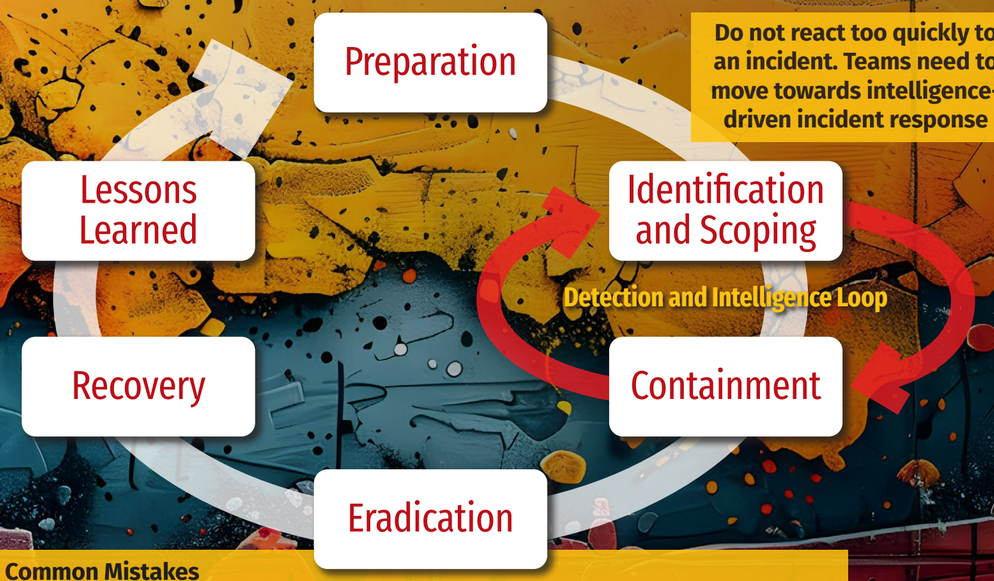
SANS DFIR

Linux Incident Response

THE INCIDENT RESPONSE PROCESS



Do not react too quickly to an incident. Teams need to move towards intelligence-driven incident response

Common Mistakes

- » Rushing the identification and scoping phase means you don't know the extent of the compromise
- » Skipping containment allows the attackers to pivot
- » Hasty actions can alert the attackers to change tactics and rarely leads to complete eradication
- » Business pressure to return to normal creates a rush to get to remediation, even if other stages are incomplete

digital-forensics.sans.org

Poster was created by Kathryn Hedley primarily based on the research and knowledge of Taz Wake in authoring FOR577.

DFIR_FOR577_0325

PACKAGE MANAGERS

› apt = Advanced Package Tool
- » Default on Ubuntu systems
- » Tool for Debian systems
- » Acts as a front end for Debian packages
- » Manages dependencies
- » Repository list (includes default repos): **/etc/apt/sources.list**
- » User-added repository lists: **/etc/apt/sources.list.d/*.list**
  - ▸ e.g., added using the command **apt-add-repository**

› dpkg = Debian Package
- » Low-level tool for Debian systems
- » Does not manage dependencies
- » Typically used for the installation of standalone packages (.deb files)
- » In general, .deb files are not digitally signed; trust is placed on the repository
- » View the contents of a .deb file: **ar -tv [package]**
- » List the contents of the data.tar.* sub-file within a .deb file: **dpkg -c [package]**
- » Extract the contents of a .deb file: **ar -xov [package]**
- » Extract the contents of the data.tar.* sub-file within a .deb file: **dpkg -x [package]**
- » Display any metadata associated with a .deb file: **dpkg -I [package]**
  - ▸ Note: The *ar* tool may not be installed by default; install the **binutils** package

› rpm = RPM Package Manager
- » Low-level tool for Red Hat-based distros
- » Found on RHEL/CentOS
- » Does not manage dependencies
- » Effectively equivalent to dpkg on Debian
- » Convert an .rpm file to a .cpio file: **rpm2cpio [RPM file] > [CPIO file]**
- » View the contents of a .cpio file: **cpio -i -tv < [CPIO file]**
- » Extract an individual file from a .cpio file: **cpio -i --to-stdout [file] < [CPIO file]**

› yum = Yellowdog Updated, Modified
- » Default on Red Hat Enterprise Linux and CentOS
- » High-level wrapper for the rpm tool
- » Manages dependencies
- » Python 2-based tool with some additional libraries
- » Repository lists (includes default repos): **/etc/yum.repos.d/*.repo**

› dnf = Dandified YUM
- » Default on Fedora
- » Rewrite of yum
- » Manages dependencies
- » Uses libdnf framework, librepo, and libcomps
- » Repository lists (includes default repos): **/etc/yum.repos.d/*.repo**

› apk = Alpine Package Keeper
- » Lightweight package management tool used on Alpine Linux
- » Designed for use in low-resource environments such as containers
- » Manages dependencies

› snap
- » App Store and package manager developed by Canonical
- » Snap applications run in a sandbox environment with limited access to the host
- » Self-contained compressed filesystem using SquashFS

› synaptic
- » GUI for APT
- » Used on Debian systems
- » Uses deb and rpm packages
- » Less common than snap

› zypper = Zen/YaST Packages Patches Patterns Products
- » Command-line interface for YaST (Yet another setup tool)
- » Uses rpm-based packages
- » Found on openSUSE and SUSE Linux Enterprise
- » Repository lists (includes default repos): **/etc/zypp/repos.d/***

Action	APT	DNF	ZYPPER	APK
Install PackageName	apt install PackageName	dnf install PackageName	zypper in PackageName	apk add PackageName
Update Package Database	apt update	dnf check-update	zypper ref	apk update
Remove PackageName	apt remove PackageName	dnf remove --nodeps PackageName	zypper rm -RU PackageName	apk del PackageName
Upgrade all packages	apt upgrade	dnf update	zypper up	apk upgrade
List available upgrades	apt list --upgradeable	dnf check-update	zypper lu	apk info (see notes)

Threat Hunting Package Managers

› Review all installed applications
- » Look for random or short (1–2 character) application names
- » Look for application names that relate to known indicators
- » Look for applications installed during any time period of interest
- » Look for unusual repository use

› Review individual packages
- » Check internal file timestamps
- » Check any signatures; they should be intact, valid, and expected
- » Review any pre- or post-installation scripts to check for unusual behavior

