System and Network Engineering - Lecture 10

$ Logging and auditing the system

Logging: /var/log/

- historically, Linux has a special directory for storing logs called `/var/log`
- It contains logs from the OS, services, and various applications running on the system.

Most of the log can be splitted as:

System logs: syslog, messages(Redhat), dmesg
Application logs: apache/httpd, nginx, vbox
Event Logs: auth.log, boot.log
Service Logs: cron

Typical structure for system logs includes:

- timestamp
- hostname
- service name (facility)
- message

```
saltanov@UbuntuPC:/var/log$ tail -n 20 syslog
Oct 30 03:22:03 UbuntuPC NetworkManager[664]: <info>  [1667085723.4998] dhcp4 (enp0s8): canceled DHCP transaction
Oct 30 03:22:03 UbuntuPC NetworkManager[664]: <info>  [1667085723.5005] policy: auto-activating connection 'Wired conn
ection 2' (c6cccb8e-b5f7-3a4a-8feb-da6dc9075a04)
Oct 30 03:22:03 UbuntuPC NetworkManager[664]: <info>  [1667085723.5007] device (enp0s8): Activation: starting connecti
on 'Wired connection 2' (c6cccb8e-b5f7-3a4a-8feb-da6dc9075a04)
Oct 30 03:22:03 UbuntuPC NetworkManager[664]: <info>  [1667085723.5008] device (enp0s8): state change: disconnected ->
 prepare (reason 'none', sys-iface-state: 'managed')
```

/var/log

- The following table describes some of the log files that are located under /var/log/ directory.
- Some of these log files are distribution specific and may not be presented on some systems

/var/log/messages or /var/lor/syslog	This file has all the global system messages located inside, including the messages that are logged during system startup. Depending on how the syslog config file is sent up, there are several things that are logged in this file including mail, cron, daemon, kern, auth, etc.
/var/log/dmesg	Contains kernel ring buffer. This file is overwritten when the system is rebooted.
/var/log/auth.log	System authorization information is included in this file, along with user logins and the authentication mechanism that were used.
/var/log/daemon.log	The various system background daemons that are running will log information to this file.
/var/log/kern.log	Contains information logged by the kernel. Helpful to troubleshoot a custom-built kernel.
/var/log/lastlog	Displays the recent login information for all the users. This is not an ascii file. An admin can use the lastlog command to view the content of this file.
/var/log/maillog	Logs information from the mail server that is running on the system. For example, sendmail logs information about all the sent items to this file.
/var/log/cron	Whenever the Cron daemon starts executing a program, it logs messages in this file.
/var/log/secure	Contains information related to authentication and authorization. For example, SSHd collects everything here, including failed login attempts.

Examples: /var/log/

Auth.log:

- Keep authentication logs for both successful or failed logins, and authentication processes (such as switching users, using sudo etc)
- Storage depends on system type. For Debian/Ubuntu, look in /var/log/auth.log. For Redhat/CentOS, go to /var/log/secure.

```
saltanov@UbuntuPC:/var/log$ tail -10 auth.log
Oct 30 03:32:46 UbuntuPC pkexec: pam_unix(polkit-1:session): session opened for user root(uid=0) by (uid=1000)
Oct 30 03:32:46 UbuntuPC pkexec[81415]: saltanov: Executing command [USER=root] [TTY=unknown] [CWD=/home/saltanov] [CO
MMAND=/usr/lib/update-notifier/package-system-locked]
Oct 30 04:17:01 UbuntuPC CRON[81658]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
Oct 30 04:17:01 UbuntuPC CRON[81658]: pam_unix(cron:session): session closed for user root
Oct 30 04:38:57 UbuntuPC sudo: saltanov : TTY=pts/0 ; PWD=/var/log ; USER=root ; COMMAND=/usr/bin/su
Oct 30 04:38:57 UbuntuPC sudo: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
Oct 30 04:38:57 UbuntuPC su: (to root) root on pts/2
Oct 30 04:38:57 UbuntuPC su: pam_unix(su:session): session opened for user root(uid=0) by saltanov(uid=0)
Oct 30 04:39:17 UbuntuPC su: pam_unix(su:session): session closed for user root
Oct 30 04:39:17 UbuntuPC sudo: pam_unix(sudo:session): session closed for user root
```

lastlog:

- `/var/log/lastlog`: holds every user's last login. A binary file you can read via `lastlog` command. (only works when logged into login tty, Gnome based login is not shown.

```
gnome-initial-setup                          **Never logged in**
hplip                                        **Never logged in**
gdm                                          **Never logged in**
saltanov         tty1                        Sun Oct 30 05:43:13 +0400 2022
vboxadd                                      **Never logged in**
sshd                                         **Never logged in**
test             tty1                        Sun Oct 30 05:42:59 +0400 2022
```

