CHEAT SHEET
V3.0

PURPOSE

Forensic Analysts are on the front lines of computer investigations. This guide aims to support Forensic Analysts in their quest to uncover the truth.

HOW TO USE THIS SHEET

When performing an investigation it is helpful to be reminded of the powerful options available to the investigator. This document is aimed to be a reference to the tools that could be used. Each of these commands runs locally on a system. ***This sheet is split into these sections:***

- **Mounting Images**
- **Shadow Timeline Creation**
- **Mounting Volume Shadow Copies**
- **Memory Analysis**
- **Recovering Data**
- **Creating Supert Timelines**
- **String Searches**
- **Sleuthkit Tools**
- **Stream Extraction**

MOUNTING DD IMAGES

mount -t *fstype [options] image mountpoint*

image can be a disk partition or dd image file

[Useful Options]

`ro`	mount as read only
`loop`	mount on a loop device
`noexec`	do not execute files
`loop`	mount on a loop device
offset=<BYTES>	logical drive mount
`show_sys_files`	show ntfs metafiles
`streams_interface=windows`	use ADS

Example: Mount an image file at mount_location

```
# mount -o
loop,ro,show_sys_files,streams_interface=windows
imagefile.dd /mnt/windows_mount
```

MOUNTING E01 IMAGES

```
# ewfmount image.E01 mountpoint
```

```
# mount -o
loop,ro,show_sys_files,streams_interface=windows
/mnt/ewf/ewf1 /mnt/windows_mount
```

MOUNTING VOLUME SHADOW COPIES

Stage 1 – Attach local or remote system drive

```
# ewfmount system-name.E01 /mnt/ewf
```

Stage 2 – Mount raw image VSS

```
# vshadowmount ewf1 /mnt/vss/
```

Stage 3 – Mount all logical filesystem of snapshot

```
# cd /mnt/vss
# for i in vss*; do mount -o
ro,loop,show_sys_files,streams_interface=
windows $i /mnt/shadow_mount/$i; done
```

RECOVER DELETED REGISTRY KEYS

 # deleted.pl <*HIVEFILE*>

 # deleted.pl

```
timeline.csv
```

`file|dir`		artifact target
`-f`	*<TYPE-INPUT>*	input format
`-o`	*<TYPE-OUTPUT>*	output format: default csv file
`-w`	*<FILE>*	append to log file
`-z`	*<SYSTEM TIMEZONE>*	
`-Z`	*<OUTPUT TIMEZONE>*	
`-r`		recursive mode
`-p`		preprocessors

```
# mount -o
loop,ro,show_sys_files,streams_interface=windows
imagefile.dd /mnt/windows_mount
```

```
# log2timeline -z EST5EDT -p -r -f win7
/mnt/windows_mount -w /cases/bodyfile.txt
```

```
# l2t_process -b /cases/bodyfile.txt -w
whitelist.txt 04-02-2012 > timeline.csv
```

STREAM EXTRACTION

```
# bulk_extractor <options> -o output_dir image
```

[Useful Options]

`-o outdir`	
`-f <regex>`	regular expression term
`-F <rfile>`	file of regex terms
`-Wn1:n2`	extract words between n1 and n2 in length
`-q nn`	quiet mode
`-e scanner`	enables a scanner
`-e wordlist`	enable scanner wordlist
`-e aes`	enable scanner aes
`-e net`	enable scanner net

```
# bulk_extractor -F keywords.txt -e net
-e aes -e wordlist -o /cases/bulk-
extractor-memory-output /cases/
memory-raw.001
```

REGISTRY PARSING - REGRIPPER

```
# rip.pl -r <HIVEFILE> -f <HIVETYPE>
```

[Useful Options]

`-r`	Registry hive file to parse <*HIVEFILE*>
`-f`	Use <*HIVETYPE*> (e.g. `sam`, `security`, `software`, `system`, `ntuser`)
`-l`	List all plugins

```
# rip.pl -r
/mnt/windows_mount/Windows/System32/config/SAM -f sam
> /cases/windowsforensics/SAM.txt
```

RECOVERING DATA

Create Unallocated Image (deleted data) using `blkls`

```
# blkls imagefile.dd >
unallocated_imagefile.blkls
```

Create Slack Image Using dls (for FAT and NTFS)

```
# blkls -s imagefile.dd > imagefile.slack
```

Foremost Carves out files based on headers and footers

`data_file.img` = raw data, slack space, memory, unallocated space

```
# ewfmount system-name.E01 /mnt/ewf
```

Step 2 – Mount VSS Volume

```
# cd /mnt/ewf
# vshadowmount ewf1 /mnt/vss
```

Step 3 – Run fls across ewf1 mounted image

```
# cd /mnt/ewf
# fls -r -m C: ewf1 >> /cases/vss-
bodyfile
```