› Extract individual installer files and review them

USEFUL LINUX COMMANDS

awk . . . . . Powerful data manipulation and pattern matching

cat . . . . . Concatenate files or display text

column . . . . . Format output with columns
- -t . . . . . Table format output
- -s . . . . . Specify an input separator (the default is whitespace)
- -o . . . . . Specify an output separator (the default is two spaces)
- -J . . . . . JSON output (requires column names to be specified)
- -N . . . . . Specify the column names with a comma-separated list; this will be used for the table header
- -n . . . . . Provide a name for the table (the default is "table")

cut . . . . . Slice input to get the bits you want
- -d . . . . . Use this as a delimiter to cut on
- -f . . . . . Select this field or combination of fields

date . . . . . Print or set the system time
- -d [STRING] . . . . . Display the date referenced by STRING, not "now"
- -u . . . . . Display the time in UTC

file . . . . . Check the file's file signature
- -z . . . . . Look inside zip files/compressed archives

grep . . . . . Search file contents for a pattern match
- -r . . . . . Search recursively
- -i . . . . . Case-insensitive search
- -o . . . . . Print only the matched [non-empty] parts of a matching line
- -v . . . . . Exclude matches
- -E . . . . . Pattern match to an extended regular expression
- -a . . . . . Force grep to treat all input as ASCII
- -A# . . . . . Show # (number) lines after the matching line
- -B# . . . . . Show # (number) lines before the matching line
- -C# . . . . . Show # (number) lines before and after the matching line

head . . . . . Show the first N lines of a file
- -n # . . . . . Show # number of lines
- -c # . . . . . Show # number of bytes

hexdump . . . . . Display file contents in hex (or octal etc.)
- -C . . . . . Canonical hex + ASCII display, which is similar output to xxd
- -n [number] . . . . . Interpret only [number] bytes of input
- -s [number] . . . . . Skip [number] of bytes from the start

kill . . . . . Signal process
- -s . . . . . Send a specific signal name or number
- -1 . . . . . Send the signal to all processes with a PID greater than 1
- -l . . . . . Print a list of signal names

ls . . . . . List directory contents
- -a . . . . . Show hidden files
- -l . . . . . Use long format and show more details
- -h . . . . . Use human-readable numbers; 1024 = 1k, etc.
- -i . . . . . Show inode numbers

man . . . . . Display the manual for a command
- -k [keyword] . . . . . Search the manual pages for entries that refer to the keyword
- -f [command] . . . . Display a concise description of the command

md5sum filename . . . Generate an MD5 hash of filename

sha1sum filename . . . Generate a SHA1 hash of filename

sha256sum filename. Generate a SHA256 hash of filename

sort . . . . . Takes input and sends to stdout in order
- -r . . . . . Reverse the sort order
- -n . . . . . Compare strings based on numeric value
- -b . . . . . Ignore leading blanks
- -u . . . . . Suppress all but one in each set where keys are equal

tail . . . . . Show the last N lines of a file
- -n # . . . . . Show # number of lines
- -c # . . . . . Show # number of bytes

tar . . . . . Manage archives
- -tf [file] . . . . . List the contents of [file] archive
- -xf [file] . . . . . Extract the contents of [file] archive

uniq . . . . . Removes duplicate adjacent lines
- -d . . . . . Only return the duplicate lines
- -c . . . . . Count the number of occurrences
- -i . . . . . Ignore the case

wc . . . . . Count the number of bytes, lines, or words in a file
- -l . . . . . Show the number of lines
- -c . . . . . Show the number of bytes
- -w . . . . . Show the number of words
- -m . . . . . Show the number of characters

xxd . . . . . Making hexdumps
- -s [number] . . . . . Start [number] bytes into the file (seek)
- -l [number] . . . . . Show [number] bytes in total (length)
- -r . . . . . Convert hexdump back into a binary format (reverse)

zcat . . . . . Read compressed files
- -d . . . . . Decompress the file
- -r . . . . . Recurse through directories

zgrep . . . . . Search compressed data sets

Common Regex metacharacters:

- **.** . . . . . Match any single character, except a newline
- **^** . . . . . Anchor the match at the start of a line
- **$** . . . . . Anchor the match at the end of a line
- ***** . . . . . Match the preceding element zero or more times
- **+** . . . . . Match the preceding element one or more times
- **?** . . . . . Match the preceding element zero or one time
- **[ ]** . . . . . Match any one of the characters inside the brackets. A range of characters can be specified using a hyphen. An example would be [a-z]
- **|** . . . . . Acts as a logical OR between patterns
- **( )** . . . . . Group parts of a pattern together

Escape a metacharacter to search for the literal character using backslash (****)

INCIDENT RESPONSE AND THE SLEUTH KIT

Incident Response Task	TSK Command
Review an image to understand its structure.	mmls [image]
Identify the layout of a filesystem.	fsstat -f [filesystem] -I [image type] [image]
Extract all filesystem timestamps for a timeline.	fls -r -m [image]
Read the contents of a file with a given filename.	fcat [file path] [image]
Read the contents of a file at a given inode.	icat [image] [inode]
Review the metadata of a file.	istat [image] [inode]
Extract the contents of a data block.	blkcat [image] [block number]
Identify the filename for a given inode.	ffind [image] [inode]
Identify the inode for a given filename.	ifind [image] -n [file path]
Extract an individual disk partition.	mmcat [image] [partition number] > [file]
List unallocated inodes in Mactime format.	ils -m [image]
List all deleted regular files and their inodes.	fls -Frd [image]

GENERAL FILESYSTEM TERMINOLOGY

BLOCK	An area of storage space that is the smallest accessible unit on a filesystem.
BLOCK GROUPS	Groups of blocks containing filesystem-related information.
EXTENT	A group of consecutive blocks
SUPERBLOCK	The block that contains the filesystem metadata that defines other structures.
INODE	An "index node." This stores file metadata and its location on the drive.
INODE BITMAP	Records which inodes are in use within a group and the locations of the data block.

TIMESTAMPS

M = mtime

Modified: Data content change time
- » Time the file data was last modified

A = atime

Accessed: Data last access time
- » Approximate time when the file data was last accessed for some reason, not necessarily by the user
- » Not reliable

C = ctime

Changed Metadata: Metadata change time
- » The time the inode entry was last updated

B = btime = crtime

Birth: File creation time
- » Time file was created on this filesystem
- » Not always recorded

General Linux Filesystem Time Rules

	File Creation	File Access	Content Change	File Copy[2]	File Move Rename	File Move (New Mount Point)	File Delete[3]
M Modification	Set to time of creation	**No change**	Set to time of change	Set to time of copy	**No change**	**No change**	Set to time of deletion
A Access	Set to time of creation	Set to time of access[1]	Set to time of change[1]	Set to time of copy	**No change**	**No change**	**No change**
C Metadata Change	Set to time of creation	**No change**	Set to time of change	Set to time of copy	Set to time of move	Set to time of move	Set to time of deletion
B Creation	Set to time of creation	**No change**	**No change**	Set to time of copy	**No change**	Set to time of move	**No change**

[1] This is controlled by the atime settings when the filesystem is mounted.
[2] These timestamps are for the new file. The source file may have an atime change.
[3] This varies by filesystem type. XFS does not set the mtime. EXT stores a 32-bit deletion timestamp. XFS does not.

