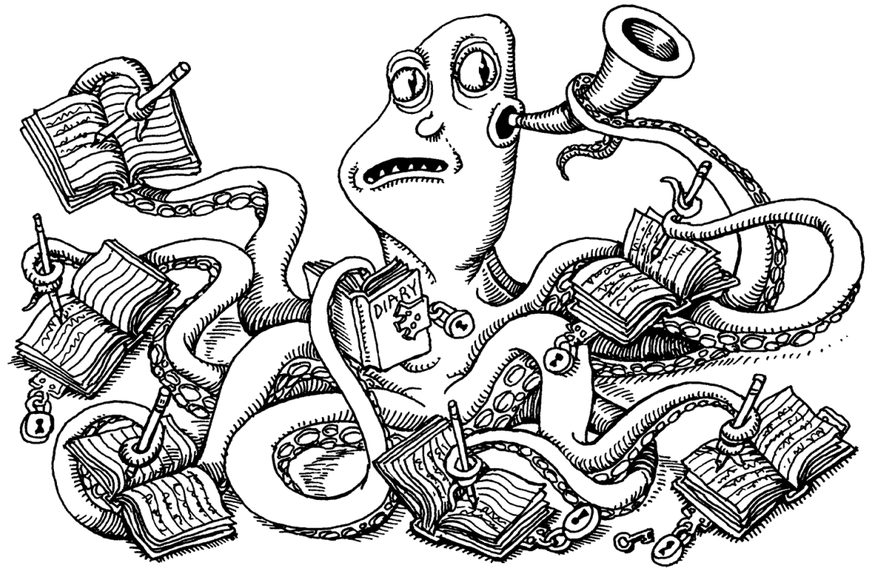
Infrastructure Security Using Linux

Computer science / Cybersecurity

Logging
Drivers and the Kernel

10 - 11

Logging



Logging

- **Log Management Overview**
  - Logs are generated by daemons, the kernel, and custom applications.
  - Over time, logs must be summarized, filtered, analyzed, compressed, archived, and ultimately discarded.
  - Access and audit logs often require strict handling due to regulatory requirements.
- **Key Log Management Tasks**
  1. Collecting logs from multiple sources.
  2. Providing tools for structured querying, filtering, and monitoring.
  3. Handling log retention, rotation, and compression.

Logging

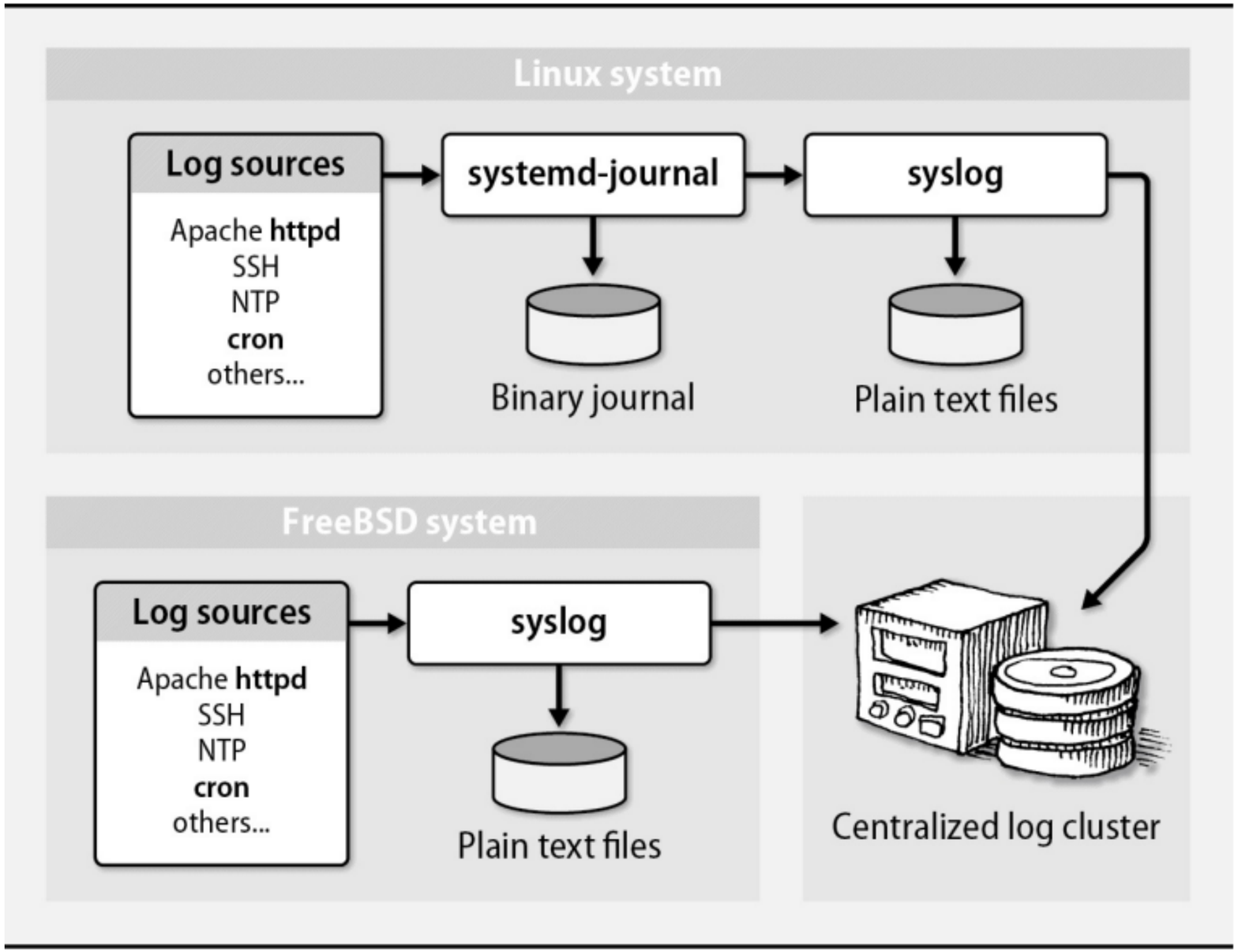
- **Syslog**
  - Traditional UNIX log management system.
  - Sorts messages and saves them to files or forwards them to remote hosts.
  - Lacks advanced querying, filtering, and analysis features.
  - Configuration varies significantly between operating systems.
- **Systemd Journal**
  - Linux’s attempt to improve logging capabilities.
  - Binary log format indexed for fast and structured queries.
  - Offers a robust set of tools for querying and monitoring logs.
  - Supports remote forwarding, rotation, and compression.

