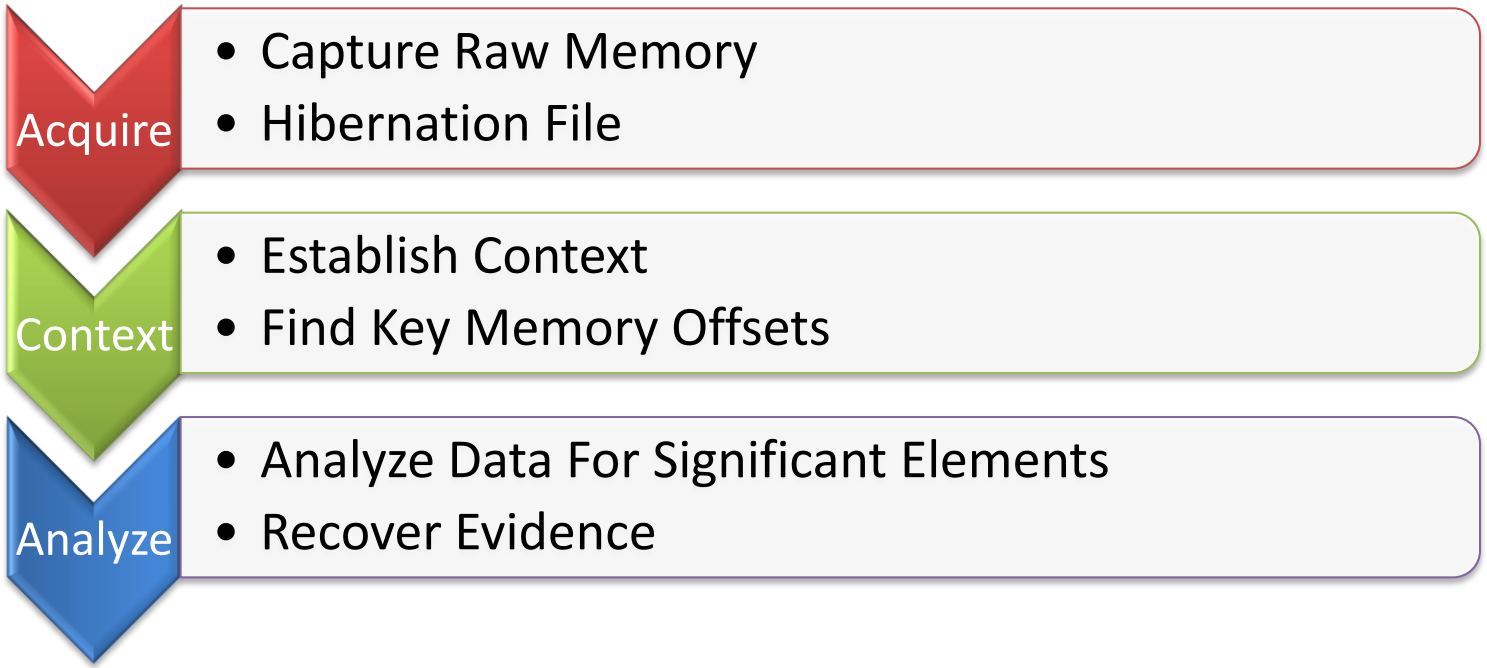
- Processes and threads
- Malware (including rootkit technologies)
- Network sockets, URLs, IP addresses
- Open files
- User generated content
  - Passwords, caches, clipboards
- Encryption keys
- Hardware and software configuration
- Windows registry keys and event logs

Memory Analysis Advantages

- Best place to identify malicious software activity
  - Study running system configuration
  - Identify inconsistencies (contradictions) in system
  - Bypass packers, binary obfuscators, rootkits (including kernel mode) and other hiding tools.
- Analyze and track recent activity on the system
  - Identify all recent activity – in context
  - Profile user or attacker activities
- Collect evidence that cannot be found anywhere else
  - Memory-only malware
  - Chat threads
  - Internet activities

What is Memory Forensics?

- Study of data captured from memory of a target system
- Ideal analysis includes physical memory data (from RAM) as well as Page File (or SWAP space) data