FILESYSTEM KEY CHARACTERISTICS

Filesystem	Theoretical Maximum Single-File Size	Theoretical Maximum Filesystem Size	Timestamps	Journal	Notes
EXT3	2 TiB	16 TiB	32-bit, no crtime Subject to "2038 timestamp issue"	Yes	Older, found on infrastructure
EXT4	16 TiB	1 EiB	64-bit, crtime Subject to "2446 timestamp issue"	Yes	Default on Debian family
XFS	8 EiB	8 EiB	64-bit, crtime Subject to "2446 timestamp issue"	Yes	Default on RHEL family
BTRFS	16 EiB	16 EiB	64-bit, crtime Subject to "2446 timestamp issue"	CoW	Found on NAS devices and some Fedora/SUSE versions
ZFS	16 EiB	256 ZiB	True 64-bit, crtime Not subject to "2446 timestamp issue"	CoW	Sometimes found on HPC/Unix systems
UFS1	8 ZiB	8 ZiB	32-bit, no crtime	No	Mainly used on Unix systems
UFS2	8 ZiB	8 ZiB	True 64-bit, crtime	No	Mainly used on Unix systems

TIMELINING

Creating a Filesystem Timeline

› Step 1: Create a bodyfile:
- » From a simple source image: **fls -a -r -m [path prefix] [source image] > [bodyfile]**
- » From a complex source image: **fls -a -r -m [path prefix] -I [image format] -om [partition offset] [source image] > [bodyfile]**

› Step 2: Convert to human-readable format:
- » Convert to CSV: **mactime -d -z UTC -b [bodyfile] [time slice] > [timeline.csv]**

› Step 3: Analyze

Creating a Super Timeline

› Option 1: psteal
- » **psteal.py --source [source image] -o [output format] -w [output file]**

› Option 2: log2timeline and psort
- » Extract events from image into storage file: **log2timeline.py --storage-file [storage file] [source image]**
- » Create a human-readable timeline from the storage file: **psort.py -w [output file] [storage file] [event filter]**

ACQUIRING DATA

Common Disk Evidence Collection Tools

› dd
- » Installed by default in most distros
- » Usage: **dd if=[input file] of=[output file] bs=[blocksize] conv=sync,noerror**

› dcfldd
- » Not installed by default
- » Forked version of dd
- » Usage: **dcfldd if=[input file] of=[output file] hash=[hash format] hashlog=[hashlog file] conv=sync,noerror**

› dc3dd
- » Not installed by default
- » Patched, more verbose version of dd
- » Usage: **dc3dd if=[input file] of=[output file] hash=[hash format] hlog=[hashlog file] log=[diagnostic log file]**

› ewfaquire
- » Not installed by default
- » Part of libewf
- » Writes E01 format files
- » Usage: **ewfacquire [device]**

Common Manual Remote Collection Tools

› dd over ssh:
- » On the source system: **dd if=[input file] | ssh [username]@[ip address] "dd of=[output file] bs=[blocksize] conv=sync,noerror"**

› dd image sent to a Google Cloud Platform storage bucket:
- » On the source system: **dd if=[input file] bs=[blocksize] | gzip | gsutil cp - gs://[bucketname/objectname]**

› dd over netcat:
- » On the source system: **dd if=[input file] bs=[blocksize] | bzip2 -c | nc [ip address] [port]**
- » On the destination system: **nc -nvlp [port] | bzip2 -d | dd bs=[blocksize] of=[output file]**

Acquiring RAM

› Acquire Volatile Memory for Linux (AVML):
- » Open-source tool written in Rust, available from: **https://github.com/microsoft/avml**
- » Does not require on-target compilation of the binary
- » Includes support for various cloud platforms, such as AWS, Microsoft Azure, and Google Cloud, to store memory images
- » Supports compression of memory images using the **--compress** option
- » Usage: **avml [output file]**

› Loadable Kernel Module (LiME):
- » Requires on-target compilation of the binary, or can be compiled externally using the same distro version and kernel release and build process as the target, but this can be complicated
- » This project is no longer maintained

› Direct access to the system's physical memory in /dev/mem:
- » Direct access to physical memory on a running system can potentially cause system crashes or other damage
- » When power is lost, this ceases to exist and is not recovered in standard disk images

› Access to the virtual filesystem representing RAM at /proc:
- » The proc filesystem contains a hierarchy of special files that represent running processes and various system resources:
  - ▸ **cmdline:** process command line and any arguments—use **cat** to view
  - ▸ **comm:** process command name; similar to cmdline but doesn't include the name of any application that called the process—use **cat** to view
  - ▸ **cwd:** current working directory that the process is running in—use **ls** to view
  - ▸ **environ:** process environment assigned to the running application—use **cat** to view
  - ▸ **exe:** symbolic link to the executable running the process—use **ls** to view
  - ▸ **fd:** folder containing details on where the file descriptors are pointing—use **ls -al** to view
  - ▸ **map_files:** folder containing links to shared objects and memory-mapped files used by the application—use **ls -al** to view
  - ▸ **maps:** file that contains similar information to the map_files folder. Use **cat** to view
  - ▸ **mountinfo:** file that holds information about any active mount points. Use **cat** to view
  - ▸ **net:** directory containing networking information for the process—this includes TCP/UDP ports, route tables, and network statistics—use **ls** to view.
  - ▸ **root:** link to the root folder for this application—this can be useful for identifying applications running in unusual environments, such as chroot—use **ls -al** to view
  - ▸ **sched:** file containing the CPU scheduling parameters as they apply to the PID being reviewed—use **cat** to view
  - ▸ **stack:** file that describes the current kernel stack of the process being reviewed—use **cat** to view
  - ▸ **status:** file containing the current status of a process—use **cat** to view
    - running (active)
    - sleeping (waiting for an event or waiting for a resource to become available)
    - zombie (this is a process that has completed execution but not yet fully exited the process list)
- » Information is provided in real time and is dynamically updated by the kernel
- » Can use filesystem tools to analyze
- » When power is lost, this ceases to exist and is not recovered in standard disk images
- » **/proc/kcore** is a snapshot of kernel and physical memory

Common Remote Collection and Analysis Tools

› Host-based enterprise DFIR agent software:
- » Provides connection from investigator to target host system
- » Can run processes on the target host and send data back to the investigator
- » Restricted by host system resources and network bandwidth

› Endpoint Detection and Response (EDR) Tools:
- » Uses a host-based agent
- » Most EDR products function as remote analysis tools run from a central management console.
- » Enables continuous monitoring of activity on endpoints
- » EDR solutions are not forensic tools
  - ▸ They rarely protect metadata, have constraints around artifacts they can access, and often allow live response, which generates new artifacts and potentially destroys evidence.
- » Needs knowledgeable and skilled analysts to drive value
- » It is rare to find EDR on Linux-based infrastructure
- » Alternatives to commercial EDR platforms include OSSEC and Velociraptor

› Sysmon for Linux:
- » Not yet widely deployed in commercial enterprises.
- » Requires deployment and maintenance of an agent on the endpoint
- » Logs it generates need to be collected and analyzed somewhere
- » Sysmon works by monitoring system activity (effectively API calls) and logging events based on user-defined rules
- » Sysmon for Linux logs events in a variety of formats, including JSON and Syslog
- » To install Sysmon on Linux, you need to get the Microsoft signing key and add that to your distro's package management tool, before updating the package manager
- » Sysmon is configured using a YAML format file that contains the rules to implement.
- » Confirm which event IDs are recorded by your version of Sysmon by running the **sysmon -s** command.

