Mounting a VHDX Image

Load NBD Module and Enable Partition Support

```
# modprobe nbd max_part=16
```

Connect the first NBD device (nbd0) to the VHDX disk image

```
# qemu-nbd -c /dev/nbd0 system-triage.vhdx
```

Inform the OS of partition table changes to /dev/nbd0

```
# partprobe /dev/nbd0
```

Create directory and mount the first partition of the nbd device

```
# mkdir /mnt/system-triage mountwin /dev/nbd0p1
/mnt/system-triage
```

Creating a Super Timeline with Plaso and Docker

➢ **Set Up Docker Container and Create a Super Timeline**

```
# docker run -v /path/on/host:/path/in/container
-v /path/on/host:/path/in/container
log2timeline/plaso:version# log2timeline.py --
timezone '[TIMEZONE]' --parsers
'[PARSER1,PARSER2]' --storage_file
/path/to/triage.plaso <data source>
```

➢ **Add the Filesystem Timeline to the Super Timeline**

```
# docker run -v /path/on/host:/path/in/container
log2timeline/plaso:version# log2timeline.py --
parsers 'mactime' --storage_file
/path/to/triage.plaso /path/to/mftecmd-final.body
```

➢ **Cut Timeframe and Output Super Timeline to CSV**

```
# docker run -v /path/on/host:/path/in/container
log2timeline/plaso:version# psort.py --output-
time-zone 'UTC' -o l2tcsv -w /path/to/output.csv
/path/to/triage.plaso "(((parser == 'winevtx')
and (timestamp_desc == 'Creation Time')) or
(parser != 'winevtx')) and ( date >
datetime('YYYY-MM-DDTHH:MM:SS') AND date <
datetime('YYYY-MM-DDTHH:MM:SS'))"
```

Registry Parsing - Regripper

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

`-r`	Registry hive file to parse <HIVEFILE>
`-f`	Specify <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/SAM.txt
```

Sleuthkit Tools

File System Layer Tools (Partition Information)

fsstat - Displays details about the file system
```
# fsstat imagefile.img
```

Data Layer Tools (Block or Cluster)

blkcat – Displays the contents of a disk block
```
# blkcat imagefile.img block_num
```

blkls – Displays contents of deleted disk blocks
```
# blkls imagefile.img > imagefile.blkls
```

blkcalc – Maps between disk images and blkls results
```
# blkcalc imagefile.img -u blkls_num
```

blkstat – Display allocation status of block
```
# blkstat imagefile.img block_number
```

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

ils – Displays summary information for all inodes
```
# ils imagefile.img
```

istat – Displays info about a specific inode
```
# istat imagefile.img inode_num
```

icat – Displays contents of blocks allocated to an inode
```
# icat imagefile.img inode_num
```

ifind – Determine which inode contains specific block
```
# ifind imagefile.img -d block_num
```

Filename Layer Tools

fls – Displays all deleted entries in the image
```
# fls -rpd imagefile.img
```

ffind – Find the filename and path for an inode
```
# ffind imagefile.img inode_num
```

SANS COMPUTER FORENSICS and INCIDENT RESPONSE

SIFT Cheat Sheet v4.0

POCKET REFERENCE GUIDE
SANS Institute
http://dfir.sans.org

by Marcus Guevara
http://sans.org/for508

Purpose

This cheat sheet supports the SANS Institute's FOR508 Advanced Incident Response, Threat Hunting, and Digital Forensics course. It is intended to be used as a reference for tools commonly found in the SANS Linux SIFT Workstation. It also serves as a reminder of the many Linux-based DFIR capabilities available.

TIME TO GO HUNTING

Recovering Data

tsk_recover [options] image [output_dir]

The default (without options) recovers only unallocated files
-a: Recover allocated files only
-e: Recover both allocated and unallocated files

```
# tsk_recover /path/to/diskimage.img
/path/to/recovery/output
```

photorec [options] [media]

No options or media needed for basic usage. Photorec operates through an interactive interface guiding the user step-by-step.