Login, logouts, failure attempts, tty

There are 3 files that records relevant information:

- **/var/run/utmp** will give you complete picture of current users logins at which terminals, logouts, system events and current status of the system, system boot time (used by uptime) etc.
  - $w and $who - uses utmp to provide relevant data
- **/var/log/wtmp** gives historical data of utmp
  - $last - uses wtmp by default
- **/var/log/btmp** records only failed login attempts
  - $lastb - uses btmp

$last can be used to parse all above log files:

- $last -f /var/log/wtmp
- $last -f /var/run/utmp
- $last -f /var/log/btmp

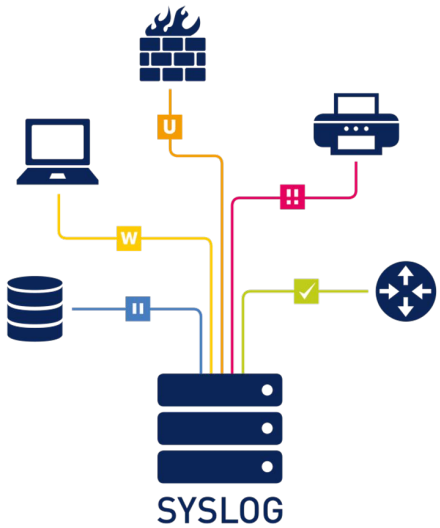
Same can be done with $utmpdump but in diff format:

- $utmpdump /var/log/wtmp
- $utmpdump /var/run/utmp
- $utmpdump /var/log/btmp

Syslog

Syslog is a standard for creating and transmitting logs. In general, "syslog" can refer to several things:

- The Syslog daemon, which receives and processes Syslog messages. It listens for events by creating a socket located at `/dev/log`, where applications can write to. Syslog can write messages to a local file or forward messages to a remote server. There are different Syslog implementations including **rsyslog** and **syslog-ng**.
- The Syslog protocol (RFC 5424), which is a transport protocol that specifies how to transmit logs over a network. It is also a data format defining how messages are structured. By default, it uses port **514** for plaintext messages and port **6514** for encrypted messages.
- A Syslog message - which is any log formatted in the Syslog message format. A Syslog message consists of a standardized header and message containing the log's contents. Typical message structure includes (timestamp, hostname, service name (facility), message)




```
saltanov@UbuntuPC:~$ ps -aux | grep syslog
message+     672  0.0  0.0  11092  6624 ?        Ss   18:04   0:00 @dbus-daemon --system --address=system
d: --nofork --nopidfile --systemd-activation --syslog-only
syslog       684  0.0  0.0 222400  5496 ?        Ssl  18:04   0:00 /usr/sbin/rsyslogd -n -iNONE
```

rsyslog

rsyslogd is a rsyslog daemon for receiving and storing logs. rsyslog configuration files:

- `/etc/rsyslog.conf` - main config file
- `/etc/rsyslog.d/*.conf` - additional configs can be included into the main

rsyslog configuration can contain the following sections:

- **Modules** - allows to add additional functional
- **Global directives** - set some global properties of whole rsyslog daemon, for example size of main message queue ($MainMessageQueueSize), loading external modules ($ModLoad) and so on. All global directives need to be specified on a line by their own and must start with a dollar-sign.
- **Templates** - allow to specify format of the logged message. They are also used for dynamic file name generation. They must be defined before they are used in rules.
- **Output channels** - provide an umbrella for any type of output that the user might want. They must be defined before they are used in rules.
- **Rules** - every rule line consists of two fields, a selector field and an action field. These two fields are separated by one or more spaces or tabs. The selector field specifies a pattern of facilities and priorities belonging to the specified action.

rsyslog

Rule structure:

- **Selector**
  - facility (type of program)
  - priority - severity (from emergency to debug)
- **Action**

The selector field consists of two parts:

- a facility and a priority, separated by a period ('.')
- The priority defines the severity of the message.