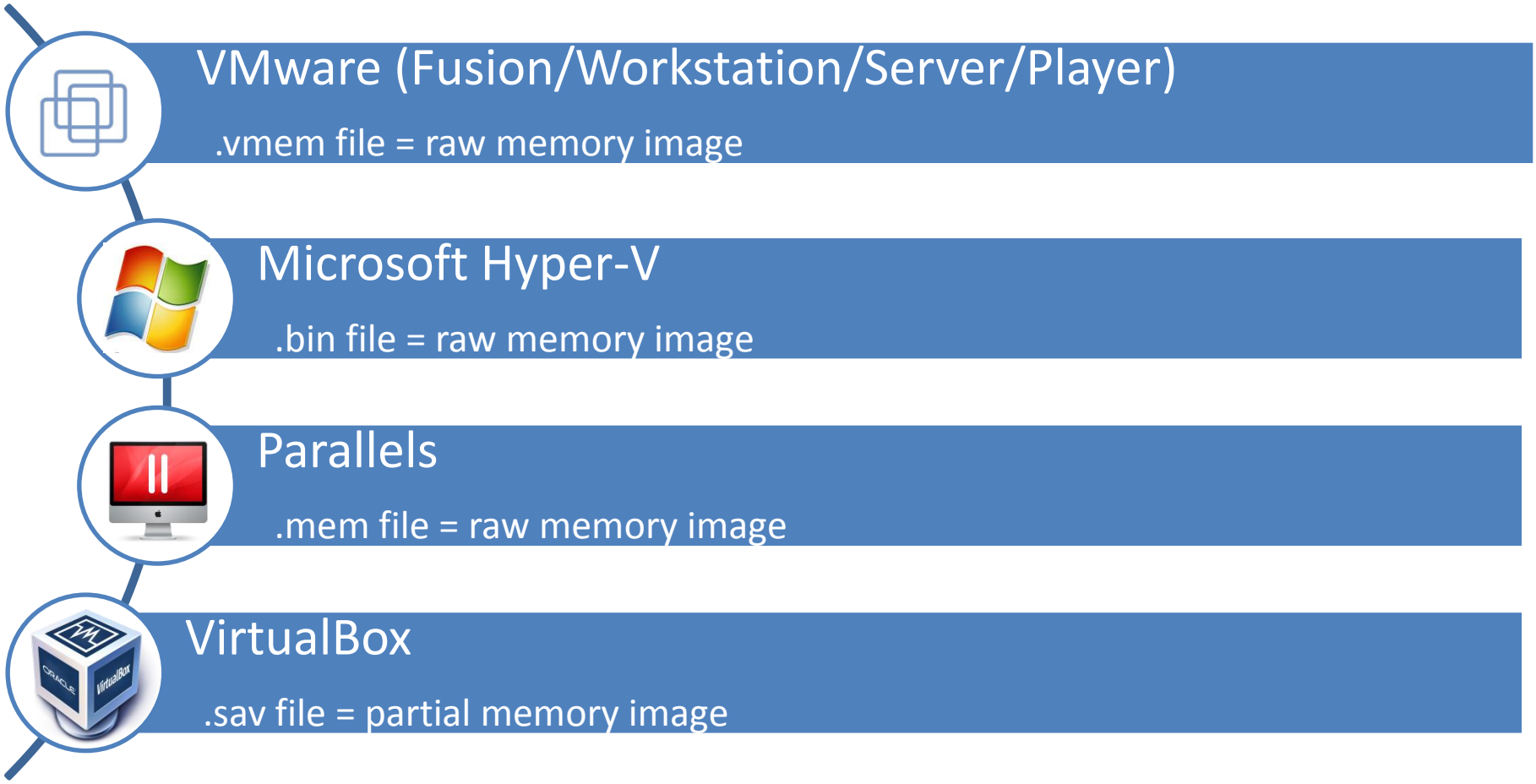
Windows Memory Acquisition

- LIVE System (RAM Acquisition)

  - **`DumpIt.exe`**
    - http://www.moonsols.com/2011/07/18/moonsols-dumpit-goes-mainstream/
  - **`win32dd.exe / win64dd.exe`**
    - Author: Matthew Suiche
    - http://www.moonsols.com/products/
  - **`Mandiant Redline`**
    - http://www.mandiant.com/products/free_software/redline/

- DEAD System
  - Hibernation File
    - Contains a compressed RAM Image
    - **`%SystemDrive%/hiberfil.sys`**

Virtual Machine Memory Acquisition



Extract Memory from Hibernation File (**hiberfil.sys**)

- **hibr2bin** can acquire physical memory (RAM) from a Windows hibernation file (XP and VISTA only)
  - Pro Version Compatible with XP-Win7/2008 (32 and 64 bit)

hibr2bin.exe <input file> <output file>

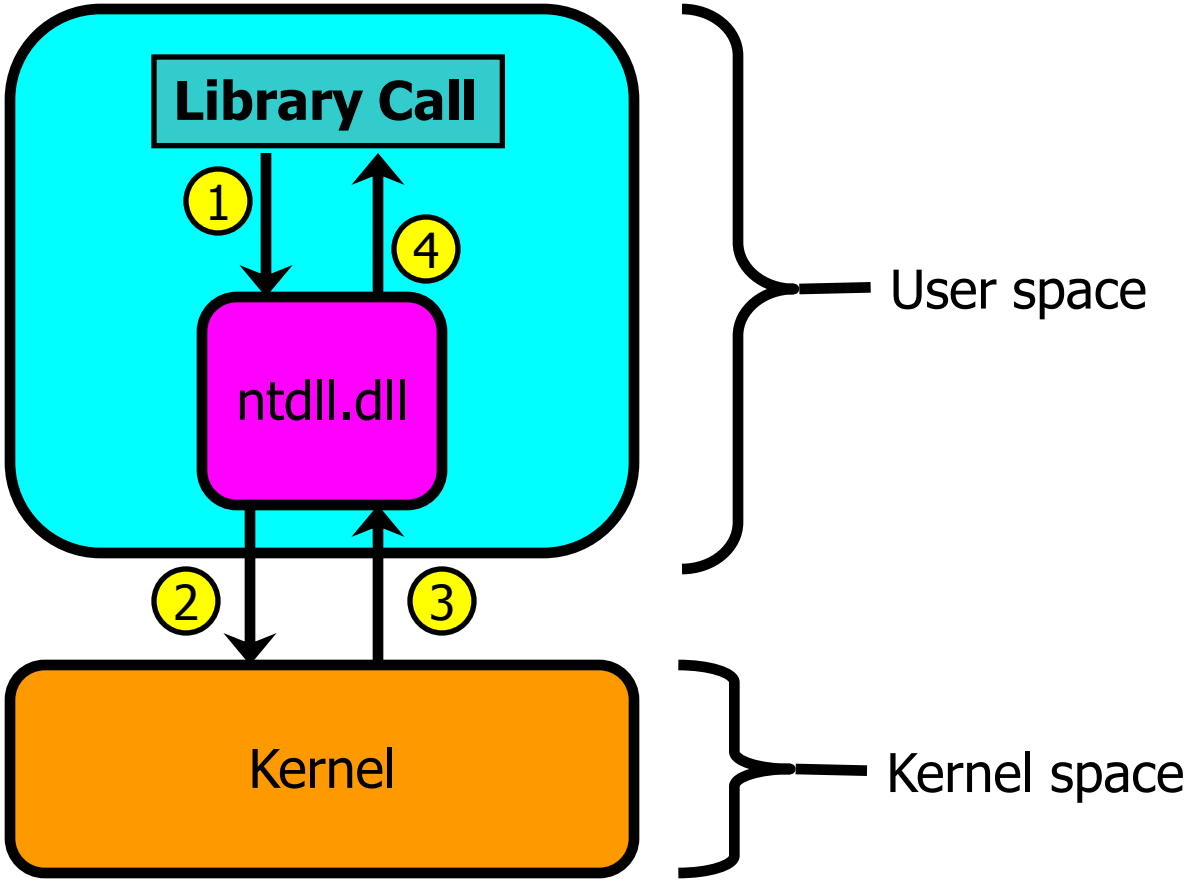
- Location on COURSE DVD:
  D:\windows forensic tools\memory imaging

