Forensics for System Administrators

Basic Memory Analysis - First Steps

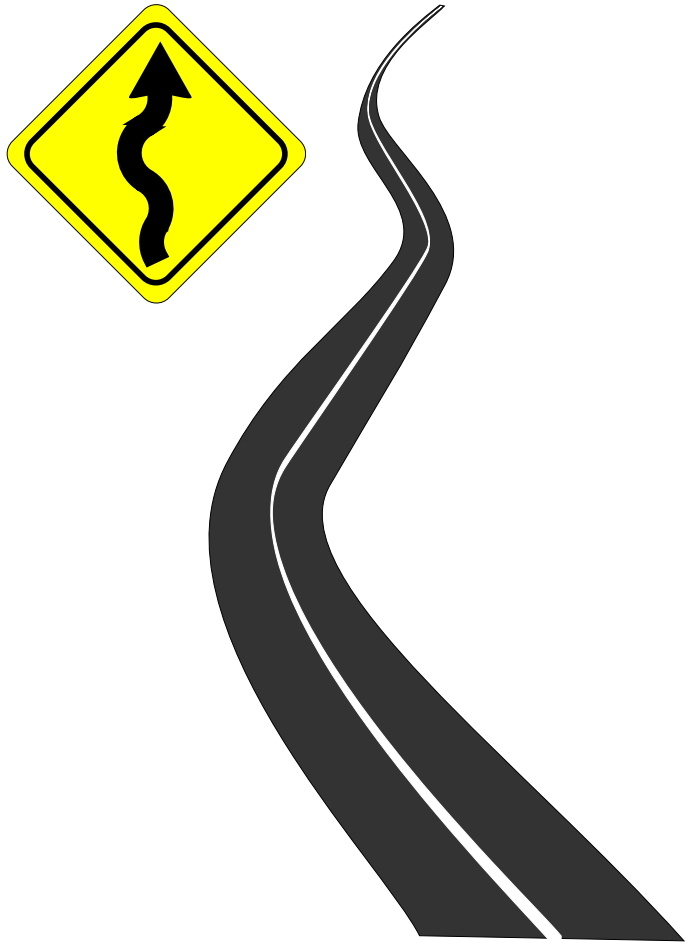
Klaus Möller
WP8-T1

Webinar, 4th of May 2022

Public

www.geant.org

Agenda - Basic Analysis

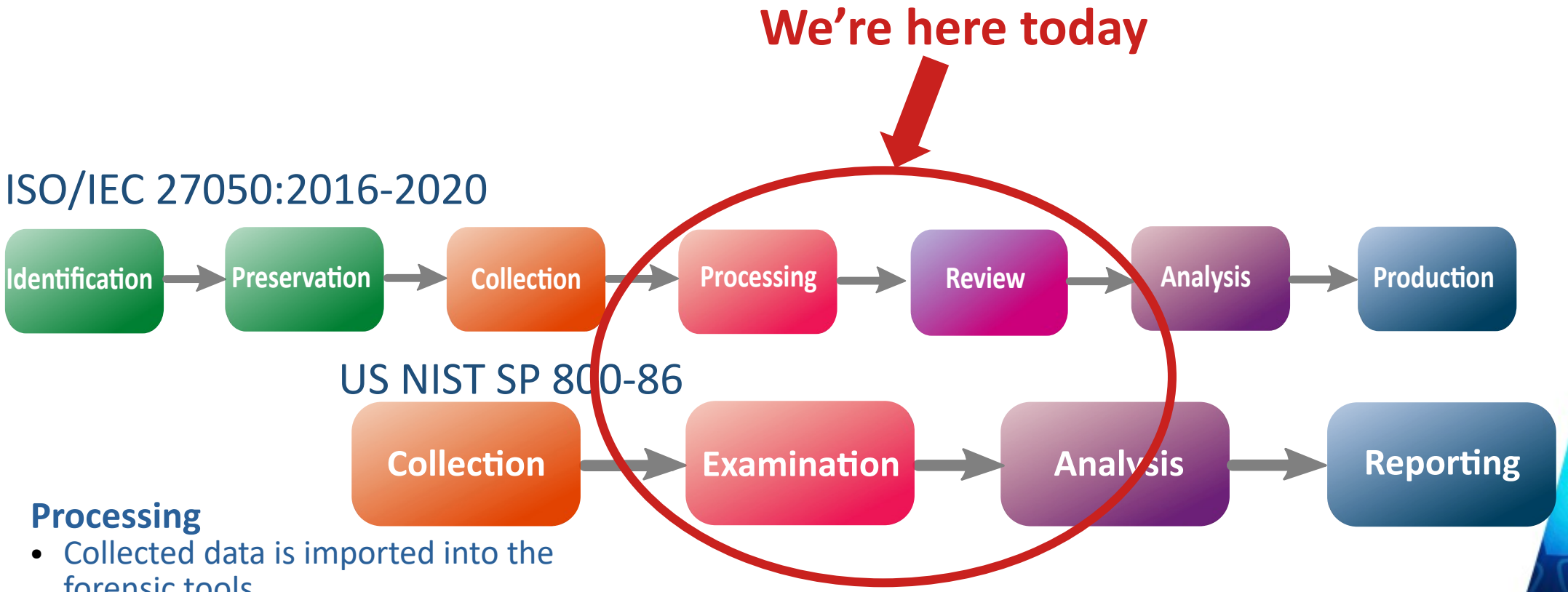


- Introduction
  - Forensic process
  - Importing profiles
- Process Analysis
  - Process listing
  - Command line arguments
- Network Analysis
  - (Open) connections
  - (Open) sockets
- Filesystem Analysis
  - Open files (objects)
  - Temporary (memory mapped) filesystems
- Other Artefacts
  - Registry (Windows)
  - Bash History (Unix/Linux)

Introduction

www.geant.org

Forensic Workflow (Recap)



Processing

- Collected data is imported into the forensic tools
- Filtering out unneeded data/information
- Normalisation of different data formats
- Building of a (super) Timeline

Review/Examination

- Assessment of the collected data
- Search for *Indicators Of Compromise* (IOC)s

Volatility Profiles/Symbol Files (Recap)

- Without additional information, ...
  - We would have no idea what kind of data is at a given address
    - Integer, float, string, structure, …
  - Or what it is used for
    - Process, socket, file, directory, etc.
- What's needed is the symbol table from the compiler
