

# Forensics Tools in Kali

January 6, 2018 By Raj Chandel

Kali Linux is often thought of in many instances, it's one of the most popular tools available to security professionals. It contains a robust package of programs that can be used for conducting a host of security-based operations. One of the many parts in its division of tools is the forensics tab, this tab holds a collection of tools that are made with the explicit purpose of performing digital forensics.

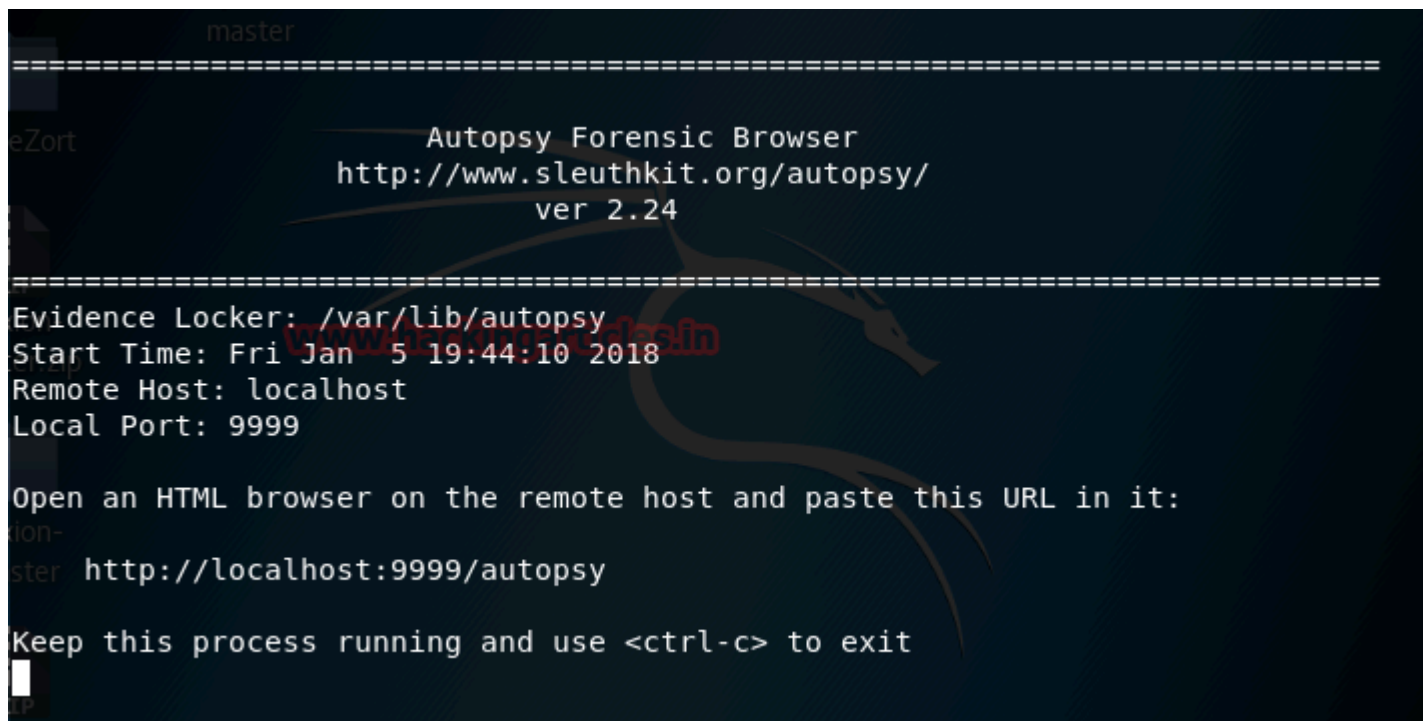
Forensics is becoming increasingly important in today's digital age where many crimes are committed using digital technology, having an understanding of forensics can greatly increase the chance of making certain that criminals don't get away with a crime.

This article is aimed at giving you an overview of the forensic capabilities possessed by Kali Linux.

So, let's start with the programs as they appear in the forensics menu:

## Autopsy

A tool used by the military, law enforcement and other entities when it comes time to perform forensic operations. This package is probably one of the most robust ones available through open source, it combines the functionalities of many other smaller packages that are more focused in their approach into one neat application with a web browser based UI.