Logging architecture for a site with centralized logging

The Growing Need for Logging Strategies

- Formal IT standards (e.g., ***PCI DSS***, ***COBIT***, ***ISO 27001***) have increased the focus on consistent, well-structured logging.

- Industry-specific regulations have further emphasized the importance of site-wide logging practices.



Log locations

- **Log File Ownership and Growth**
  - ***Root*** typically owns log files, though conventions vary.
  - Logs for high-activity services (web, database, DNS) can grow rapidly, potentially filling disks and disrupting the system.
- **Best Practice**
  - Place ***/var/log*** on a separate partition or filesystem.
    - Helps prevent log growth from affecting other system operations.
    - Applies equally to physical servers, cloud instances, and private virtual machines.

File	Program	Where[a]	Freq[a]	Systems[a]	Contents
apache2/*	**httpd**	F	D	D	Apache HTTP server logs (v2)
apt*	APT	F	M	D	Aptitude package installations
auth.log	**sudo**, etc.[b]	S	M	DF	Authorizations
boot.log	**rc** scripts	F	M	R	Output from system startup scripts
cloud-init.log	**cloud-init**	F	–	–	Output from cloud init scripts
cron, cron/log	**cron**	S	W	RF	**cron** executions and errors
daemon.log	various	S	W	D*	All daemon facility messages
debug*	various	S	D	F,D*	Debugging output
dmesg	kernel	H	–	all	Dump of kernel message buffer
dpkg.log	**dpkg**	F	M	D	Package management log
faillog[c]	**login**	H	W	D*	Failed login attempts
httpd/*	**httpd**	F	D	R	Apache HTTP server logs
kern.log	kernel	S	W	D	All kern facility messages
lastlog	**login**	H	–	R	Last login time per user (binary)
mail*	mail-related	S	W	RF	All mail facility messages
messages	various	S	W	R	The main system log file
samba/*	**smbd**, etc.	F	W	–	Samba (Windows/SMB file sharing)
secure	**sshd**, etc.[b]	S	M	R	Private authorization messages
syslog*	various	S	W	D	The main system log file
wtmp	**login**	H	M	RD	Login records (binary)
xen/*	Xen	F	1m	RD	Xen virtual machine information
Xorg.*n*.log	**Xorg**	F	W	R	X Windows server errors
yum.log	**yum**	F	M	R	Package management log

a. Where: F = Configuration file, H = Hardwired, S = Syslog
Frequency: D = Daily, M = Monthly, NNm = Size-based (in MB, e.g., 1m), W = Weekly
Systems: D = Debian and Ubuntu (D* = Debian only), R = Red Hat and CentOS, F = FreeBSD
b. **passwd**, **sshd**, **login**, and **shutdown** also write to the authorization log.
c. Binary file that must be read with the **faillog** utility

Files Not to Manage

- **wtmp**:
  - Binary file tracking user logins, logouts, and system events.
  - Accessed via the ***last*** command for login histories.
- **lastlog**:
  - Records only the last login time per user.
  - Size remains constant, so rotation isn’t needed.
- **Binary transaction logs** (e.g., database logs):
  - Should not be manually managed.
  - Attempting to view them can disrupt your terminal.