Value	Severity	Keyword
0	Emergency	`emerg`
1	Alert	`alert`
2	Critical	`crit`
3	Error	`err`
4	Warning	`warning`
5	Notice	`notice`
6	Informational	`info`
7	Debug	`debug`

The facility specifies the subsystem produced the message:

Facility code	Keyword	Description
0	kern	Kernel messages
1	user	User-level messages
2	mail	Mail system
3	daemon	System daemons
4	auth	Security/authentication messages
5	syslog	Messages generated internally by syslogd
6	lpr	Line printer subsystem
7	news	Network news subsystem
8	uucp	UUCP subsystem
9	cron	Clock daemon
10	authpriv	Security/authentication messages
11	ftp	FTP daemon
12	ntp	NTP subsystem
13	security	Log audit
14	console	Log alert
15	solaris-cron	Scheduling daemon
16–23	local0 – local7	Locally used facilities

rsyslog

rsyslogd configuration also allows the following extensions:

- an asterisk ('*') stands for all facilities or all priorities, depending on where it is used (before or after the period).
- the keyword none stands for no priority of the given facility.
- you can specify multiple facilities with the same priority pattern in one statement using the comma (',') operator.
- multiple selectors may be specified for a single action using the semicolon (';') separator.

```
#### RULES ####

# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog


# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages
*.emerg                                                 :omusrmsg:*

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
```

rsyslog

The action field of a rule describes what to do with the message. In general, message content is written to a file, but other actions can be configured, like writing to a database table or forwarding to another host

- Saving rsyslog messages to log files (`/var/log/cron.log`)
- Sending rsyslog messages over the network (`@[zNUMBER]HOST:[PORT]`)
  - Use a single at sign (@) to specify UDP as the transport protocol.
  - Use a double at sign (@@) to specify TCP.
  - The optional zNUMBER field enables a level of zlib compression from 1 to 9.
  - The HOST field specifies the receiving host.
  - The optional PORT field specifies the port number on the receiving host
- Sending rsyslog messages to specific users (e.g. bob)
- Write rsyslog messages into a database (`:PLUGIN:DB_HOST,DB_NAME,DB_USER,DB_PASSWORD;[TEMPLATE]`)
  - The PLUGIN field specifies the plug-in that performs the database writing.
  - rsyslog provides support for MySQL and PostgreSQL databases.
  - MySQL integration requires the rsyslogmysql software package.
  - PostgreSQL requires the rsyslog-pgsql package.
- Sending rsyslog messages to standard output (`/dev/console`)

rsyslog

logger is a command-line tool that provides a shell command interface and gives the user an easy approach to make entries in the system log. It can be useful for testing rsyslog configuration

- `logger [options] <message>`, with `-p` set priority

```
$ logger -p mail.emerg test_message_1
```

```
$ logger -p auth.alert test_message_2
```

```
$ logger -p security.info test_message_3
```

systemd - journald

journald - is the part of systemd that deals with logging. It creates and maintains structured, indexed journals based on logging information that is received from a variety of sources.

Benefits:

- **Indexing.** journald uses a binary storage for logs, where data is indexed, timestamped -> lookups are much faster than with plain text files
- **Structured logging.** Combined with indexing, it means you can easily filter specific logs (e.g. with a set priority, in a set timeframe etc)
- **Access control.** By default, storage files are split by user, with different permissions to each. As a regular user, you won't see everything root sees, but you'll see your own logs
- **Automatic log rotation.** You can configure journald to keep logs only up to a space limit, or based on free space

Core concepts:

- supports in-memory and on-disk data storage
- stores messages in /run folder (non-persistent)
- in order to enable persistent storage, it is necessary to create directory /var/log/journal
- configuration file: /etc/systemd/journald.conf

journald

- **$journalctl** - command is used to work with logs
- **Useful options**:
  - `-f` – follow the journal (runtime)
  - `-r` – show the newest entries first
  - `--since/--until` – apply time filter (supports strings like "2022-10-30 18:17:16" and words like today, tomorrow/yesterday)
  - `-o` – change output format ('-o verbose' can be helpful, and any field from verbose can be used for filtering NAME=value)
  - `-u` – show logs from the specified unit
  - `-p` – show entries with the specified priority

```
    PRIORITY=3
    SYSLOG_FACILITY=0
    SYSLOG_IDENTIFIER=kernel
saltanov@UbuntuPC:~$ journalctl PRIORITY=3
Oct 24 23:53:31 UbuntuPC kernel: rcu: INFO: rcu_sched self-detected stall on CPU
Oct 24 23:53:31 UbuntuPC kernel: rcu:         1-...!: (1 GPs behind) idle=cf3/0/0
Oct 24 23:53:31 UbuntuPC kernel: rcu: rcu_sched kthread timer wakeup didn't happe
Oct 24 23:53:31 UbuntuPC kernel: rcu:         Possible timer handling issue on cp
```