A terminal window with a dark background and light-colored text. The text shows the installation of Autopsy Forensic Browser. It includes the URL 'http://www.sleuthkit.org/autopsy/' and the version 'ver 2.24'. Below this, it shows the installation path '/var/lib/autopsy', the start time 'Fri Jan 5 19:44:10 2018', the remote host 'localhost', and the local port '9999'. It then instructs the user to open an HTML browser on the remote host and paste the URL 'http://localhost:9999/autopsy'. Finally, it says to keep the process running and use '<ctrl-c>' to exit. There is a faint watermark of a dragon in the background of the terminal output.

```
master
=====
Autopsy Forensic Browser
http://www.sleuthkit.org/autopsy/
ver 2.24
=====
Evidence Locker: /var/lib/autopsy
Start Time: Fri Jan 5 19:44:10 2018
Remote Host: localhost
Local Port: 9999

Open an HTML browser on the remote host and paste this URL in it:
http://localhost:9999/autopsy

Keep this process running and use <ctrl-c> to exit
```

It is used to investigate disk images. When you click on Autopsy, it starts the service and its user interface can be accessed on the web browser at **<http://9999:localhost/autopsy>**. It gives the user a full range of options

required to create a new case file: Case Name, Description, Investigators Name, Hostname, Host time zone, etc.

Its functionalities include – Timeline analysis, keyword search, web artifacts, hash filtering, data carving, multimedia and indicators of compromise. It accepts disk images in RAW or E01 formats and generates reports in HTML, XLS and body file depending on what is required for a particular case.

Its robustness is what makes it such a great tool, be it case management, analysis or reporting, this tool has you covered.



## Binwalk

This tool is used while dealing with binary images, it has the capability of finding the embedded file and executable code by exploring the image file. It is a very powerful tool for those who know what they are doing, if used right, it can be used to find sensitive information hidden in firmware images that can be lead to uncovering a hack or used to find a loophole to exploit.

This tool is written in python and uses the libmagic library, making it perfect for usage with magic signatures created for Unix file utility. To make things easier for investigators, it contains a magic signature file which holds the most commonly found signatures in firmware's, making it easier to spot anomalies.

```
winphisher-
Binwalk v2.1.1
Craig Heffner, http://www.binwalk.org

Usage: binwalk [OPTIONS] [FILE1] [FILE2] [FILE3] ...

Signature Scan Options:
-B, --signature          Scan target file(s) for common file signatures
-R, --raw=<str>          Scan target file(s) for the specified sequence of bytes
-A, --opcodes            Scan target file(s) for common executable opcode signatures
-m, --magic=<file>       Specify a custom magic file to use
-b, --dumb               Disable smart signature keywords
-I, --invalid            Show results marked as invalid
-x, --exclude=<str>      Exclude results that match <str>
-y, --include=<str>      Only show results that match <str>

Extraction Options:
-e, --extract            Automatically extract known file types
-D, --dd=<type:ext:cmd>  Extract <type> signatures, give the files an extension of <ext>
-M, --matryoshka         Recursively scan extracted files
-d, --depth=<int>        Limit matryoshka recursion depth (default: 8 levels deep)
-C, --directory=<str>    Extract files/folders to a custom directory (default: current)
-j, --size=<int>         Limit the size of each extracted file
-n, --count=<int>        Limit the number of extracted files
-r, --rm                Delete carved files after extraction
-z, --carve              Carve data from files, but don't execute extraction utilities
```

## Bulk Extractor

This is a very interesting tool when an investigator is looking to extract certain kind of data from the digital evidence file, this tool can carve out email addresses, URL's, payment card numbers, etc. This tool works on directories, files, and disk images. The data can be partially corrupted or it can be compressed, this tool will find its way into it.