Viewing Logs in the Systemd Journal

- Use ***journalctl*** to print messages from the systemd journal.
- For example, to see SSH daemon logs:

  journalctl -u sshd
- To follow logs in real time (similar to ***tail -f***):

  journalctl -f -u sshd
- ***-u*** option: Specifies the unit name (e.g., ***sshd*** corresponds to ***sshd.service***).

The systemd journal

- **The systemd journal**
  - systemd includes its own logging daemon: ***systemd-journald***.
  - It can duplicate syslog's functions but also works alongside syslog if needed.
  - Unlike syslog's plain text logs, the journal uses a ***binary format***.
    - Indexed message attributes make searching faster and more efficient.
- **Message Sources**
  - ***/dev/log***: Harvests messages from software that follows syslog conventions.
  - ***/dev/kmsg***: Collects messages from the Linux kernel.
  - ***/run/systemd/journal/stdout***: Gathers messages written to standard output.
  - ***/run/systemd/journal/socket***: Accepts messages submitted via the systemd journal API.
  - ***Audit messages***: Captured from the kernel's ***auditd*** daemon.

Filtering Options in journalctl

- **By Time**:

  journalctl --since=yesterday --until=now

  - Displays messages from yesterday at midnight until the current time.
- **By Priority**:
  - Use priority flags (e.g., ***-p err*** to show only error messages).
- **By Unit**:

  journalctl -u sshd

  - Displays messages from the SSH daemon.
- **By User or Content**:
  - Filter messages by the user who submitted them or by specific keywords.

Additional Filtering Commands

- **Disk Usage**:
  journalctl --disk-usage
  - Displays how much disk space is being used by the journal.
- **List Boots**:
  journalctl --list-boots
  - Shows a list of past system boots with numerical identifiers.
  - Example:
    journalctl -b -1
    - Displays messages from the previous boot session.
- **Number of Entries**:
  journalctl -n 100 /usr/bin/sshd
  - Shows the 100 most recent entries from a specific binary.

Syslog (Rsyslog Architecture)

- **Configuration**:
  - Rsyslog's primary configuration file is located at ***etc/rsyslog.conf***.
- **Default Operation**:
  - The ***rsyslogd*** process typically starts at boot and runs continuously.
  - Programs that are syslog-aware send messages through the UNIX domain socket ***dev/log***.
  - For non-systemd systems, rsyslogd reads messages from ***dev/log***, checks the configuration file for routing instructions, and sends each message to its configured destination.
- **Applying Changes**:
  - After modifying ***etc/rsyslog.conf*** or included files, restart the rsyslogd daemon.
  - **Signals**:
    - ***TERM signal***: Stops the daemon. Ex. kill -TERM $(pidof rsylogd)
    - ***HUP signal***: Closes and reopens log files, useful for log rotation.

Examples of commands

- System Log (general messages)
  - **cat /var/log/syslog**
  - **less /var/log/syslog**
  - **tail -f /var/log/syslog**

Management of Logs at Scale

Managing logs across hundreds or thousands of servers requires specialized tools that support centralized collection, storage, and analysis.

- **ELK Stack** (Elasticsearch, Logstash, Kibana)
  - ***Elasticsearch***: Distributed search engine for indexing and querying log data.
  - ***Logstash***: Log pipeline that collects, parses, and enriches logs.
  - ***Kibana***: Web interface for visualizing and exploring log data.
- **Graylog**
  - Alternative to the ELK stack with a similar architecture.
  - Uses ***Elasticsearch*** for storage.
  - Accepts logs directly or via ***Logstash***.
  - Lightweight and easier to manage for some use cases.