  - Can be used directly for debuggers
- Some forensic tools build more abstract, condensed structures from it
  - Volatility 2 term: Profile
  - Volatility 3 term: Intermediate Symbol File (ISF)
- Before analysis, profile/ISF has to be imported into Volatility 2/3

Volatility 2 Profiles

- Generation of a profile

```
> git clone https://github.com/volatilityfoundation/volatility.git
> cd volatility/tools/linux/
> make
> zip newprofile.zip module.dwarf /boot/System.map-$(uname -r)
```

- Import

```
> cp newprofile.zip volatility/volatility/plugins/overlays/linux
```

- Alternatively

Volatility 3 ISF

- Intermediate Symbol File (ISF) generation

```
> git clone https://github.com/volatilityfoundation/dwarf2json.git
> cd dwarf2json
> go build
> dwarf2json linux --system-map /boot/System.map -$(uname -r) \
  $(uname -r).json
```

- Import
  - `(linux|mac|windows).zip` downloaded from volatility git repo

```
> mv *.zip volatility3/volatility3/symbols/
```

  - Move newly generated ISF into appropriate sub-directory

```
> mkdir volatility3/volatility3/symbols/linux
> xz newprofile.json                  # optional, saves disk space
> mv newprofile.json volatility3/volatility3/symbols/linux
```