- Example: Extract hibernation file memory and save to a USB DRIVE
  D:\> hibr2bin D:\hiberfil.sys E:\hibernation_memory.img

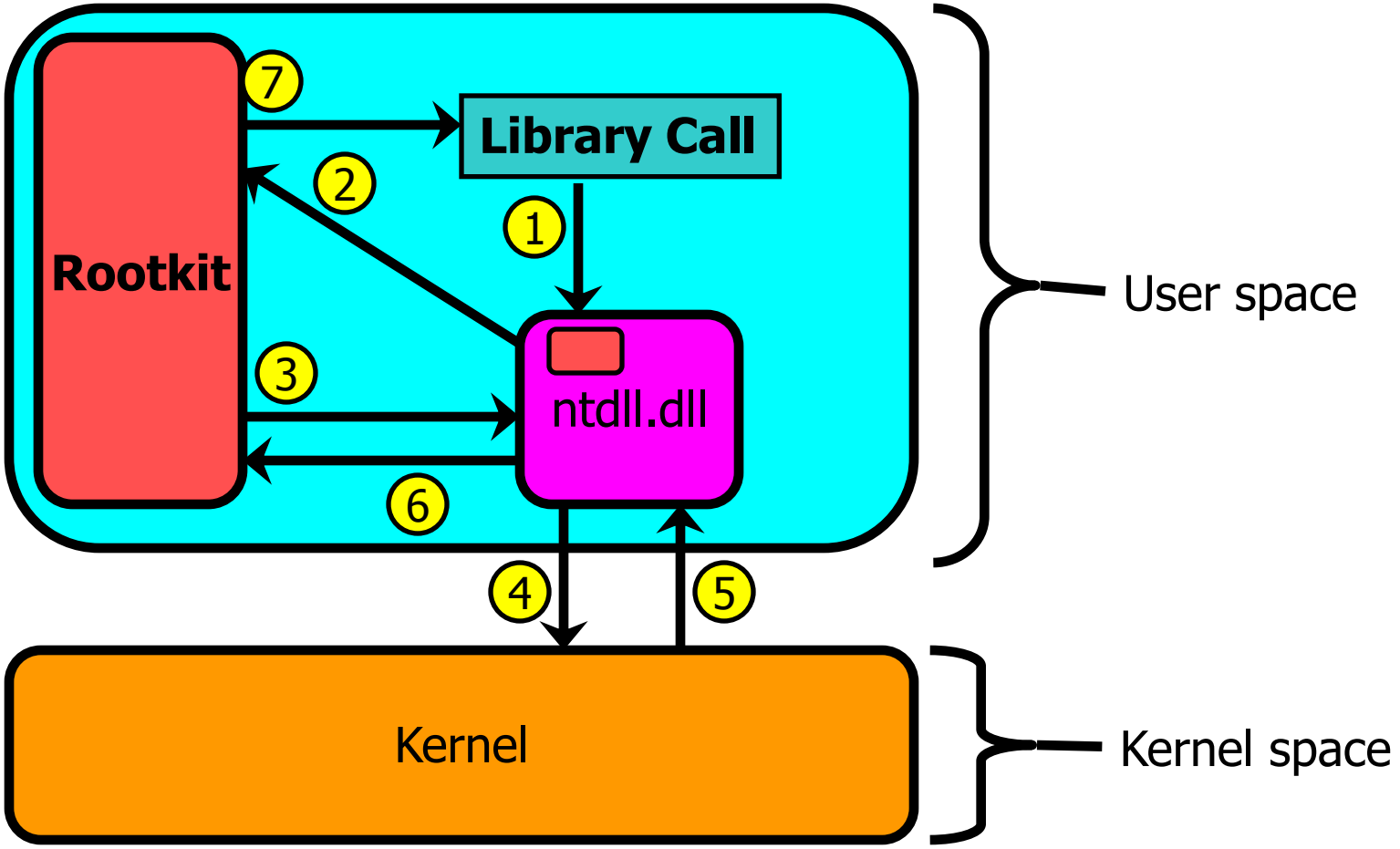
**** Volatility can also convert hibernation files ****