e.g. using verbose mode to filter by PRIORITY

```
saltanov@UbuntuPC:~$ journalctl --disk-usage
Archived and active journals take up 272.0M in the file system.
saltanov@UbuntuPC:~$ sudo journalctl --vacuum-size=200M
[sudo] password for saltanov:
Vacuuming done, freed 0B of archived journals from /var/log/journal.
Vacuuming done, freed 0B of archived journals from /run/log/journal.
Deleted empty archived journal /var/log/journal/921b5cd30a614eb490b017
b70113-c275c66df3851a86.journal~ (8.0M).
Deleted archived journal /var/log/journal/921b5cd30a614eb490b017259d8e
bb04965d0f61dc.journal~ (8.0M).
Deleted archived journal /var/log/journal/921b5cd30a614eb490b017259d8e
c21a4b0015cf1d.journal~ (16.0M).
Deleted archived journal /var/log/journal/921b5cd30a614eb490b017259d8e
-2a1ebf4b4c03053a.journal~ (16.0M).
Deleted archived journal /var/log/journal/921b5cd30a614eb490b017259d8e
d52757d6a3d4b7.journal~ (8.0M).
Deleted archived journal /var/log/journal/921b5cd30a614eb490b017259d8e
64af9436833353.journal~ (8.0M).
Vacuuming done, freed 64.0M of archived journals from /var/log/journal
d8.
saltanov@UbuntuPC:~$ journalctl --disk-usage
Archived and active journals take up 208.0M in the file system.
```

e.g. cleaning up journal and limit it

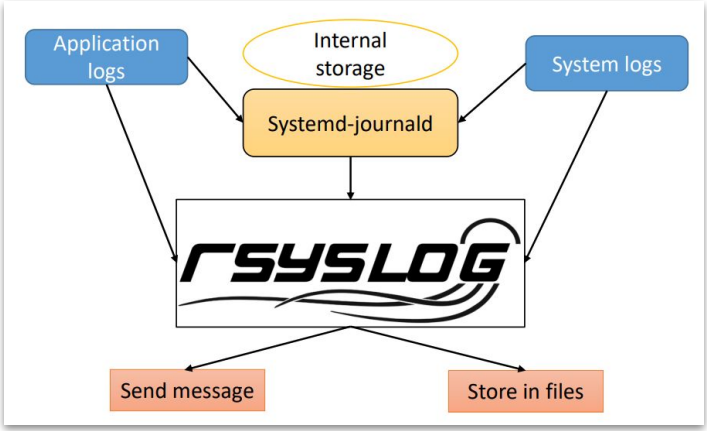
journald vs syslog

There's good integration between the two:

- Journald provides a syslog API and can forward to syslog.
- On the other hand, syslog daemons have journal integrations (e.g. rsyslog provides plugins to both read from journald and write to journald)

In fact, it is recommended two architectures:

- **A small setup** - by centralizing journald logs. If environments don't have systemd/journald but have syslog, they can centralize via syslog to the server and finally write to the server's journal.
- **A larger setup** - by aggregating journal entries through a syslog daemon.
  - There are two ways of centralizing journal entries via syslog:
    1. syslog daemon acts as a journald client (like journalctl or Logstash in ELK) - *is slower as reading from the journal is slower than reading from the socket – but captures all the fields from the journal.*
    2. journald forwards messages to syslog (via socket) - *journal will only forward traditional syslog fields (like severity, hostname, message..).*



Log rotation for the rsyslog

Because **rsyslog** just write logs to files, additional tool is required to configure retention.

The main idea is to configure storing, archiving and removing log files for different periods of time:

- **`$logrotate`** is a tool to rotate log files
  - It is normally run as a daily cron job; cron task can be found in `/etc/cron.daily/logrotate`
- logrotate can help with specific rotation - e.g., running specific scripts before/after rotating (sending signal, archiving log data etc)
  - `man logrotate.conf`
- logrotate configuration files:
  - `/etc/logrotate.conf`
  - `/etc/logrotate.d/*`

Log rotation for the rsyslog

Log rotation configuration rules describes as follows:

- `Log_file_address {directives}`
- Some useful action directives:
  - rotate
  - create
  - dateext
  - compress
  - extension
  - mail
  - maxage
  - missingok
  - olddir
  - postrotate/endscript
  - start
- Time directives:
  - hourly
  - daily
  - weekly
  - monthly
  - yearly

```
# no packages own btmp -- we'll rotate it here
/var/log/btmp {
    missingok
    monthly
    create 0660 root utmp
    rotate 1
}
```

Example - /etc/logrotate.d/btmp