  - Or put the ISF into the ZIP archive

Volatility 2 vs. Volatility 3

- Volatility commands
  - Volatility 2: **`xxx (Windows), linux_xxx, mac_xxx`**
  - Volatility 3: **`windows.xxx, linux.xxx, mac.xxx`**
  - Be careful with the position of arguments and options!
- Development goes into Volatility 3
  - Python 3 codebase (Volatility 2 is based on Python 2)
  - However, still beta code (since January 2020)
- Volatility 2 is not obsolete!
  - It's not getting feature updates any more, but sometimes bugfixes
  - Code is still working fine
  - Lots of plug-ins and functionality that has not been ported to Volatility 3
  - Very old Windows versions (XP, Server 2003) are better supported

Process Analysis

www.geant.org

Process Listings

- Listing (pslist, pstree)
  - Walk the process table structures of the OS kernel (EPROCESS, task_struct)
  - Can be fooled by direct manipulation of kernel data structures
- Scan (psscan)
  - Scan (carve) the memory for process control blocks (PCBs)
  - Cannot easily be fooled by adversaries
  - Finds dead processes too
    - Like in filesystems, process table entries are only freed, not overwritten
- Cross-View (psxview) (Volatility 2, Windows only)
  - Walk different link structures in the process table
  - Differences may be hint of adversary manipulation
  - Investigator needs background knowledge to avoid false positives

Psxview Columns

- pslist: Find processes by walking from `PsActiveProcessHead`
  - Dead processes are listed with *False*
- psscan: Find processes by scanning for `_EPROCESS` blocks
- thrdproc: Find processes by following links from `_THREAD` blocks
  - Very suspicious, when pslist and/or psscan column show *False*
- pspcdid: Scan of the Cid table for process- and thread-IDs
- csrss: Windows processes, i. e. childs of `csrss.exe`
  - Column entry is *False* for System, smss.exe, and csrss.exe

Command line

- Command lines may hint at malicious process
  - Such as **`/tmp/vi`**
  - Or or **`notepad -d 1000 -i 203.0.113.7`**
  - **`cmdline`**, **`linux_psaux`**, **`mac_psaux`** (Volatility 2)
  - **`mac.psaux`** (Volatility 3)
- Environment variables may affect program behaviour too
  - Think of "." in **`PATH`**
  - Or **`LD_PRELOAD`**, **`LD_LIBRARY_PATH`**
  - envars, linux_psenv, mac_psenv (Volatility 2)
  - windows.envars (Volatility 3)

Network Analysis

www.geant.org

Network Connections

- Active network connection can be a good source of forensic information
  - Esp. connections to C&C servers, etc.
  - Volatility 2: **`connections (XP), netscan (Vista), linux_netscan, mac_netstat`**
    - On Windows XP, IPv4 connections only
  - Volatility 3: **`windows.netstat, mac.netstat`**
- Of course, closed connections can be just as good
  - Carving for socket data structures (c. f. `psscan`)
  - However, some information may already have been overwritten
  - Volatility 2: **`linux_netscan`**
  - Volatility 3: **`windows.netscan`**

Open Sockets

Live Demo

- Open (listening) sockets can be a sign of a backdoor
  - Volatility 2: **`sockets, sockscan, linux_netstat, mac_dead_sockets`**
  - Volatility 3: **`windows.netstat, mac.netstat`**

```
> ./vol.py -f /mnt/hgfs/Images/zeus.vmem sockets
Volatility Foundation Volatility Framework 2.6.1
Offset(V)       PID   Port  Proto Protocol        Address         Create Time
---------- -------- ------ ------ -------------- --------------- -----------
0x80fd1008        4      0     47 GRE            0.0.0.0         2010-08-11 06:08:00 UTC+0000
0xff258008      688    500     17 UDP            0.0.0.0         2010-08-11 06:06:35 UTC+0000
0xff367008        4    445      6 TCP            0.0.0.0         2010-08-11 06:06:17 UTC+0000
0x80ffc128      936    135      6 TCP            0.0.0.0         2010-08-11 06:06:24 UTC+0000
0xff37cd28     1028   1058      6 TCP            0.0.0.0         2010-08-15 19:17:56 UTC+0000
0xff20c478      856  29220      6 TCP            0.0.0.0         2010-08-15 19:17:27 UTC+0000
0xff225b70      688      0    255 Reserved       0.0.0.0         2010-08-11 06:06:35 UTC+0000
0xff254008     1028    123     17 UDP            127.0.0.1       2010-08-15 19:17:56 UTC+0000
…
```