DLL Injection

Normal DLL Interaction



DLL Injection



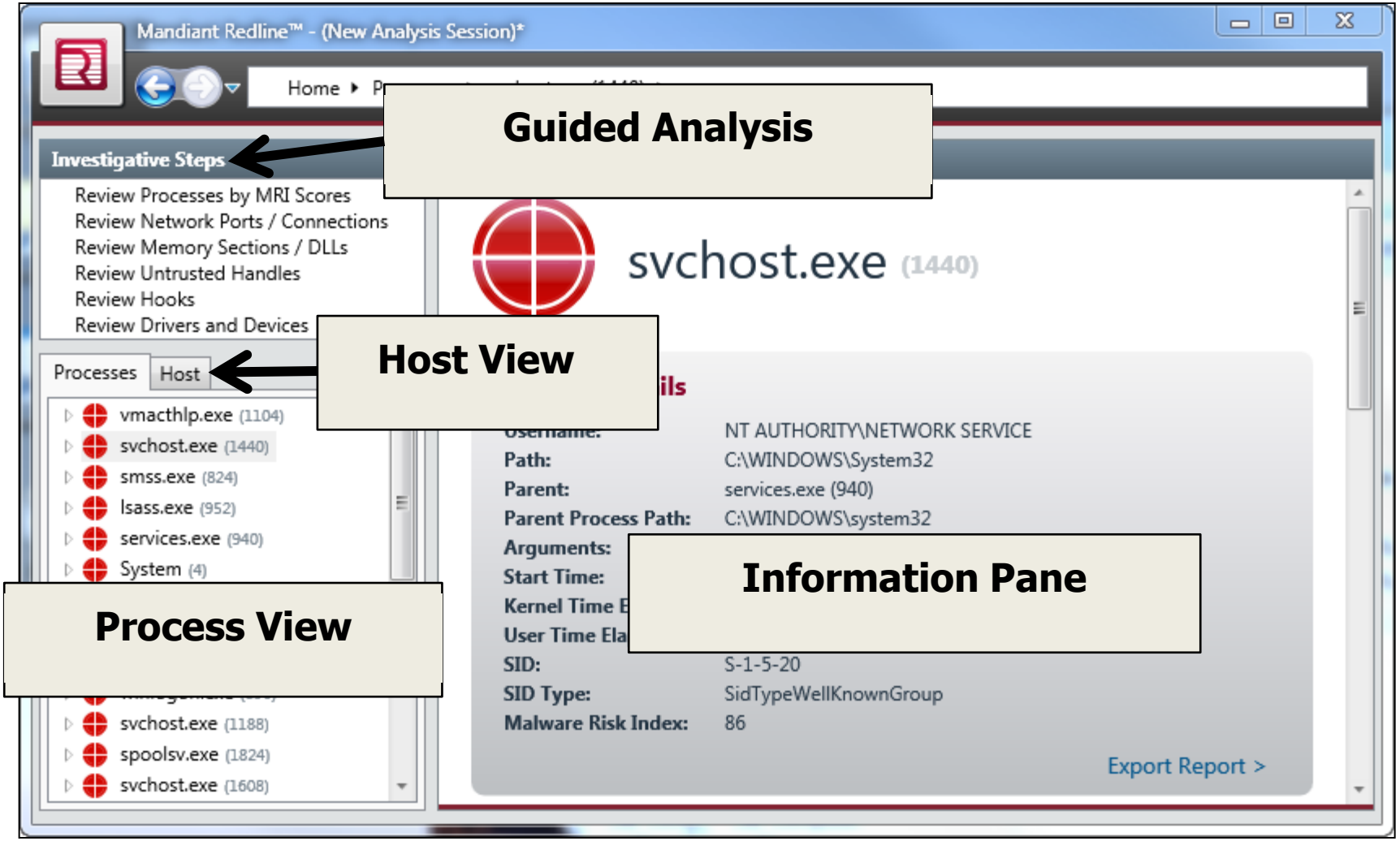
Detecting Injection

- DLL injection is **very** common with modern malware
  - VirtualAllocEx( ) and CreateRemoteThread( )
  - SetWindowsHookEx( )
- Process hollowing is another injection technique
  - Malware starts a new instance of legitimate process
  - Original process code de-allocated and replaced
  - Retains DLLs, handles, data, etc. from original process
- **Code injection is relatively easy to detect**
  - Review memory sections marked as **`Page_Execute_ReadWrite`** and having no memory-mapped file present
    - Scan for DLLs (PE files) and shellcode
  - Process image not backed with file on disk = process hollowing