› Commercial Forensic Suites:
- » Full forensic suites use host-based agents
- » Can deliver incredible analytical capabilities but often need significant resources
- » Check that the distro in use is supported; capabilities vary

Common Triage Collection Tools

› C# Live Response (CyLR):
- » Open-source tool for rapid collection of triage data and forensic evidence
- » Runs from a pre-compiled binary that can be dropped in the target environment
- » Retains metadata
- » CyLR needs to be run as root on the target system
- » No longer under development

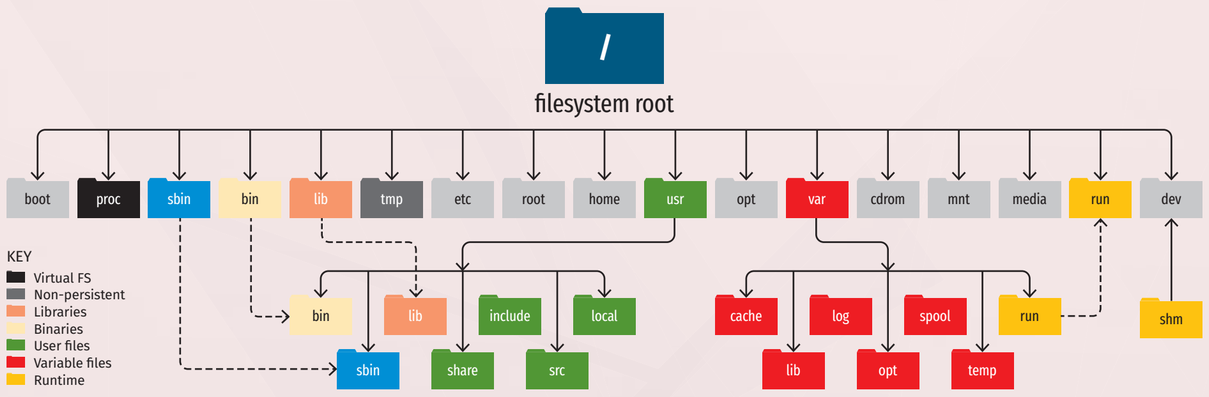
› Google Rapid Response (GRR):
- » Open-source IR framework from Google designed for forensics at scale
- » Uses a client-server model and provides a dashboard for monitoring
- » Can work well for a limited set of use cases
- » Recommended deployment of GRR is via Docker

› Unix-like Artifacts Collector (UAC):
- » Open-source tool to gather triage data from Unix-based systems, such as Linux, macOS, and FreeBSD
- » Uses binaries already on the host system, so compromised binaries may result in incorrect or omitted output
- » Respects the Order of Volatility
- » Can also capture RAM
- » Includes support to write output to various cloud platforms
- » Collect artifacts in a profile: **uac -p [collection profile] [output dir]**
- » Collect specific artifacts: **uac -a [artifacts] [output dir]**
- » Collect data from a mounted volume: **uac [-p|-a] [profile|artifacts] [output dir] --mount-point [mount point]**

› Velociraptor:
- » Open-source EDR-like tool with exceptional forensic collection capabilities.
- » It is easy to deploy and configurable
- » Triage collection can be achieved using an offline collector or via the stand-alone query mode

› Dissect:
- » Open-source forensic collection and analysis framework developed by Fox-IT.
- » It is easy to use and configurable.
- » Tool allows for collection (Acquire) and analysis (Dissect) of data
  - ▸ These two components need to be installed separately.
- » Running Acquire requires elevated privileges for full functionality and to read raw disk data
- » Default behavior of Acquire runs against the local host
- » Acquire a target: **acquire -p [profile] -o [output dir] [target]**
- » Extract information from evidence: **target-query -q -f [functions to run] [evidence]**
- » Run command on evidence: **target-fs -q [evidence] [command] [path to run command against]**
- » Mount evidence: **target-mount -q [evidence] [mount point]**
- » Open virtual shell environment on evidence: **target-shell -q [evidence]**

GENERAL LINUX FOLDER HIERARCHY



Some important locations for DFIR:

- **/bin and /sbin – binary executables**
- **/etc – configuration files**
- **/dev – device files**
- **/home – user home directories**
- **/lib* – libraries and system files**
- **/mnt and /media – mounted filesystems**
- **/opt – optional third-party software packages**
- **/root – root user's home directory**
- **/tmp – temporary files**
- **/usr – user-related data, including executables and libraries**
- **/var – log files, and temporary or persistent data for various services and applications**

MOUNTING AND UNMOUNTING IMAGES

Types of image file

- **Raw (often referred to as DD)**
  - » Often with an extension: .dd, .raw, .001, .img
  - » Easiest to mount; only requires the built-in **mount** command
- **E01**
  - » EnCase Expert Witness Format (EWF)
  - » Industry standard in forensic evidence image formats
  - » Files normally have an .E01 extension, but this can vary if the file is split
  - » Files need to be converted to raw format first—this can be achieved using **ewfmount**, which is not installed by default
- **VMDK**
  - » Virtual machine disk file
  - » Can be mounted with **vdfuse** or **guestmount**, but neither is installed by default
- **VHDX**
  - » Less commonly seen in Linux incidents
  - » Can be mounted using **guestmount**, which is not installed by default

EWF image files

- **Mounting an EWF image file with one partition:**
  - » **mkdir [dir to use as mount point for ewfmount command]**
  - » **ewfmount [E01 image file] [mount point for raw image]**
  - » **mkdir [dir to use as mount point for mount command]**
  - » **mount -o ro,norecovery [raw image file] [mount point]**
- **Mounting an EWF image file with multiple partitions:**
  - » **mkdir [dir to use as mount point for ewfmount command]**
  - » **ewfmount [E01 image file] [mount point for raw image]**
  - » **mkdir [dir to use as mount point for mount command]**
  - » **fdisk -l [raw image file]** OR **mmls [E01 image file]**
    - Identify the partition you want to mount and multiply the partition start location by the sector size to get the offset
  - » **mount -o ro,norecovery,offset=[offset] [raw image file] [mount point]**
- **Unmounting an EWF image file:**
  - » **umount [raw mount point]**
  - » **umount [EWF mount point]**
  - » **rmdir [raw mount point dir]**
  - » **rmdir [EWF mount point dir]**

