Forensic Analysis Cheat Sheet v1.4

Forensics
POCKET REFERENCE GUIDE

SANS Institute
incidents@sans.org
+1 317.580.9756
http://forensics.sans.org
http://isc.sans.org

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
- Imaging Systems
- Integrity Checking
- Sorter
- Automated Forensic Data Collection
- Recovering Data
- Creating Timelines
- String Searches
- The Sleuthkit

The key to successful forensics is minimizing your data loss, accurate reporting, and a thorough investigation.

Imaging Systems

```
#dcfldd if=input file of=output file options
```

Example Input Files (if = *input file*)

LINUX

`/dev/hda`	(First IDE Physical Drive)
`/dev/hda2`	(Second Logical Partition)
`/dev/sda`	(First SCSI Physical Drive)

WINDOWS

`\\.\PhysicalDrive0`	(First Physical Drive)
`\\.\D:`	(Logical Drive D:)
`\\.\PhysicalMemory`	(Physical Memory)

Example Output Files (of = *output file*)

`\\hostname\share\imagefile.img`	(Windows Share)
`imagefile.img`	(Bit Image File)
`/dev/usb`	(USB Drive)
`/dev/hdb`	(2nd IDE Drive)

Useful Options

`bs= block size`	(sets the block size)
`count=N`	(copy only N blocks FILE)
`skip=N`	(skip ahead N blocks FILE)
`conv=noerror,sync`	(do not stop on errors)
`hashwindow=num`	(hash every num bytes)
`hashwindow=0`	(hash entire file)
`hashlog=filename`	(write md5 hash to file)

`mmls` to split out partitions from physical image

# mmls -t dos imagefile (-t is the type of drive)

```
Slot      Start (skip=)   End         Length (count=)   Description
02:  00:00  0000000063  0001028159  0001028097  Win95 FAT32 (0x0B)
```

Example: Use `dd` to carve logical image

# dd if=imagefile bs=512 skip=63 count=1028097 of=imagefile.partition1.img

Sorter

```
# sorter <options> -d dir imagefile.dd
```

Options:
-e: extension mismatch only
-s: Save the data to category directories (**-h** will produce thumbnails)
-d: directory for saving info
-c: config file
-m: mount point (so you can see full path of the file)

# sorter -h –m / –s -d <*outputdir*> imagefile.dd

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
```

Mounting DD Images

mount -t *fstype* [*options*] *image mountpoint*

image can be a disk partition or dd image file

Useful Options (`-o`)

`ro`	mount as read only
`loop`	mount on a loop device
`noexec`	do not execute files
`noatime`	do not adjust last access times
`uid=` *user_id*	mount as a specific user
`gid=` *group_id*	mount as a group
`umask=` *set permissions*	

Example: Mount an image file at mount_location

```
# mount -t fs_type -o loop,
ro,umask=0222,uid=forensic,gid=users
imagefile.dd /mnt/hack/mount_location
```

Mounting NTFS DD Images

```
# ntfs-3g [options] image mountpoint
```

image can be a disk partition or dd image file

Useful Options (`-o`)

`ro`	mount as read only
`loop`	mount on a loop device
`show_sys_files`	show ntfs volume files on mount

```
# ntfs-3g -o loop,ro,
imagefile.dd /mnt/hack/mount_location
```

Creating Timelines

Create the body file of all filename data using fls

```
# fls -m mountpoint -r imagefile.dd >
imagefile.body
```

mountpoint *= location of mount (**/** or* **C:** *)*

Create the timeline

```
# mactime -b imagefile.mac > timeline.all
```

String Searches

<u>`ASCII` string search and list the byte offset</u>

```
# srch_strings -t d imagefile.dd >
imagefile.ascii.str
```

<u>`UNICODE` string search and list byte offset</u>

```
# srch_strings -e l -t d imagefile.dd >
imagefile.uni.str
```

<u>Search for a specific string using grep</u>

GREP Useful Options

`-i`	ignore case
`-f`	`dirty_word_list_filename`

```
# grep -i password -f dirty_words.txt
imagefile.ascii.str
```

Automated Forensic Data Collection

<u>WINDOWS (Windows Forensic Toolchest)</u>

Use WFT to automate the gathering of information on your windows system. You can execute this from a CDROM D:
`D:\IR\wft\wft.exe`

Answers should be `DEFAULT` except for:

```
   1. What is the toolpath you would like
      to use?
     D:\IR
   2. What is the destination path you
      would like to use?

\\<IPADDRESS>\images\windowsforensics\wft\
```

Recovering Data

<u>Create Unallocated Image</u> (deleted data) using **`blkls`**

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

<u>Create Slack Image</u> Using dls (for FAT and NTFS)

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

<u>Foremost</u> 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
```

<u>Sigfind</u> - search for a binary value at a given offset (-o)

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

```
# sigfind <hexvalue> -o <offset> data_file.img
```