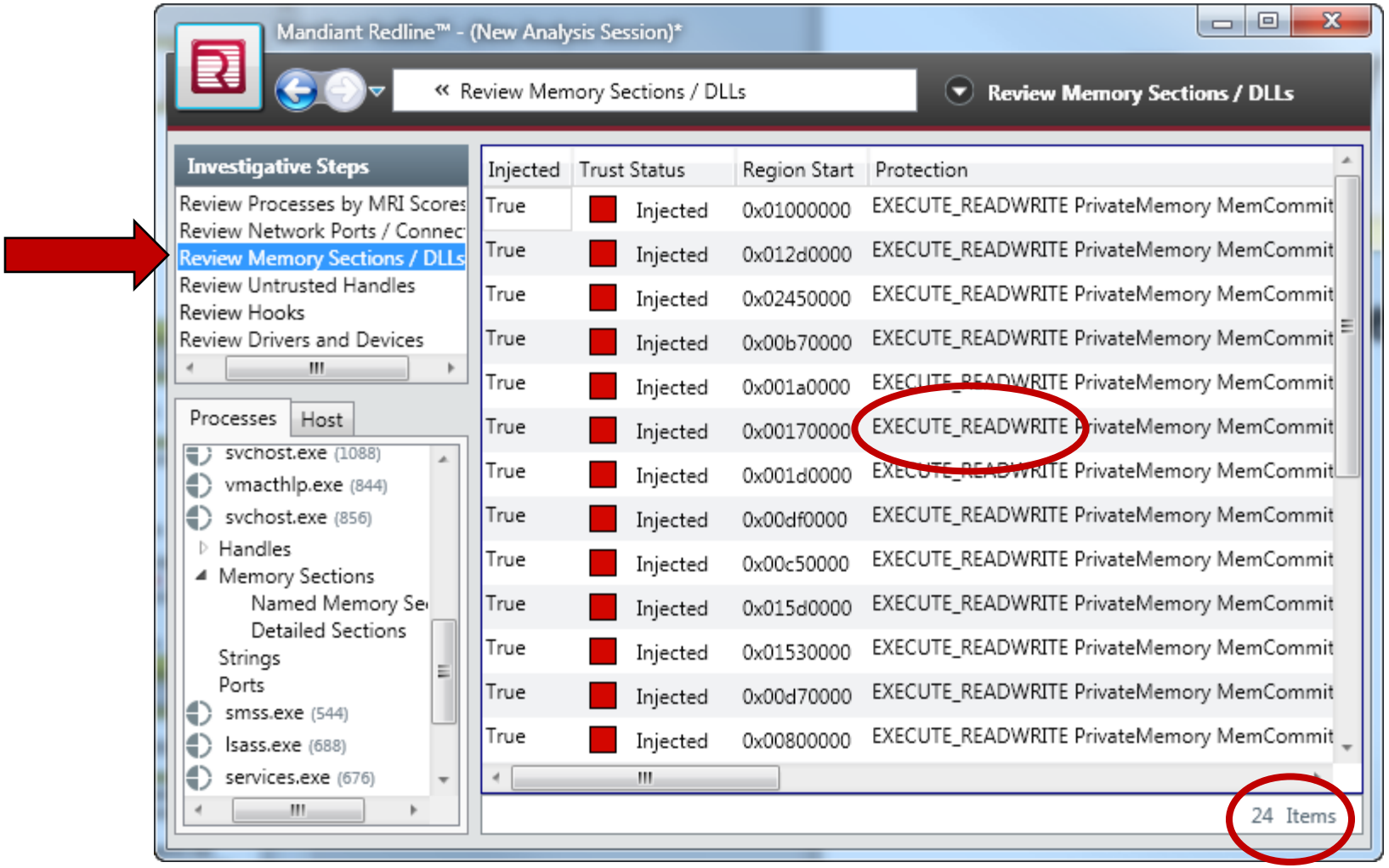
Zeus / Zbot Overview

- Persistent malware designed to steal credentials
- Many variants. A popular one does the following:
  - Copies itself to %system32%\sdra64.exe
  - **Injects code into winlogon.exe or explorer.exe**
    - Further injects code into every process but csrss & smss
  - Auto-start path: HKLM\Software\Microsoft\Windows NT\winlogon\userinit
  - Creates local.ds & user.ds in %sytem32%\lowsec\
  - Retrieves files from command and control server
  - Mutant: _AVIRA_
  - Hooks over 50 system APIs