Raw Image Files

- **Mounting a raw image file with one partition:**
  - » **mkdir [dir to use as mount point]**
  - » **mount -o ro [raw image file] [mount point dir]**
    - The **-o ro** option mounts the image in read-only mode.
- **Mounting a raw image file with multiple partitions:**
  - » **fdisk -l [raw image file]** OR **mmls [raw image file]**
    - Identify the partition you want to mount and multiply the partition start location by the sector size to get the offset.
  - » **mount -o ro,offset=[offset] [raw image file] [mount point dir]**
- **Unmounting a raw image file:**
  - » **umount [mount point]**
  - » **rmdir [mount point dir]**
- **Mounting an LVM/LVM2 partition:**
  - » **fdisk -l [raw image file]** OR **mmls [raw image file]**
    - Identify the partition you want to mount and multiply the partition start location by the sector size to get the offset.
  - » **losetup -rf -o [offset] [raw image file]**
  - » **losetup -a | grep [raw image file]**
    - Identify the loopback device address.
  - » **pvdisplay [loopback address]**
    - Identify the Volume Group Name (VG Name).
  - » **vgchange -a y [VG Name]**
    - You can, generally, ignore warnings about old PV headers.
  - » **lvscan | grep [VG Name]**
    - Validate the Volume Group is ACTIVE and identify root device to mount.
  - » **mkdir [dir to use as mount point]**
  - » **mount -o ro, norecovery, noexec [device to mount] [mount point dir]**
- **Unmounting an LVM/LVM2 partition:**
  - » ***umount [mount point]***
  - » ***vgchange -a n [VG Name]***
  - » ***losetup -d [loopback address]***
  - » ***rmdir [mount point dir]***

VMDK and VHDX files

- **Mounting a VMDK file:**
  - » **mkdir [dir to use as mount point]**
  - » **guestmount -a [VMDK file] -i --ro -o allow_other [mount point]**
    - -a Specifies the source image to be mounted
    - -i Uses inspector to attempt to detect the operating system and mount points automatically
    - --ro Mount the image as read-only.
    - -o Options
      - allow_other: Allow other accounts to see the mounted filesystem
- **Mounting a VHDX file:**
  - » **mkdir [dir to use as mount point]**
  - » **guestmount -a [VHDX file] -m [mount point in VHDX] --ro -o allow_other [mount point]**
    - -a Specifies the source image to be mounted
    - -i Uses inspector to attempt to detect the operating system and mount points automatically
    - -m Specifies the mount point within the VHDX file to mount
    - --ro Mount the image as read-only.
    - -o Options
      - allow_other: Allow other accounts to see the mounted filesystem.
- **Unmounting a VMDK or VHDX file:**
  - » **guestunmount [mountpoint]**

Mounting and Unmounting with qemu-nbd

- **Mounting an image file:**
  - » **modprobe nbd**
  - » **qemu-nbd -f [image file format] -c /dev/nbd0 [image file]**
  - » **mkdir [dir to use as mount point]**
  - » **mount /dev/nbd0p1 [mount point]**
- **Unmounting an image file:**
  - » **umount [mount point]**
  - » **qemu-nbd -d /dev/nbd0**
  - » **rmdir [mount point dir]**

HUNTING THROUGH ARTIFACTS

MODIFIED USER ACCOUNTS | STRANGE ENTRIES IN SHELL HISTORY | TEXT EDITOR HISTORY FILES | STRANGE NETWORKING | COMMON PERSISTENCE | ALTERED FILES | SUSPICIOUS LOG DATA

Account Modification

- **Signs of Attack:**
  - » Brute-force attacks
  - » Creating new accounts
  - » Adding accounts to groups to elevate permissions
  - » Adding SSH keys to the **authorized_keys** file
  - » Giving themselves sudo rights
- **Locations:**
  - » Authentication logs (auth.log, secure, utmp, wtmp, btmp, etc.): Check for large numbers of failed logins
  - » **/etc/passwd**: Check the user accounts, look for file last modification times, and check for users who have login shells first
    - Readable by all users
    - File structure: **[Username]:[Password info (x = stored in shadow)]:[User ID (UID)]:[Group ID (GID)]:[User information (comments)]:[Home folder]:[Login shell]**
  - » **/etc/shadow**: Same as the passwd file, look for any unexpected accounts and check the time the file was last modified. Check for unusual entries in the password field.
    - Only readable by root user
    - File structure: **[Username]:[Password info]:[Last password change]:[Minimum password age]:[Maximum password age]:[Password warning period]:[Password inactivity period]:[Account expiration date]:[Reserved field]**
      - The [Password info] field has the structure: **$[Hash algorithm code]$[Salt]$[Hash]**
      - If the [Password info] field contains an asterisk (*) or exclamation marks, the account is locked and cannot be used for password-based login.
      - The last three fields are rarely used.
  - » **/etc/group**: Check what groups exist and which accounts are in privileged groups (start with wheel, sudo, and adm). Check the last modification time.
  - » **/etc/sudoers**: Validate the last modification time. Check for users with excessive accounts, especially users who have ANY:ANY set with or without a password. The sudoers file is often overwritten by the system as part of updates/patches.
  - » **/etc/sudoers.d/***: As for the sudoers file, attackers prefer this location because it survives patching/system updates
  - » **/home/[username]/.ssh/** and **/root/.ssh/**
    - The **known_hosts** file stores a record of public keys for systems the user account has connected to.
    - The **authorized_keys** file contains public keys that are allowed to authenticate on this system.

Shell and Text Editor History

- **Signs of Attack:**
  - » Use of sudo/su
  - » Cleartext passwords
  - » Unusual command lines
  - » Locations:
  - » **.bash_history** or **.zsh_history** (shell history): Look for any unusual commands that have been run
    - The location of this file is stored in the $HISTFILE variable
    - Only written to disk on shell exit (if exit recognized by the OS)
    - Order of commands is not reliable
    - Trivial for an attacker to edit or remove
    - May be truncated based on the current HISTFILESIZE setting, with oldest entries removed first
  - » **.lesshst**: Look for unusual searches that were conducted
  - » **.viminfo**: Look for any unusual use of vi/vim
  - » **.mysql_history**: Look for any unusual mysql activity
  - » Any other history files such as **.python_history**, **.gdb_history**, or **.local/share/nano/search_history**

Common Persistence Methods

- **Signs of Attack:**
  - » New user accounts
  - » SSH access
  - » New cron jobs
  - » New startup scripts
  - » Modified binaries
  - » Hidden files
- **Locations:**
  - » **crontab**; **/etc/cron.d**; **/etc/cron.hourly**; **/etc/cron.daily**; **/etc/cron.weekly**; **/etc/cron.monthly**; and **/var/spool/crontabs** folders: Look for a last modification time within the incident window and unusual or unexpected entries
  - » Known locations for startup scripts for that distro (mostly in the /etc/ folders)
  - » Check for new startup scripts in:
    - **~/.bashrc**
    - **~/.bash_aliases**
    - **~/.profile** – user specific profile, the exact filename can vary between systems
    - **~/.bash_profile** – user specific profile, the exact filename can vary between systems
    - **~/.bash_login** – user specific profile, the exact filename can vary between systems
    - **~/.bash_logout**
    - **/etc/profile** – system-wide, global initialization file for login shells and called each time a user logs in
    - **/etc/rc.local** – system-wide boot script that is called as the system starts up
  - » Check for binaries modified during the incident window
  - » Check for any hidden files (preceded with ".")