GÉANT
Networks • Services • People

Filesystem Analysis

www.geant.org

Open Files

- Knowledge of what files are open by what process can uncover hints of malicious behaviour
  - Volatility 2: **`handles, linux_lsof, mac_lsof`**
  - Volatility 3: **`windows.handles, linux.lsof, mac.lsof`**
- Windows `handles` command covers more than files
  - Semaphores, Registry keys, etc.
  - Use **`--object-type`** to narrow down the output (Volatility 2 only)
  - Pipe through **`grep`** when using Volatility 3
- Likewise, `linux_lsof` also finds sockets and pipes
- And `linux_netstat` finds Unix sockets

Temporary Filesystems

- Linux tmpfs contents are kept in memory
  - Lost on reboot or power-off
  - Adversaries often use these filesystems as storage
- But they can be recovered from (linux) memory dumps
  - Volatility 2 only: **`linux_tmpfs`**
  1. Get the list of (all) tmpfs superblocks

```
> ./vol.py -f linux.img --profile=Linux64 linux_tmpfs -L
Volatility Foundation Volatility Framework 2.6.1
1 -> /sys/fs/cgroup
2 -> /run
3 -> /run/user/0
4 -> /dev/shm
```

  2. Dump the (full) tmpfs filesystem to a local directory

```
> ./vol.py -f linux.img --profile=Linux64 linux_tmpfs -S 4 -D /tmp/dumptmpfs
```

Other Artifacts

Windows Registry

- Hives are loaded from backing files into memory
- Keys are always read from memory, never from files directly
- Changed keys are marked “volatile” and saved back to files
- Adversaries can change keys in memory without marking them as volatile
  - Thus, changed keys are not written back to files
  - Persistent storage analysis can’t find anything
  - Keys changed this way are lost on reboot, of course
- Memory analysis can find the hives in main memory

Windows Registry Hives

- Show the loaded hives: **`hivelist`**
  - Volatility 3: **`windows.registry.hivelist`**
- Also: **`hivescan`** for carving

Live Demo

```
> ./vol.py -f /mnt/hgfs/Images/zeus.vmem hivelist
Volatility Foundation Volatility Framework 2.6.1
Virtual    Physical   Name
---------- ---------- ----
0xe1c49008 0x036dc008 \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\
Application Data\Microsoft\Windows\UsrClass.dat
0xe1c41b60 0x04010b60 \Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT
0xe1a39638 0x021eb638 \Device\HarddiskVolume1\Documents and Settings\NetworkService\Local Settings\
Application Data\Microsoft\Windows\UsrClass.dat
0xe1a33008 0x01f98008 \Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DAT
0xe153ab60 0x06b7db60 \Device\HarddiskVolume1\WINDOWS\system32\config\software
0xe1542008 0x06c48008 \Device\HarddiskVolume1\WINDOWS\system32\config\default
0xe1537b60 0x06ae4b60 \SystemRoot\System32\Config\SECURITY
0xe1544008 0x06c4b008 \Device\HarddiskVolume1\WINDOWS\system32\config\SAM
…
```

Windows Registry Keys

- Dump the complete Registry to disk: **dumpregistry**
- Print a key: **printkey (windows.registry.printkey)**


```
>  ./vol.py -f laqma.vmem printkey  -K "microsoft\windows\currentversion\run"
Volatility Foundation Volatility Framework 2.6.1
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\software
Key name: Run (S)
Last updated: 2010-08-15 19:09:19 UTC+0000

Subkeys:

Values:
REG_SZ        VMware Tools    : (S) "C:\Program Files\VMware\VMware Tools\VMwareTray.exe"
REG_SZ        VMware User Process : (S) "C:\Program Files\VMware\VMware Tools\VMwareUser.exe"
REG_SZ        lanmanwrk.exe   : (S) C:\WINDOWS\System32\lanmanwrk.exe
```