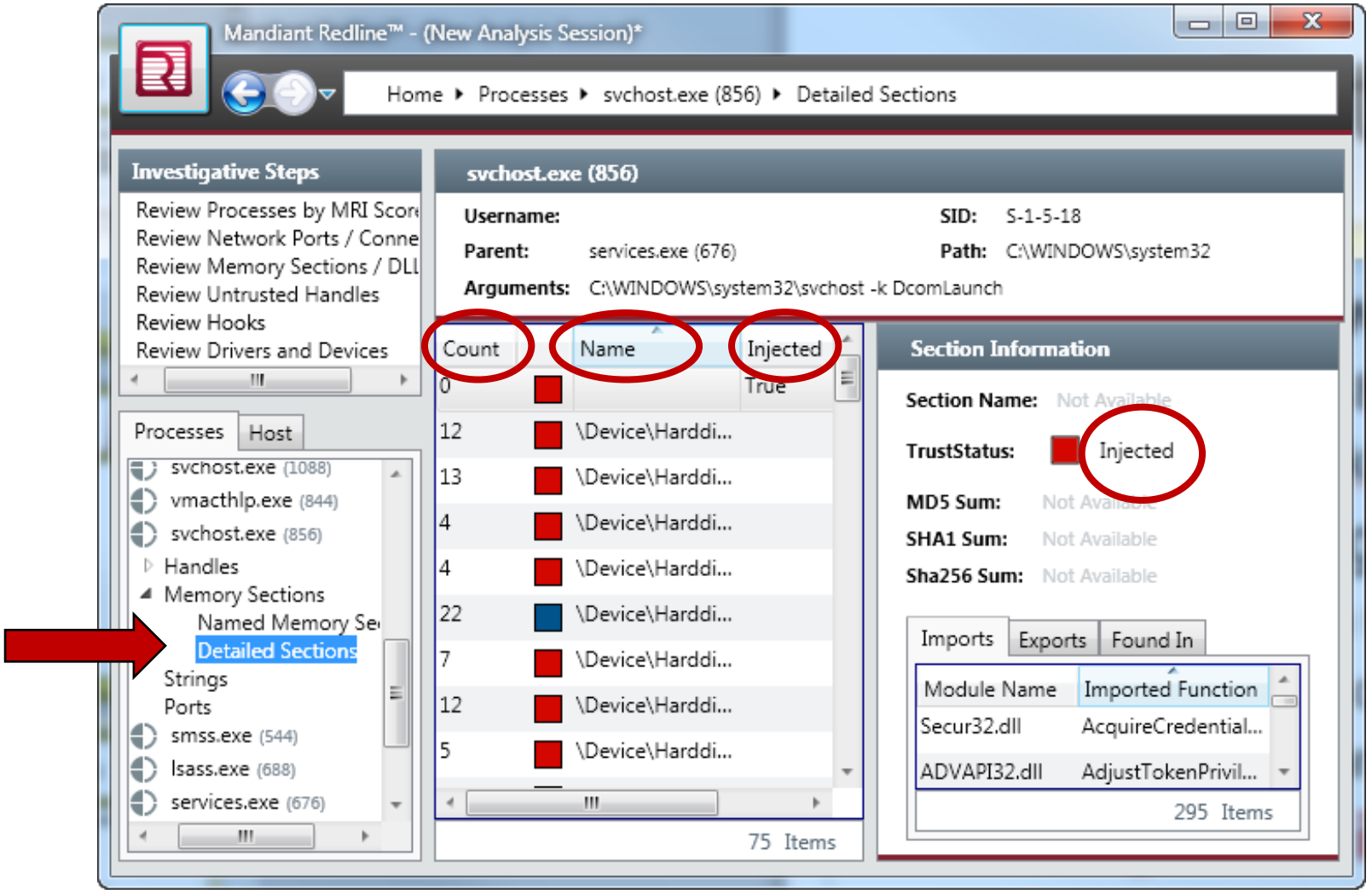
Using Mandiant Redline



Detecting Code Injection: Zeus/Zbot DLL Injection



Detecting Code Injection: Finding Injected Sections



Volatility

- Command-line memory forensic tool
- Primarily Windows-focused
- Linux (Android) & Mac support now available
- Modular, portable

Help!

- The –h flag gives configuration information in Volatility
  - Used alone it identifies the version, currently loaded plugins, and common parameters
- Use –h with a plugin to get details and plugin-specific usage

```
root@SIFT-Workstation:/# vol.py malfind -h

  -D DUMP_DIR, --dump-dir=DUMP_DIR
                        Directory in which to dump executable files
  -Y YARA_RULES, --yara-rules=YARA_RULES
                        Use YARA rules in addition to finding injected code
  -K, --kernel          Scan kernel modules
----------------------------------
Module Malfind
----------------------------------
[MALWARE] Find hidden and injected code
```

Code Injection
`ldrmodules`

Purpose

- DLLs are tracked in three different linked lists for each process. Stealthy malware can unlink loaded DLLs from these lists. This plugin queries each list and displays the results for comparison.

Important Parameters

- Verbose -- show full paths from each of the three DLL lists (-v)
- Show information for specific process IDs (-p)

Investigative Notes

- Most loaded DLLs will be in all 3 lists, having a “1” in each column.
- Legitimate entries may be missing in some of the lists
  - e.g. the process executable will not be present in the “InInit” list
- If an entry has no “MappedPath” information it is indicative of an injected DLL not available on disk (usually bad)

Rootkit Detection

`apihooks`

Purpose

- Detect inline and Import Address Table function hooks used by rootkits to modify and control information returned

Important Parameters

- Operate only on these process IDs (-p *PID*)
- Scan kernel modules instead of user-mode objects (-k)

Investigative Notes

- A large number of legitimate hooks can exist, weeding them out takes practice and an eye for looking for anomalies
- This plug-in can take a long time to run due to the sheer number of locations it must query – be patient!

Analyzing Process Objects: `malfind`

Purpose

- Scans process memory sections looking for indications of code injection. Identified sections are extracted for further analysis.

Important Parameters

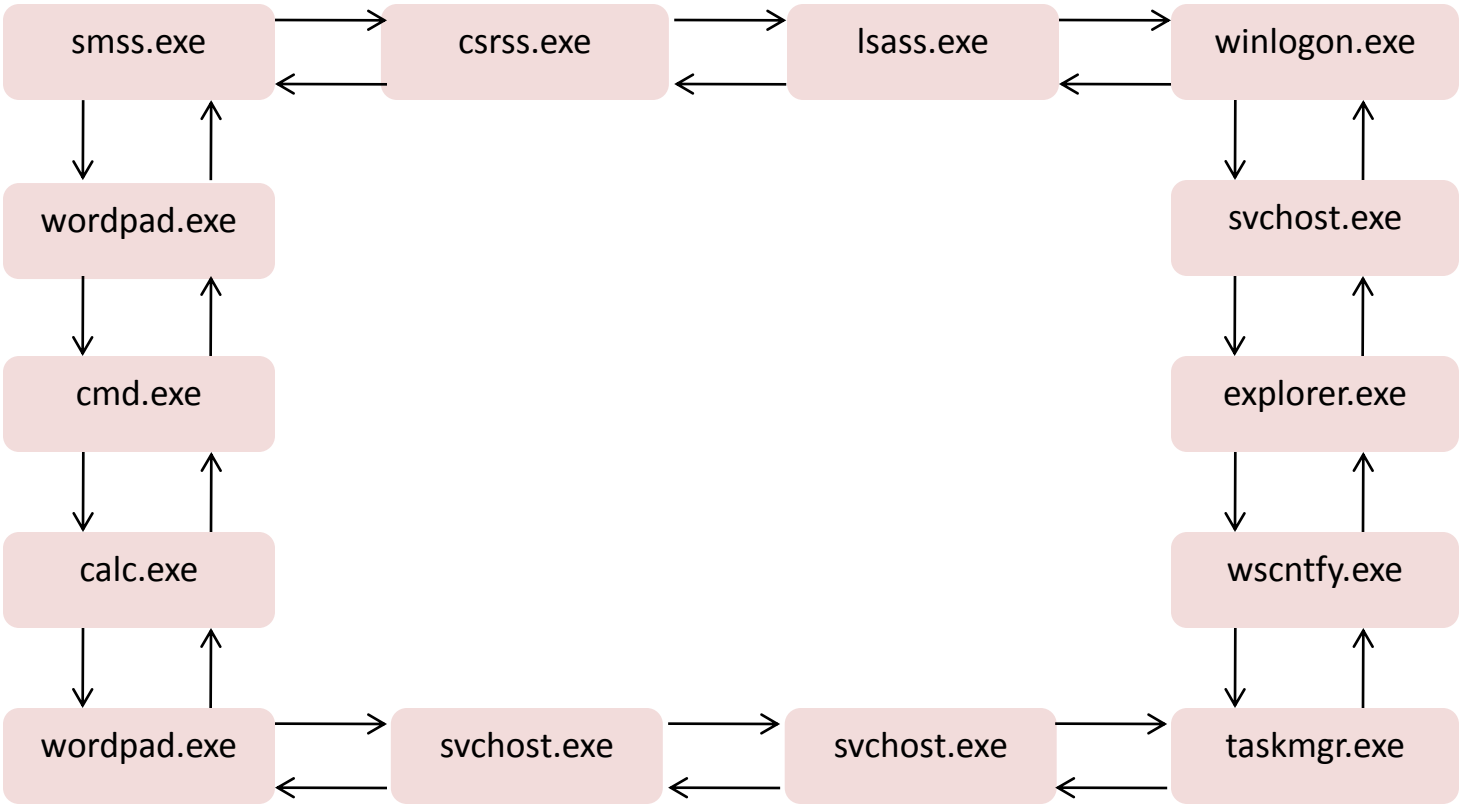
- Directory to save extracted files (--dump-dir=*directory*)
- Show information for specific process IDs (-p *PID*)
- Use **`psscan`** to find processes = more rigorous (-s)
- Search using YARA rules (-y *YARA rules file*)
- Scan kernel modules/drivers using Yara Rules (-K)

