Shadow Timeline Creation

Step 1 – Attach Local or Remote System Drive

```
# 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
```

SANS DFIR

DIGITAL FORENSICS & INCIDENT RESPONSE

SIFT WORKSTATION

Cheat Sheet v3.1

DFIR.SANS.ORG

Incident Responders are on the front lines of intrusion investigations. This guide aims to support DFIR analysts in their quest to uncover the truth.

How To Use This Sheet

When performing an response or an investigation it is helpful to be reminded of the powerful options available to the analyst. 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
- The Sleuthkit
- Stream Extraction

TIME TO GO HUNTING

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
ro	mount as read only
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=window
s imagefile.dd /mnt/windows_mount
```

Mounting E01 Images

```
# ewfmount image.E01 mountpoint

# mount -o
loop,ro,show_sys_files,streams_interface=window
s /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
```

Creating Super Timelines

```
# log2timeline.py plaso.dump [SOURCE]

# psort.py plaso.dump FILTER >
supertimeline.csv
```

EXAMPLE:

- **Step 1 – Create Comprehensive Timeline**
  - `# log2timeline.py plaso.dump /dev/sdc`
- **Step 2 – Filter Timeline**
  - `# psort.py -z "EST5EDT" -o L2tcsv plaso.dump "date > 'YYYY-MM-DD HH:MM:SS' AND date < 'YYYY-MM-DD HH:MM:SS'" > supertimeline.csv`
    artifact not the entire image.

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

Recover Deleted Registry Keys

 # deleted.pl <*HIVEFILE*>

 # deleted.pl /mnt/windows_mount/Windows/System32/config/SAM > /cases/windowsforensics/SAM_DELETED.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

```
# foremost -o outputdir -c
/path/to/foremost.conf  data_file.img
```

Sigfind - search for a binary value at a given offset (-o)

-o <*offset*> start search at byte <**offset**>

```
# sigfind <hexvalue> -o <offset> data file.img
```