Networking

- **Signs of Attack:**
  - » Unusual ports being used
  - » Long-lasting connections
  - » Unexpected processes using the network
- **Locations:**
  - » **/etc/hosts**: Check that hostnames point to legitimate targets and haven't been modified by an attacker
  - » **/etc/resolv.conf** and **/etc/system/resolved.conf**: Check that the nameserver settings are valid

Altered files

- **Signs of Attack:**
  - » Adding users
  - » Deleting history/anti-forensics
  - » Hiding files
  - » Staging data
  - » Leaving backdoors
- **Locations:**
  - » Look for large files, especially archives. This can often be a sign that attackers are staging data for exfiltration.
  - » **/dev** – Look for regular files in here; it should only hold devices or links
  - » Search for files with a modification time stamp that fits within the incident window. Remember that timestamps can be manipulated.

Log Files

- » Signs of Attack:
- » Failed attempts to use sudo
- » Root account changing key system settings
- » New accounts/cron jobs being created
- » Unusual log entries during the incident window
  - In general, the most effective way to work through logs is to follow the line of attack.
  - Log files are typically very noisy.
- » Maliciously modified log entries
- » Deleted log entries or entire log files
- » Attack to logging service causing log corruption
- **Locations:**
  - » **/var/log/**
  - » **/var/run/** (symlink)
  - » Log management tool (logrotate) configuration:
    - Configuration file : **/etc/logrotate.conf**
    - Individual configurations : **/etc/logrotate.d/***
    - Scheduled task: **/etc/cron.daily/logrotate**
    - Systemd timer: **/usr/lib/systemd/system/logrotate.***
    - State file (last rotation time):
      - Ubuntu: **/var/lib/logrotate/status**
      - RHEL: **/var/lib/logrotate.status**

ROOTKITS

Types of Rootkit

- **User-mode Rootkits:**
  - » Operate in user space, where regular applications run
  - » Typically hook application and system binaries to intercept and alter system calls
- **Kernel-mode Rootkits:**
  - » Most popular rootkits
  - » More powerful and stealthy than user-mode rootkits
  - » Operate within kernel space, giving direct access to hardware and kernel subsystems
  - » They often modify kernel data structures or load as malicious kernel modules
  - » Techniques include hiding their presence by tampering with kernel data structures such as linked lists and process tables
  - » The most commonly found attack is the abuse of Loadable Kernel Modules (LKMs)
    - Review Loadable Kernel Modules in the live kernel with the lsmod command. This command draws its information from the **/proc/modules** container.
- **Bootloader Rootkits:**
  - » Infect the bootloader, such as GRUB (GNU GRand Unified Bootloader), which allows launch on boot
  - » They can initiate before the Linux kernel loads, making detection very difficult and allowing them to manipulate the boot process
- **Hardware/Firmware Rootkits:**
  - » Target the firmware on devices
  - » These rootkits can persist through operating system reinstallations and have control from the earliest stages of the boot process
  - » Particularly challenging to detect and remove because they operate below the operating system level
- **Memory Rootkits:**
  - » Reside solely in the system's RAM
  - » Typically inject code into the running kernel and manipulate its execution while the system is running
  - » Harder to detect but will not persist after a reboot unless reinjected
- **Library Rootkits:**
  - » Target system libraries, replacing them or modifying their functions
  - » Can intercept and modify inputs and outputs of library functions, altering the behavior of applications using the libraries without altering the applications directly

Checking for rootkits

- **Check for signs of known Linux rootkits on a live system:**
  - » **/usr/sbin/chkrootkit [options] [test]**
  - » **rkhunter [options]**
- **Review Loadable Kernel Modules:**
  - » Does the file name look legitimate?—malicious LKMs are often single-character names or contain typos from legitimate modules
  - » Check the kernel log file (for example, dmesg) for anything unusual like tainting
  - » Where possible, compare known-good lists of loaded modules to the evidence
- **Review running processes for library attacks:**
  - » Check and validate the paths to the files are legitimate in the environment variables for the running process—this information is stored in **/proc/$pid/environ**
  - » Use **ldd** to review an executable on disk and identify which shared libraries it calls compared to the libraries in the running process—this information is stored in **/proc/[PID]/maps**
- **Review system calls.**
  - » Review the global system call table for anything unusual
  - » Use ***strace –p <PID>*** to connect a debugger to Process ID <PID> and validate what system calls each process is making
  - » Use ***strace <command>*** to run a new command to check its system calls
- **Check the audit logs, if configured**
- **Scan the environment with Yara rules, to look for known indicators**

LINUX LOG FILES

System Logs – Kernel Messages (dmesg)

- **Description**

The Kernel Messages log contains information related to the boot process of the system, generated by the kernel and device drivers. The name dmesg comes from "Diagnostic Messages." Entries in this log provide information about hardware and device initialization.

- **Location**
  - » **/var/log/dmesg**
  - » Also, on Ubuntu 22.04 and older: **/var/log/kern.log**
- **Interpretation**
  - » The dmesg file is written at the end of the system startup cycle
  - » The log is stored in a buffer in the kernel and can be viewed using the **dmesg** command on a live system
  - » RHEL8/CentOS8 (Fedora 28), and Ubuntu 24.04 or newer, no longer include this service by default; dmesg events are written to the system journal and can still be viewed with the dmesg command, but there is no /var/log/dmesg file
  - » Includes entries relating to USB devices or drive activity

System Logs – Boot History

- **Description**

Linux distros maintain a record of scripts run and events that fire during the boot process, in log files. This therefore logs the last time the system was booted.

- **Location**
  - » **/var/log/boot.log***
- **Interpretation**
  - » Plaintext log file
  - » This may be disabled in some distros; on newer systems, it is recommended to use the journal for boot-time data
  - » The log may not have timestamps on each line; there is sometimes only a timestamp per file
  - » Check the log for services that were started, to identify signs of attacker persistence or malicious activity

System Logs – Global System Log

- **Description**

This log is used to store various system messages, including information about system startup and shutdown, system services and drivers, network events, and hardware devices.

- **Location**
  - » Ubuntu: **/var/log/syslog**
  - » RHEL: **/var/log/messages**
- **Interpretation**
  - » Plaintext log file
  - » Typically very large in size; use tools like **grep** to search for key strings
  - » The log file is typically rotated regularly, and older log data may be archived or deleted
  - » The size and rotation schedule of the logs can be configured by the system administrator, normally with **logrotate**
  - » Can contain similar information to **/var/log/boot.log**; check both files

System Logs – Running Services

- **Description**

When available, the daemon.log (controlled by the syslog facility) provides a record of events and messages generated by system daemons. The log typically contains messages the status of services, configuration changes, and error messages. This can provide valuable information for troubleshooting and debugging purposes.