The tool comes with features which help create a pattern in the data that is found repeatedly, such as URL's, email ids and more and presents them in a histogram format. It has a feature by which it creates a word list from the data found, this can assist in cracking the passwords of encrypted files.

```

bulk_extractor version 1.6.0-dev
Usage: bulk_extractor [options] imagefile
    runs bulk_extractor and outputs to stdout a summary of what was found where

Required parameters:
    imagefile the file to extract
or -R filedir - recurse through a directory of files
                HAS SUPPORT FOR E01 FILES
                HAS SUPPORT FOR AFF FILES
    -o outdir    - specifies output directory. Must not exist.
                  bulk_extractor creates this directory.

Options:
    -i           - INFO mode. Do a quick random sample and print a report.
    -b banner.txt - Add banner.txt contents to the top of every output file.
    -r alert_list.txt - a file containing the alert list of features to alert
                      (can be a feature file or a list of globs)
                      (can be repeated.)
    -w stop_list.txt - a file containing the stop list of features (white list)
                      (can be a feature file or a list of globs)
                      (can be repeated.)
    -F <rfile>    - Read a list of regular expressions from <rfile> to find
    -f <regex>    - find occurrences of <regex>; may be repeated.
                      results go into find.txt
    -q nn         - Quiet Rate; only print every nn status reports. Default 0; -1 for
    -s frac[:passes] - Set random sampling parameters

```

## Chkrootkit

This program is mostly used in a live boot setting. It is used to locally check the host for any installed rootkits. It comes in handy trying to harden an endpoint or making sure that a hacker has not compromised a system.

It has the capability to detect system binaries for rootkit modification, last log deletions, quick and dirty string replacements, and temp deletions. This is just a taste of what it can do, the package seems simple at first glance but to a forensic investigator, its capabilities are invaluable.

```

Usage: /usr/sbin/chkrootkit [options] [test ...]
Options:
    -h           show this help and exit
    -v           show version information and exit
    -l           show available tests and exit
    -d           debug
    -q           quiet mode
    -x           expert mode
    -e           exclude known false positive files/dirs, quoted,
                  space separated, READ WARNING IN README
    -r dir       use dir as the root directory
    -p dir1:dir2:dirN path for the external commands used by chkrootkit
    -n           skip NFS mounted dirs

```

## Foremost



Deleted files which might help solve a digital incident? No problem, Foremost is an easy to use open source package that can carve data out of formatted disks. The filename itself might not be recovered but the data it holds can be carved out.

Foremost was written by US Air Force special agents. It can carve files by referencing a list of headers and footers even if the directory information is lost, this makes for fast and reliable recovery.

```
foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus.
$ foremost [-v|-V|-h|-T|-Q|-q|-a|-w|-d] [-t <type>] [-s <blocks>] [-k <size>]
  [-b <size>] [-c <file>] [-o <dir>] [-i <file>]

-V - display copyright information and exit
-t - specify file type. (-t jpeg,pdf ...)
-d - turn on indirect block detection (for UNIX file-systems)
-i - specify input file (default is stdin)
-a - Write all headers, perform no error detection (corrupted files)
-w - Only write the audit file, do not write any detected files to the disk
-o - set output directory (defaults to output)
-c - set configuration file to use (defaults to foremost.conf)
-q - enables quick mode. Search are performed on 512 byte boundaries.
-Q - enables quiet mode. Suppress output messages.
-v - verbose mode. Logs all messages to screen
```

## Galleta

When following a trail of cookies, this tool will parse them into a format that can be exported into a spreadsheet program.

Understanding cookies can be a tough nut to crack, especially if the cookies might be evidence in a cyber-crime that was committed, this program can lend a hand by giving investigators the capability to structure the data in a better form and letting them run it through an analysis software, most of which usually require the data to be in some form of a spreadsheet.

```
Usage: galleta [options] <filename>
       -d Field Delimiter (TAB by default)
```

## Hashdeep

This program is a must when dealing with hashes. Its defaults are focused on MD5 and SHA-256. It can be existing files that have moved in a set or new files placed in a set, missing files or matched files, Hashdeep can work with all these conditions and give reports that can be scrutinized, it is very helpful for performing audits.