```
# photorec
```

foremost -o output -c signature_file -i target

Carves (recovers) files based on header and footer signatures
Target can be raw data, slack, memory, or unallocated space

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

Filesystem Timeline Creation

Step 1 - Extract Timeline Data Using `fls`

`fls` - from The Sleuth Kit (TSK) can extract timeline data from the metadata of file systems found within a disk image.

Options:

-m: sets the output format to mactime
-r: recursively lists all directories and files
-i: Specifies the type of image (raw, ewf, vhd, vmdk)
-o: [starting-sector]

Example:

```
# fls -m C: -r -i ewf /path/to/diskimage.E01 >
/output/bodyfile.txt
```

Example 2:

```
# fls -m C: -r -i raw /path/diskimage.dd >
/output/bodyfile.txt
```

Step 2 - Create Timeline Using `mactime`

```
# mactime –d –b /output/bodyfile.txt –z EST5EDT
MM-DD-YYYY..MM-DD-YYYY > /cases/timeline.csv
```

Using Zimmerman Tools in SIFT

The Eric Zimmerman tool suite can now run in Linux. Installation of .NET6 is required.

Download Zimmerman's Tools (.NET 6 versions)
`https://ericzimmerman.github.io/`

```
# dotnet EvtxECmd.dll -f ~/evtx_files/Sample.evtx
-csv ~/evtx_output

# dotnet bstrings.dll -f ~/binary_files/sample.bin
-o ~/sample_output
```

To run from anywhere, create a location alias for each of the tools:

```
# alias mftecmd='dotnet /path/to/tools/MFTECmd.dll

# alias bstrings='dotnet /path/to/tools/bstrings.dll
```

Search & Filter Tools

grep is a powerful tool used to search for specific patterns of text (regular expressions) within files.

`grep [options] pattern [files]`

-a: Treat binary files as text files
-i: Ignore case distinctions
-v: Invert the match, i.e., only show lines that do not match
-r: Recursively search through directories
-l: List only the names of files that contain the matching pattern
-n: Prefix each line of output with the line no. in its input file
-E: Interpret pattern as an extended regular expression (ERE)

Example: Reference a text file with a list of case-insensitive terms to exclude from a body file.

```
# grep -a -v -i -f /path/to/timeline_noise.txt
/input/mftecmd.body > /output/mftecmd-final.body
```

Example: Recursively search directories to list any files containing "base64", case-insensitive.

```
# grep -rail “base64”
```

find is used to search for files and directories in a directory hierarchy. It can be very useful for locating files based on various criteria and performing actions on those files.

`find [path] [options] [expression]`

-name pattern: Search for files matching the pattern.
-type t: Search for files of type t (e.g., f for regular files)
-mtime n: Search for files modified n days ago
-exec command {} \;: Execute command on matched file
-size n: Search for files of size n (e.g., +100M)
-user username: Search for files owned by username
-perm mode: Search for files with specific permissions

Example: recursively find all files ending in .exe and calculate an MD5 for each file.

```
# find . -type f -name \*.exe -exec md5sum {} \;
```

Mounting DD Images

`mount -t fstype [-o options] image mount_location`

`image` can be a disk partition or dd image file

[Useful options to use with "**-o**"]

ro	mount as read only
rw	mount as read-write
loop	mount on a loop device
no exec	do not allow execution
offset=<BYTES>	logical drive mount
show_sys_files	show ntfs metafiles
streams_interface=windows	display ADS

Example: Mount an image file at mount_location

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

Mounting E01 Images

```
# ewfmount diskimage.E01 /mnt/ewf

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

Mounting Windows Volume Shadow Copies

Step 1 - Access Evidence as a Raw Image (skip if not E01 format)

```
# ewfmount diskimage.E01 /mnt/ewf
```

Step 2 – Access Volume Shadows Within Raw Image

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

Step 3 – Mount Filesystem Within Volume Shadows

```
# cd /mnt/vss

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