- **Location**
  - » **/var/log/daemon.log**
- **Interpretation**
  - » Plaintext log file
  - » Typically not recorded by default; a lot of this data is now written to the Journal or Syslog
    - To check if this log is enabled, look for the following line in **rsyslog.conf**:
      - daemon.* /var/log/daemon.log
  - » Some systems may use different names or locations, depending on the logging system in use

Authentication Logs – User Logins

- **Description**

User login data is recorded in three binary log files. Data captured includes username, login terminal, and login/logoff times.

- **Location**
  - » Users/sessions currently logged in: **/var/run/utmp**
  - » Historical utmp data: **/var/log/wtmp**
  - » Failed login events: **/var/log/btmp**
- **Interpretation**
  - » ***utmp*** is only available on a live system and may not be available depending on how you have captured the data
  - » Not all systems will record a btmp file
  - » All three files are binary "database" files and need to be read with commands such as:
    - **last -F -f [evidence utmp/wtmp/btmp file]**
    - **lastb -F -f [evidence btmp file]**
    - **utmpdump [evidence wtmp file]**
  - » Note that **lastlog** and **faillog** are unreliable

Application Logs – Web Server Logs

- **Description**

Web servers are the backbone of the Internet and are essential for serving and delivering web content to users. This is one of the most common uses for Linux platforms globally. Two of the most popular applications are Nginx and Apache, and both store access logs that record information about requests made to the server. The error log records information about errors and problems that occur on the server.

- **Location**
  - » Nginx: **/var/log/nginx/**
  - » Apache HTTP Server: **/var/log/apache2/**
  - » RHEL/CentOS: **/var/log/httpd/**
- **Interpretation**
  - » Access logs can include the client's IP address, the date and time of the request, the request method, the requested resource, the HTTP status code, and the size of the response
  - » Both the Nginx and Apache default access log format is the Combined Log Format
  - » Older Apache servers have the default access log format of the Common Log Format
  - » Almost all commercial or enterprise web servers will use either the Common Log Format or the Combined Log Format for access logs
  - » By default, the error log is stored in the same folder as the access log

System Logs – The Journal

- **Description**

The journal provides an audit trail of activity on the system by logging various types of events, including system startup and shutdown, kernel messages, and system service and application events. It stores metadata, such as the hostname, system architecture, and boot ID, allowing responders to correlate activity across systems and time.

- **Location**
  - » **/var/log/journal/[UUID]/*.log**
- **Interpretation**
  - » Logs are stored in a binary format
  - » In live response, the journal can be analyzed using the **journalctl** tool or programmatically via the **systemd-journal** API
  - » There are a lot of command line arguments and options available, some of the more useful ones for DFIR are:
    - -S or --since= Show entries on or newer than the date specified (yyyy-mm-dd hh:mm:ss)
    - -U or --until= Show entries on or older than the date specified
    - -b or --boot= Show messages from a specific boot session
    - -u Show messages for the specified systemd unit
    - -t Show messages for the specified syslog identifier
    - _COMM= Match the script name
    - _EXE= Match the executable name
    - -g Pattern match (grep) for the specified regular expression
    - -k or --dmesg Show only kernel messages
  - » The journalctl command can also analyze offline journal files. Example commands:
    - Analyzing a single file: **journalctl --file [journal file]**
    - Analyzing a folder of journal records: **journalctl --directory [journal dir]**
    - Analyzing a mounted filesystem: **journalctl --root /mnt/evidence1**
    - Analyzing a raw disk image: **journalctl --image /dev/loop22**
  - » The journal subfolder name will correspond to the unique machine identifier for the relevant device. Confirm the correct machine identifier by reading the text file at **/etc/machine-id**.

Authentication Logs – Authorization Session Logs

- **Description**

Account creation, user logins from external services and privilege use such as using sudo are recorded in an authorization log. The log location differs depending on the distro.

- **Location**
  - » Ubuntu: **/var/log/auth.log**
  - » RHEL: **/var/log/secure**
- **Interpretation**
  - » Plaintext log file

Application Logs – SSL/TLS Logs

- **Description**

Information about incoming Secure Sockets Layer (SSL) encrypted traffic or traffic using Transport Layer Security (TLS) to a web server, is logged in three different log files.

- **Location**
  - » **/var/log/[Application]/ssl_access_log**
  - » **/var/log/[Application]/ssl_request_log**
  - » **/var/log/[Application]/ssl_error_log**
- **Interpretation**
  - » The ***ssl_access_log*** is used to track connections and for administrators to monitor the performance and security of the server and troubleshoot issues. This log can include:
    - Date and time of the connection
    - IP address of the client making the connection
    - Certificate information presented by the client
    - Results of the SSL handshake process
    - Encryption algorithm and key size used for the connection
    - SSL protocol version used
  - » The ***ssl_request_log*** provides a more detailed record of SSL-encrypted connections. This log can include:
    - SSL requests made by the client
    - Contents of the SSL request
    - Results of processing the request
  - » The ***ssl_error_log*** tracks errors related to SSL encryption. It records information about any issues that occur during the SSL handshake process or that prevent the establishment of a secure SSL connection.

Application Logs – MySQL

- **Description**

MySQL is a widely used open-source relational database management system. It also includes a comprehensive logging system, which by default involves writing to various log files, including the error log, the slow query log, and the general query log. MySQL allows configuration of the level of detail recorded in logs, although the default is basic logging and it may write to unexpected locations.

- **Location**

Typically, logging and log locations are defined by the configuration file at:

- » **/etc/mysql/my.cnf**
  - Or a subfolder of this folder
- **Interpretation**
  - » Check the configuration file to locate logs and determine current configuration
  - » Log files are plaintext, so easily readable with a text editor or using the **cat** command

Application Logs – PostgreSQL

- **Description**

PostgreSQL is an open-source relational database management system that provides advanced features like transaction management, user authentication, and data integrity. PostgreSQL has a comprehensive logging system and can be configured to ship logs to the standard system log, as well as configure the level of detail that is recorded. However, the default logging options are often insufficient for security purposes and require deliberate effort to turn on.

- **Location**

Typically, logging and log locations are defined by the configuration file at:

- » **/var/lib/pgsql/data/postgresql.conf**
- **Interpretation**
  - » Check the configuration file to locate logs and determine current configuration
  - » Log files are plaintext, so easily readable with a text editor or using the ***cat*** command

Application Logs – vsftpd

- **Description**

vsftpd is one of the more commonly found enterprise FTP clients on Linux systems, which unlike most other FTP packages, provides a secure method for transferring files over a network. It is configurable and provides robust logging capabilities.

- **Location**

Typically, logging is defined by the configuration file at:

- » **/etc/vsftpd.conf** or **/etc/vsftpd/vsftpd.conf**

Log location:

- » **/var/log/vsftpd.log**
- **Interpretation**
  - » The location of the configuration file on a live system can be established with the **systemctl statud vsftpd** command
  - » By default, vsftpd logs events to the syslog facility
  - » By default only events of notice level or higher are logged

Application Logs – Samba (SMB)

- **Description**

Samba uses the SMB protocol to allow Linux/Unix servers to share files, printers, and other resources with Windows-based clients. Samba logs important events in its operation, including user authentication attempts and server status changes.