One of its biggest strengths is performing recursive hash computations with multiple algorithms, which is integral when the time is of the essence.

```

hashdeep version 4.4 by Jesse Kornblum and Simson Garfinkel.
$ hashdeep [OPTION]... [FILES]...
-c <alg1,[alg2]> - Compute hashes only. Defaults are MD5 and SHA-256
                  legal values: md5,sha1,sha256,tiger,whirlpool,
-p <size> - piecewise mode. Files are broken into blocks for hashing
-r - recursive mode. All subdirectories are traversed
-d - output in DFXML (Digital Forensics XML)
-k <file> - add a file of known hashes
-a - audit mode. Validates FILES against known hashes. Requires -k
-m - matching mode. Requires -k
-x - negative matching mode. Requires -k
-w - in -m mode, displays which known file was matched
-M and -X act like -m and -x, but display hashes of matching files
-e - compute estimated time remaining for each file
-s - silent mode. Suppress all error messages
-b - prints only the bare name of files; all path information is omitted
-l - print relative paths for filenames
-i/-I - only process files smaller than the given threshold
-o - only process certain types of files. See README/manpage
-v - verbose mode. Use again to be more verbose
-d - output in DFXML; -W FILE - write to FILE.
-j <num> - use num threads (default 4)

```

## Volafox

This is a memory analysis tool that has been written in Python, it is focused towards memory forensics for MAC OS X. It works on the Intel x86 and IA-32e framework. If you're trying to find malware or any other malicious program that was or is residing on the system memory, this is the way to go.

```

volafox: Mac OS X Memory Analysis Toolkit
project: http://code.google.com/p/volafox
support: 10.6-8; 32/64-bit kernel
  input: *.vmem (VMWare memory file), *.mmr (Mac Memory Reader, flattened x86, IA-32e)
  usage: python /usr/bin/volafox -i IMAGE [-o COMMAND [-vp PID][-x PID][-x KEXT_ID][-x TASKID]]

Options:
-o CMD           : Print kernel information for CMD (below)
-p PID          : List open files for PID (where CMD is "lsof")
-v             : Print all files, including unsupported types (where CMD is "lsof")
-x PID/KID/TASKID : Dump process/task/kernel extension address space for PID/KID/Task ID (where CMD is
                  "ps"/"kextstat"/"tasks")

COMMANDS:
system_profiler : Kernel version, CPU, and memory spec, Boot/Sleep/Wakeup time
mount           : Mounted filesystems
kextstat        : KEXT (Kernel Extensions) listing
ps             : Process listing
tasks          : Task listing (& Matching Process List)
sysstab        : Syscall table (Hooking Detection)
mtt            : Mach trap table (Hooking Detection)
netstat        : Network socket listing (Hash table)
lsof           : Open files listing by process (research, osxmem@gmail.com)
pestate        : Show Boot information (experiment)
efiinfo        : EFI System Table, EFI Runtime Services(experiment)
keychaindump   : Dump master key candidates for decrypting keychain(Lion, ML)

```

## Volatility

Probably one of the most popular frameworks when it comes to memory forensics. This is a python based tool that lets investigators extract digital data from volatile memory (RAM) samples. It is compatible to be used with the majority of the 64 and 32-bit variants of windows, selective flavors of Linux distros including android. It accepts memory dumps in various forms, be it raw format, crash dumps, hibernation files or VM snapshots, it can give a keen insight into the run-time state of the machine, this can be done independently of the host's investigation.

Here's something to consider, decrypted files and passwords are stored in the RAM, and if they are available, investigating files that might be encrypted in the hard disk can be a lot easier to get into and the overall time of the investigation can be considerably reduced.

```
Volatility Foundation Volatility Framework 2.6
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help                list all available options and their default values.
                           Default values may be set in the configuration file
                           (/etc/volatilityrc)
  --conf-file=/root/.volatilityrc
                           User based configuration file
  -d, --debug               Debug volatility
  --plugins=PLUGINS         Additional plugin directories to use (colon separated)
  --info                    Print information about all registered objects
  --cache-directory=/root/.cache/volatility
                           Directory where cache files are stored
  --cache                   Use caching
  --tz=TZ                   Sets the (Olson) timezone for displaying timestamps
                           using pytz (if installed) or tzset
  -f FILENAME, --filename=FILENAME
                           Filename to use when opening an image
  --profile=WinXPSP2x86     Name of the profile to load (use --info to see a list
                           of supported profiles)
  -l LOCATION, --location=LOCATION
                           A URN location from which to load an address space
```

We will be following up this particular article with an in-depth review of the tools we have mentioned, with test cases.

Have fun and stay ethical.