Step 4 – Run fls Across All Snapshot Images

```
# cd /mnt/vss
# for i in vss*; do fls -r -m C: $i
>> /cases/vss-bodyfile; done
```

Step 5 – De-Duplicate Bodyfile using sort and uniq

```
# sort /cases/vss-bodyfile | uniq >
/cases/vss-dedupe-bodyfile
```

Step 6 – Run mactime Against De-Duplicated Bodyfile

```
# mactime -d -b /cases/vss-dedupe-
bodyfile -z EST5EDT MM-DD-YYYY..MM-
DD-YYYY > /cases/vss-timeline.csv
```

MEMORY ANALYSIS

```
vol.py command -f
/path/to/windows_xp_memory.img  --
profile=WinXPSP3x86
```

[Supported commands]

`connscan`	Scan for connection objects
`files`	list of open files process
`imagecopy`	Convert hibernation file
`procdump`	Dump process
`pslist`	list of running processes
`sockscan`	Scan for socket objects

SLEUTHKIT TOOLS

File System Layer Tools (Partition Information)

fsstat	Displays details about the file system	`# fsstat imagefile.dd`

Data Layer Tools (Block or Cluster)

blkcat	Displays the contents of a disk block	`# blkcat imagefile.dd block_num`
blkls	Lists contents of deleted disk blocks	`# blkls imagefile.dd > imagefile.blkls`
blkcalc	Maps between dd images and blkls results	`# blkcalc imagefile.dd -u blkls_num`
blkstat	Display allocation status of block	`# blkstat imagefile.dd cluster_number`

MetaData Layer Tools (Inode, MFT, or Directry Entry)

ils	Displays inode details	`# ils imagefile.dd`
istat	Displays information about a specific inode	`# istat imagefile.dd inode_num`
icat	Displays contents of blocks allocated to an inode	`# icat imagefile.dd inode_num`
ifind	Determine which inode contains a specific block	`# ifind imagefile.dd -d block_num`

Filename Layer Tools

fls	Displays deleted file entries in a directory inode	`# fls -rpd  imagefile.dd`
ffind	Find the filename that using the inode	`# ffind  imagefile.dd inode_num`

CHEAT SHEET
V1.1

PURPOSE

This cheat sheet supports the SANS FOR508 Advanced Forensics and Incident Response and SANS FOR526 Memory Analysis courses. It is not intended to be an exhaustive resource of Volatility™ or other highlighted tools. Volatility™ is a trademark of Verizon. The SANS Institute is not sponsored or approved by, or affiliated with Verizon.

HOW TO USE THIS DOCUMENT

Memory analysis is one of the most powerful tools available to forensic examiners. This guide hopes to simplify the overwhelming number of available options.

Analysis can be generally broken up into six steps:

1. **Identify Rogue Processes**
2. **Analyze Process DLLs and Handles**
3. **Review Network Artifacts**
4. **Look for Evidence of Code Injection**
5. **Check for Signs of a Rootkit**
6. **Dump Suspicious Processes and Drivers**

We outline the most useful Volatility™ plugins supporting these six steps here. Further information is provided for:

- **Memory Acquisition**
- **Converting Hibernation Files and Crash Dumps**
- **Memory Artifact Timelining**
- **Registry Analysis Volatility™ Plugins**
- **Memory Analysis Tool List**

MEMORY ACQUISITION

Remember to open command prompt as Administrator

Win32dd / Win64dd (x86 / x64 systems respectively)

`/f` Image destination and filename

```
C:\> win32dd.exe /f E:\mem.img
```

Mandiant Memoryze MemoryDD.bat

`-output` image destination

```
C:\> MemoryDD.bat -output E:\
```

Volatility™ WinPmem

`-` (single dash) Output to standard out

`-l` Load driver for live memory analysis

```
C:\> winpmem_<version>.exe E:\mem.img
```

CONVERTING HIBERNATION FILES AND CRASH DUMPS

Volatility™ **`imagecopy`**

`-f` Name of source file (crash dump, hibernation file, etc.)

`-O` Output file name

`--profile` Source OS from `imageinfo`

```
# vol.py imagecopy -f hiberfil.sys -O
hiber.img --profile=Win7SP1x64

# vol.py imagecopy -f Memory.dmp -O
memdmp.img --profile=Win7SP1x64
```

LOOK FOR EVIDENCE OF CODE INJECTION

`malfind` - Find injected code and dump sections

`-p` Show information only for specific PIDs

`-o` Provide physical offset of single process to scan

`--dump-dir` Directory to save extracted memory sections

http://code.google.com/p/volatility/

Mandiant Redline (Windows)
http://www.mandiant.com/resources/download/redline

Volafox (Mac OS X and BSD)
http://code.google.com/p/volafox/

GETTING STARTED WITH VOLATILITY™

Getting Help

```
# vol.py -h                 (show general options and supported plugins)
# vol.py plugin -h          (show plugin usage)
# vol.py plugin --info      (show available OS profiles)
```