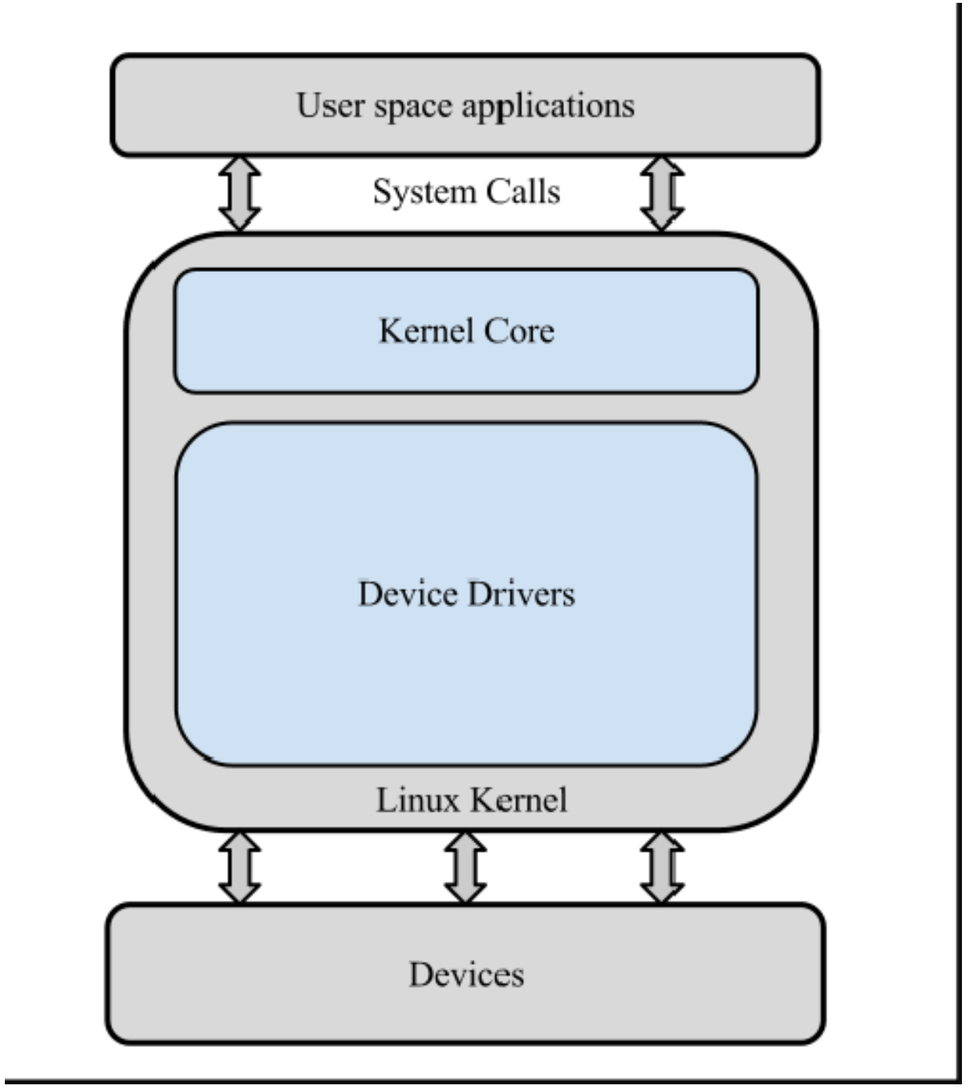
Management of Logs at Scale

- **Logging as a Service** (***Splunk)***:
  - Mature, enterprise-grade solution.
  - Available in both hosted and on-premises versions.
  - Supports advanced analytics beyond log management.
  - **Costly**, but widely used in large-scale corporate environments.

The Kernel and Drivers

The Kernel

- Acts as the **central authority** of a UNIX or Linux system.
- Enforces rules, manages resources, and provides essential services for user processes.
- Abstracts hardware complexity behind a clean, high-level interface—similar to an API for developers.



Core Kernel Features

1. ***Device management and abstraction***
2. ***Process and thread control***, including inter-process communication
3. ***Memory management***: virtual memory, isolation, and protection
4. ***I/O facilities***: filesystems, network, and serial interfaces
5. ***Housekeeping***: startup, shutdown, timers, multitasking, etc.

- **Device Drivers**
  - Only drivers know the hardware's specific protocols and capabilities.
  - The rest of the system remains hardware-agnostic.
- **Example**
  - A disk-based filesystem and a network-based one behave differently at the hardware level, but the kernel's ***VFS (Virtual File System)*** layer presents a unified interface to user processes and internal components.

Linux Kernel Versions

- Check the running kernel with: ***uname -r***
- Follows ***semantic versioning***:
  - ***Major version***
  - ***Minor version***
  - ***Patch level***
- **Stability**
  - Version numbers do **not** reliably indicate stability.
  - A kernel is considered stable only when the developers declare it so.