Investigative Notes

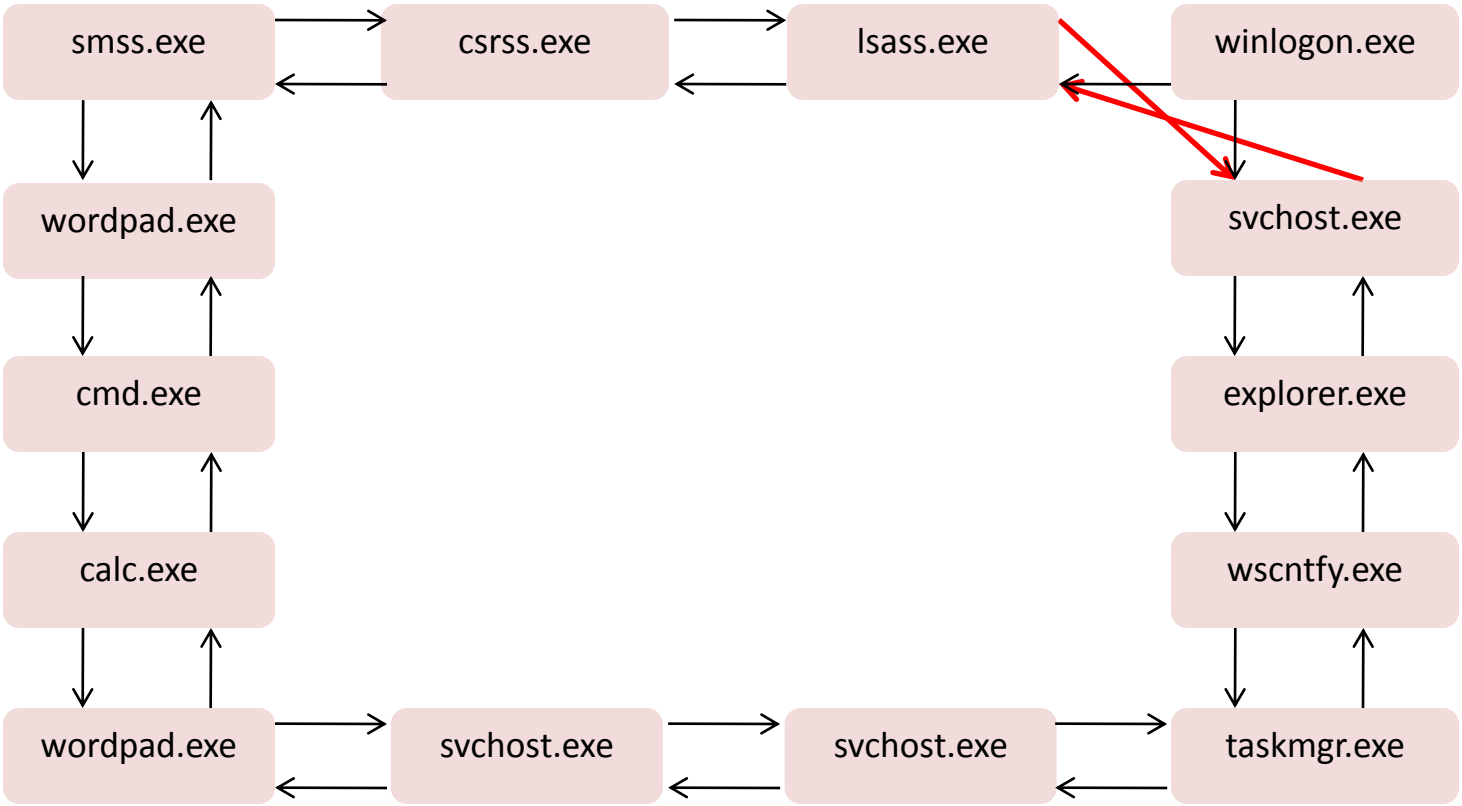
- While **`malfind`** has an impressive hit rate, false positives do occur
  - Disassembled code provided can be helpful as a sanity check
- You may see multiple injected sections within the same process
- Dumped sections can be reverse engineered or sent to A/V