- **Location**

Typically, logging is defined by the configuration file at:

- » **/etc/samba/smb.conf**

Log location:

- » **/var/log/samba/log.[hostname|IPaddress]**
- **Interpretation**
  - » Samba can act as a client or a server
  - » The default logging configuration is basic and largely set to support troubleshooting
  - » Log files are plaintext, so easily readable with a text editor or using the **cat** command

Firewall Logs – iptables

- **Description**

The most basic host firewall in Linux is iptables. It provides a way to control incoming and outgoing network traffic based on a set of rules. If the LOG option is used, information on packets matching the rule will be logged to the kernel message log.

- **Location**
  - » System Logs – Kernel Messages (dmesg)
- **Interpretation**
  - » On a live system, iptables logs in the kernel message log can be viewed using one of the commands:
    - **dmesg**
    - **journalctl -t iptables**
  - » See the Kernel Messages (dmesg) section for more information on these logs

Firewall Logs – The Uncomplicated Firewall (UFW)

- **Description**

UFW is a user-friendly front-end for iptables and a default firewall application in many Linux distributions, including Ubuntu. UFW is a default deny firewall, meaning it blocks all incoming traffic by default. Rules are then added to allow specific types of traffic.

- **Location**
  - » **/var/log/ufw.log**
- **Interpretation**
  - » Logging is not turned on by default and must be enabled by the root account using **ufw logging on**
  - » Log data generated by UFW (and iptables) can be noisy
  - » Log files are plaintext, so easily readable with a text editor or using the **cat** command

Firewall Logs – firewalld

- **Description**

firewalld is a common firewall management tool that is the default on RHEL/CentOS7 or later versions. It controls network traffic based on the rules specified in the firewall configuration, which can be changed dynamically without interrupting the network connections. The firewall also logs any events that occur if the "log" option is used.

- **Location**
  - » **/var/log/firewalld**
- **Interpretation**
  - » firewalld does not log dropped packets by default; the "log" option is required
  - » The default configuration is minimal logging
  - » Logs can be very verbose, and therefore difficult to read
  - » In mixed-mode, firewalld writes info-level log messages to syslog
  - » Debug messages are written to the firewalld log file by default, but this can be configured
  - » Info messages also go to stdout and stderr

Audit Logs – auditd

- **Description**

auditd is a Linux utility for auditing system events, including user activity, system calls, and other events that can have a security impact. It operates at the kernel level, collecting data from the system and storing it in a binary format.

- **Location**

Typically, logging and the log location is defined by the configuration file at:

- » **/etc/audit/auditd.conf**

Log location:

- » **/var/log/audit/audit.log**
- **Interpretation**
  - » auditd is not installed by default on Debian-family distros
  - » Even where auditd is installed, the default settings are insufficient to provide much assistance to an investigation
  - » Logs generated by auditd can be very verbose
  - » Generate human-readable summaries of audit activity using the ***aureport*** tool
  - » Search audit logs using the **ausearch** tool
  - » Both **aureport** and **ausearch** query the live log on the running system by default. However, this can be overridden by specifying an input file with the ***-if*** argument

DEVICE PROFILING

Computer Name

- **Description**

The computer name, or hostname, is stored in a plaintext file in the /etc folder.

- **Location**
  - » **/etc/hostname**
- **Interpretation**
  - » This file can be read with any text editor or via the ***cat*** command
  - » Live response commands to identify the hostname:
    - **hostname**
    - **hostname --fqdn**
    - **hostname --long**

Hosts File

- **Description**

The hosts file is used to map IP addresses with hostnames and takes precedence over a DNS lookup.

- **Location**
  - » **/etc/hosts**
- **Interpretation**
  - » This file can be read with any text editor or via the ***cat*** command
  - » Check the hosts file for anything that may have been added/changed by an attacker, or if it provides any clues or visibility into the environment

Timezone

- **Description**

Only the device's current timezone setting is stored in these files.

- **Location**
  - » **/etc/timezone**
  - » **/etc/localtime** (symbolic link)
- **Interpretation**
  - » **timezone** can be read with any text editor or via the ***cat*** command.
  - » **localtime** can be read using the ***ls -l*** command.

Distro Information

- **Description**

The Linux distro name and version number are often recorded in multiple ways and formats, depending on the distro.

- **Location**
  - » **/etc/os-release**

On Ubuntu (and Debian-based distros):

- » **/etc/lsb-release**

On RHEL/CentOS-based distros:

- » For RHEL systems: **/etc/redhat-release**
- » For Fedora systems: **/etc/fedora-release**
- » For CentOS systems: **/etc/centos-release**
- » For Rocky Linux systems: **/etc/rocky-release**
- » For Amazon Linux systems: **/etc/system-release**
- » For Oracle Linux systems: **/etc/oracle-release** and **/etc/redhat-release**
- » For SLES systems: **/etc/SuSE-release** (older) or **/etc/SUSE-brand** (newer)
- **Interpretation**
  - » This file can be read with any text editor or via the ***cat*** command.
  - » The key fields in **os-release** are:
    - **NAME:** The distro name, which is defined by the creator/release organization
    - **VERSION:** Version number and name, if appropriate
    - **ID:** Also the distro name, but often in an abbreviated form
    - **ID_LIKE:** The distro family
    - **PRETTY_NAME:** A human-readable string describing the distro and version number
    - **VERSION_ID:** A shortened representation of the version string
  - » The key fields in ***lsb-release*** are:
    - **DISTRIB_ID:** The distro name
    - **DISTRIB_RELEASE:** The release or version number of the distro
    - **DISTRIB_CODENAME:** The codename associated with the distro release
    - **DISTRIB_DESCRIPTION:** A human-readable description of the distro, combining the name, version, and codename
  - » RHEL/CentOS-based distro files contain a string with the distro name and version number
  - » Live response commands to identify the distro information:
    - **lsb_release -a**
      - Note: This is not available on default Fedora or RHEL/CentOS 7+ distros.
    - **hostnamectl**
    - On remote systems (connecting via SSH): **hostnamectl -H [user]@[IP address]**
    - **cat /proc/version**

Partition Information

- **Description**

Information about partitions currently present on the system's block devices, such as hard drives and flash drives, is available under the /proc virtual filesystem. This only exists while the system is live.

- **Location**
  - » **/proc/partitions**
- **Interpretation**
  - » Plaintext file; view using the ***cat*** command.

Mount Points

- **Description**

Information about the filesystems currently mounted in the live system, including their device, mount point, filesystem type, mount options, and other details, is present in a virtual file. This only exists while the system is live.

- **Location**
  - » **/proc/mounts**
- **Interpretation**
  - » Plaintext file; view using the ***cat*** command.
  - » Does not provide a historical record for previously mounted filesystems.