Windows Registry - More commands

- Volatility 2
  - **`getservicesids`** - Get the names of services in the Registry and return Calculated SID
  - **`lsadump`** - Dump (decrypted) LSA secrets from the registry
  - **`shimcache`** - Parses the Application Compatibility Shim Cache registry key
  - **`shutdowntime`** - Print ShutdownTime of machine from registry
  - **`userassist`** - Print userassist registry keys and information
- Volatility 3
  - **`windows.registry.certificates.Certificates`** - Lists the certificates in the registry's Certificate
  - **`windows.registry.userassist.UserAssist`** - Print userassist registry keys and information

Bash History

- Useful source of information in security incidents
  - List of all commands and arguments entered
  - No timestamps (not by default)
- Can be suppressed by adversaries
- However, there is always a history in the bash process memory
  - **With** timestamps
  - **`unset HISTFILE`** - History is still in memory
  - **`set +o history`** - no history
- Volatility 2: **`linux_bash`**, **`mac_bash`**
  - Also: **`linux_bash_env`**, **`mac_bash_env`** - environment variables
  - Also: **`linux_bash_hash`**, **`mac_bash_hash`** - internal command hash table
- Volatility 3: **`linux.bash`**, **`mac.bash`**

Bash History Example

Live Demo

```
> ./vol.py  -f os151-h3.lime --profile=Linuxopensuse_15_1-4_12_14-lp151_28_67-defaultx64
linux_bash -p 70948
Volatility Foundation Volatility Framework 2.6.1
Pid      Name                 Command Time                   Command
-------- -------------------- ------------------------------ -------
   70948 bash                 2022-04-29 16:15:27 UTC+0000   ps
   70948 bash                 2022-04-29 16:15:28 UTC+0000   list
   70948 bash                 2022-04-29 16:15:29 UTC+0000   ls
   70948 bash                 2022-04-29 16:15:33 UTC+0000   whoami
   70948 bash                 2022-04-29 16:15:34 UTC+0000   ?????U
   70948 bash                 2022-04-29 16:15:34 UTC+0000   history
   70948 bash                 2022-04-29 16:15:43 UTC+0000   echo $HISTFILE
   70948 bash                 2022-04-29 16:15:48 UTC+0000   unset HISTFILE
   70948 bash                 2022-04-29 16:15:49 UTC+0000   history
   70948 bash                 2022-04-29 16:15:57 UTC+0000   echo $HISTFILE
   70948 bash                 2022-04-29 16:16:06 UTC+0000   ll .bash_history
```

Wrapping Up

www.geant.org

What have you learned?

- Basic memory forensic analysis can be done with simple commands
  - Process lists
  - Network connections/sockets
  - Open files
  - Registry
  - Bash history
- No deep reverse engineering/malware knowledge required
  - Solid background about the operating system is mandatory
- Volatility 3 is promising, but Volatility can still do the job

Thank you

Any questions?

Next Webinar: Advanced ***Memory Analysis***

May 12th, 2022

www.geant.org

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

References

- VirtualBox
  - https://www.andreafortuna.org/2017/06/23/how-to-extract-a-ram-dump-from-a-running-virtualbox-machine/
  - https://gist.github.com/ssnkhan/60a2ab79480bd966876aac5d2c6c2e68
- Hibernation file
  - Hiber2bin: `https://github.com/comaeio/Hibr2Bin`
  - Volatility Hibernation file address space `https://github.com/volatilityfoundation/volatility/wiki/Hiber-Address-Space`
  - Windows hibernation file for fun 'n' profit, `https://www.blackhat.com/presentations/bh-usa-08/Suiche/BH_US_08_Suiche_Windows_hibernation.pdf`
- Linux swap_digger: `https://github.com/sevagas/swap_digger`