Process Hiding

EPROCESS Linked List



Hiding a Process



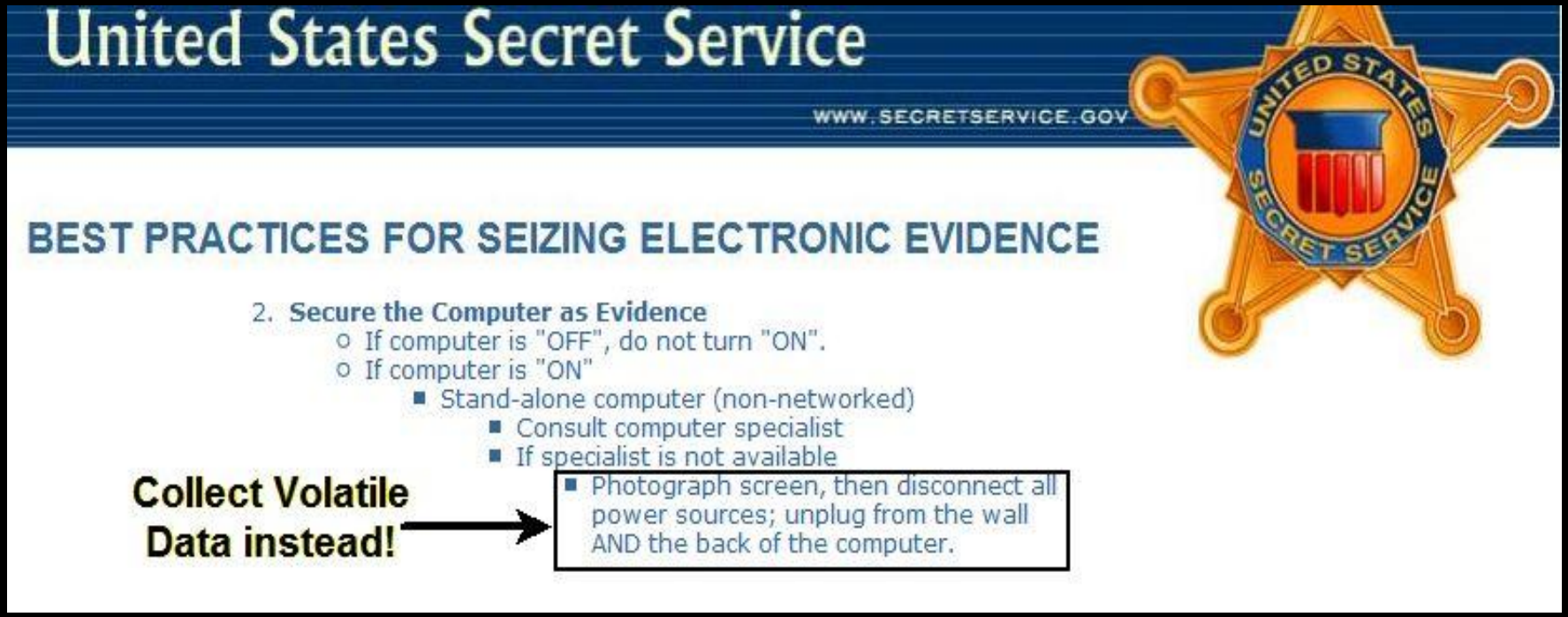
Rootkit Detection
psxview (FU Rootkit)

```
root@SIFT-Workstation:/memory# vol.py -f rootkit.img psxview
Volatile Systems Volatility Framework 2.1_alpha
Offset     Name                 Pid      pslist   psscan   thrdproc
0x81666a70L winlogon.exe        896      1        1        1
0x819bc590L alg.exe             1924     1        1        1
```

Name	Pid	pslist	psscan	thrdproc
svchost.exe	1608	0	1	1

```
0x8169bda0L svchost.exe         1188     1        1        1
0x815eb270L svchost.exe         1320     1        1        1
0x81ab1a20L services.exe        940      1        1        1
0x81617600L explorer.exe        1288     1        1        1
0x81655798L vmtoolsd.exe        308      1        1        1
0x81a385a0L smss.exe            824      1        1        1
0x819887f0L spoolsv.exe         1824     1        1        1
0x81651da0L VMUpgradeHelper     580      1        1        1
0x819922c0L svchost.exe         1608     0        1        1
0x8169d700L VMwareTray.exe      1228     1        1        1
0x815ed020L VMwareUser.exe      1484     1        1        1
0x81a55d78L vmacthlp.exe        1104     1        1        1
```

Stop Pulling the Plug



Wrapping Up

- Any final questions?
- Thanks for listening!

Hal Pomeranz SANS Institute

hal@sans.org Twitter: @hal_pomeranz

http://computer-forensics.sans.org/blog/author/halpomeranz/

http://www.sans.org/security-training/instructors/Hal-Pomeranz

http://www.deer-run.com/~hal/

SANS Digital Forensics and Incident Response

CURRICULUM

Website:
http://computer-forensics.sans.org

Blog

Blogs:
http://computer-forensics.sans.org/blog

SIFT Workstation:
http://computer-forensics.sans.org/community/downloads

Challenges

Digital Forensics Challenge:
http://computer-forensics.sans.org/challenges

Twitter:
www.twitter.com/sansforensics

FOR408
Computer Forensic Investigations – Windows In-Depth
GCFE

FOR508
Advanced Computer Forensic Analysis & Incident Response
GCFA

FOR558
Network Forensics

FOR563
Mobile Device Forensics

LEARN R.E.M

FOR610
REM: Malware Analysis Tools & Techniques
GREM

Additional Forensics Course

FOR526
Windows Memory Forensics In-Depth