Sample Command Line

```
# vol.py -f image --profile=profile plugin
```

Identify System Profile

```
imageinfo                   - Display memory image metadata
# vol.py -f mem.img imageinfo
```

Using Environment Variables

Set name of memory image (takes place of **-f**)

```
# export VOLATILITY_LOCATION=file:///images/mem.img
```

Set profile type (takes place of --profile=)

```
# export VOLATILITY_PROFILE=WinXPSP3x86
```

IDENTIFY ROGUE PROCESSES

pslist - High level view of running processes `# vol.py pslist`

psscan - Scan memory for EPROCESS blocks `# vol.py psscan`

pstree - Display parent-process relationships `# vol.py pstree`

CHECK FOR SIGNS OF A ROOTKIT

psxview - Find hidden processes using cross-view `# vol.py psxview`

driverscan - Scan memory for _DRIVER_OBJECTs `# vol.py driverscan`

apihooks - Find API/DLL function hooks
- **-p** Operate only on specific PIDs
- **-k** Scan kernel modules instead of user-mode objects

```
# vol.py apihooks
```

ssdt - Hooks in System Service Descriptor Table

```
# vol.py ssdt | egrep -v '(ntoskrnl|win32k)'
```

driverirp - Identify I/O Request Packet (IRP) hooks
- **-r** Analyze drivers matching REGEX name pattern

```
# vol.py driverirp -r tcpip
```

idt - Display Interrupt Descriptor Table `# vol.py idt`

ANALYZE PROCESS DLLS AND HANDLES

dlllist - List of loaded dlls by process
- **-p** Show information only for specific process identifiers (PIDs)

```
# vol.py dlllist -p 4,868
```

getsids - Print process security identifiers
- **-p** Show information only for specific PIDs

```
# vol.py getsids -p 868
```

handles - List of open handles for each process
- **-p** Show information only for specific PIDs
- **-t** Display only handles of a certain type

connscan - [XP] ID TCP connections, including closed `# vol.py connscan`

sockets - [XP] Print listening sockets (any protocol) `# vol.py sockets`

sockscan - [XP] ID sockets, including closed/unlinked `# vol.py sockscan`

netscan - [Win7] Scan for connections and sockets `# vol.py netscan`

DUMP SUSPICIOUS PROCESSES AND DRIVERS

dlldump - Extract DLLs from specific processes
- **-p** Dump DLLs only for specific PIDs
- **-b** Dump DLLs from process at physical memory offset
- **-r** Dump DLLs matching REGEX name pattern (case sensitive)
- **--dump-dir** Directory to save extracted files

```
# vol.py dlldump --dump-dir ./output -r metsrv
```

moddump - Extract kernel drivers
- **--dump-dir** Directory to save extracted files
- **-o** Dump driver using offset address (from `driverscan`)
- **-r** Dump drivers matching REGEX name pattern (case sensitive)

```
# vol.py moddump --dump-dir ./output -r gaopdx
```

procmemdump - Dump process to executable sample
- **-p** Dump only specific PIDs
- **-o** Specify process by physical memory offset
- **--dump-dir** Directory to save extracted files

```
# vol.py procmemdump --dump-dir ./out -p 868
```

memdump - Dump every memory section into a file
- **-p** Dump memory sections from these PIDs
- **--dump-dir** Directory to save extracted files

```
# vol.py memdump -dump-dir ./output -p 868
```

MEMORY ARTIFACT TIMELINING

The Volatility Timeliner plugin parses time-stamped objects found in memory images. Output is sorted by:

- **Process creation time**
- **Thread creation time**
- **Driver compile time**
- **DLL / EXE compile time**
- **Network socket creation time**
- **Memory resident registry key last write time**
- **Memory resident event log entry creation time**

timeliner
- **--output-file** Optional file to write output
- **--output=body** body for mactime

```
# vol.py -f mem.img timeliner --output-file out.csv
  --profile=Win7SP1x86
```

REGISTRY ANALYSIS VOLATILITY™ PLUGINS

hivelist - Find and list available registry hives `# vol.py hivelist`

hivedump - Print all keys and subkeys in a hive
- **-o** Offset of registry hive to dump (virtual offset from hivelist)

```
# vol.py hivedump -o 0xe1a14b60
```

printkey - Output a registry key, subkeys, and values
- **-K** "Registry key path"
- **-o** Only search hive at this offset (virtual offset from hivelist)

```
# vol.py printkey -K
"Software\Microsoft\Windows\CurrentVersion\Run"
```

userassist - Find and parse userassist key values
- **-o** Only search hive at this offset (virtual offset from hivelist)