References: Books on Forensics

- Michael Hale Ligh, et al: *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*, John Wiley & Sons, Inc. 2014, ISBN: 978-1-118-82509-9
- Bruce Nikkel: *Pactical Forensic Imaging*, No Starch Press Inc. 2016, ISBN-13: 978-1-59327-793-2
- Harlan Carvey: *Windows Forensic Analysis*, Syngress Publishing Inc. 2009

References: Operating System Internals

- Pavel Yosifovich et al: *Windows Internals, Part 1 (System architecture), 7th Ed.*, Microsoft Press 2017, ISBN-13: 978-0735684188
- Allievi Andrea et al: *Windows Internals, Part 2 (Developer Reference), 7th Ed.*, Microsoft Press 2021, ISBN-13: 978-0135462409
- Robert Love: *Linux Kernel Development 3rd Ed.*, Addison-Wesley Professional 2010, ISBN-13: 978-0672329463
- Robert Love: *Linux System Programming: Talking Directly to The Kernel And C Library*, 2nd Ed., O'Reilly2013, ISBN-13 : 978-1449339531
- The FreeBSD Documentation Project: FreeBSD Handbook, `https://docs.freebsd.org/en/books/handbook/`
- The FreeBSD Documentation Project: FreeBSD Developers' Handbook, `https://docs.freebsd.org/en/books/developers-handbook/`
- The FreeBSD Documentation Project: FreeBSD Architecture Handbook, `https://docs.freebsd.org/en/books/arch-handbook/`
- Marshall Kirk McKusick et al.: *The Design and Implementation of the FreeBSD Operating System: Edition 2*, Addison-Wesley Professional 2014, ISBN-13: 978-0321968975

References: Images und Testcases

- Computer Forensic Reference Data Sets (CFReDS)
  `http://www.cfreds.nist.gov/`
- Digital Forensics Tool Testing Images
  `http://dftt.sourceforge.net/`
- Digital Forensics Research Workshop (DFRWS)
  `http://www.dfrws.org/`
- Honeynet Project Challenges
  `https://www.honeynet.org/challenges`

References: Memory Imaging Tools (Open Source)

- Microsoft AVML: https://github.com/microsoft/avml
- Volatility LiME: https://github.com/504ensicsLabs/LiME
  - Schlafwandlers kcore_dump https://schlafwandler.github.io/posts/dumping-/proc/kcore/
- Hal Pomeranz automation script for AVML/LiME: https://github.com/halpomeranz/lmg
- Nate Brunes fmem: https://github.com/NateBrune/fmem
- Velocidex Pmem Suite (lin|win|osx)pmem: https://winpmem.velocidex.com/
- Moonsols mdd (v 1.3, 2013, for very old Windows versions): https://sourceforge.net/projects/mdd/

Sample Forensic Distributions

- SIFT (SAS Investigative Forensic Toolkit): https://www.sans.org/tools/sift-workstation/
- CAINE (Computer Aided Investigative Environment): https://www.caine-live.net/
- GRML Forensic: https://grml-forensic.org/
- ALT Linux Rescue: https://en.altlinux.org/Rescue
- BlackArch: https://blackarch.org/
- BackBox: https://www.backbox.org/
- KALI (formerly Backtrack): https://www.kali.org/downloads/
- Matriux: http://www.matriux.com/
- Safe Boot Disk (Windows based): https://www.forensicsoft.com/help/SAFE_Boot1-1/

References: Standards

- US NIST Special Publication 800-86 *Guide to Integrating Forensic Techniques into Incident Response*, 2006, https://doi.org/10.6028/NIST.SP.800-86
- ENISA *Trainings for Cybersecurity Specialists*, https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-material?tab=articles
- IETF RFC 3227 *Guidelines for Evidence Collection and Archiving*, https://tools.ietf.org/html/rfc3227