Logging

As you have probably noticed by now, your Linux installation is a complex system of different parts that work together. As a Linux sysadmin you can't monitor your system and all the services that run on it 24/7. This is why your system collects information for you as part of normal operation. This information is gathered into log-files. Your log-files contain a history of events that occured on your system. Most of those events are purely informative, signalling things that are expected when a computer system is running (a user signing in, a process starting, ...), while others try to alert the administrator that something out of the ordinary has occured (a user entering a wrong password, disk space running low, a process that's crashing, ...). As the amount of data logging generates can easily become intimidating it is important that you know how to filter this information, so that you can find the information and events that are of interest to you. If you know how to do this you'll be able to notice when something might become a problem or, when things are going wrong. It gives you the skills to be able to understand what has happened and hopefully avoid the same problems in the future. As logs also contain a lot of security related information, knowing how to use them is an important step in keeping your systems secure and safe from all kind of attacks.

Traditionally the log-files are a collection of textfiles written to the /var/log /-directory. Collecting the information from all the system components was the job of the rsyslog-daemon. While it is a fast and robust system, text files are not always the most user friendly way of presenting information. While there are some excellent tools to filter text based files, as you have seen in previous chapters, the simple structure of a line in a log make fine grained filtering difficult.
Journald is a new service that takes care of logging, trying to work around the limitations of rsyslog. It stores the log data in a binary file, more like a database, which makes it possible to capture a lot more details about an event and allows for greater flexibility when filtering the data. It also provides faster searching than plain text files. These logfiles are however, because of their binary format,

not universily readable. Syslogd on the other hand makes it easier to send events to a networked computer, like a logging server.

Most Linux systems, including Ubuntu, choose a mix of both systems with journald capturing the events, and writing them to binary files, and then passing them on to rsyslogd to handle them in the traditional way. By making this combination, you are able to get the best of both worlds.

Journald

Journald is a daemon, a service, a process that runs in the background to collect logged events and process them. Like any daemon you can check it's current status by using `systemctl status systemd-journald` .

```bash
student@linux-ess:~$ systemctl status systemd-journald
● systemd-journald.service - Journal Service
   Loaded: loaded (/lib/systemd/system/systemd-journald.service; static)
   Active: active (running) since Fri 2022-12-09 22:43:07 CET; 2min 49s
   TriggeredBy:● systemd-journald.socket
               ● systemd-journald-audit.socket
               ● systemd-journald-dev-log.socket
   Docs: man:systemd-journald.service(8)
         man:journald.conf(5)
   Main PID: 237 (systemd-journal)
   Status: "Processing requests..."
   Tasks: 1 (limit: 4625)
   Memory: 14.1M
   CPU: 582ms
   CGroup: /system.slice/systemd-journald.service
      └─237 /lib/systemd/systemd-journald

dec 09 22:43:07 linux-ess systemd-journald[237]: Journal started
dec 09 22:43:07 linux-ess systemd-journald[237]: Runtime Journal (/run/lo
dec 09 22:43:07 linux-ess systemd-journald[237]: Time spent on flushing t
dec 09 22:43:07 linux-ess systemd-journald[237]: System Journal (/var/log
Notice: journal has been rotated since unit was started, output may be in
student@linux-ess:~$
```

You can use this command to see the status of a running service: You can see it's active at the moment, with a status of "Processing requests". This means it is listening in the background, waiting for a process to send it something to add to the log. Underneath the information for the process, you see the last few log messages generated by this service. You'll always get these messages when examining a service with `systemctl`, it can be a quick way to check why a particular service is not working like it should.

Systemd-journald is controlled by the `/etc/systemd/journald.conf` configuration file. When opening this file you'll notice that all the options are commented out (preceded by a #). This is because the default options are embedded in the executable file itself, and are not directly accessible. When you remove the # you'll override that particular default. The values that are filled in in

the configuration file are the default values, so you only need to actually overwrite the ones you need to change. Examining all the options present in this file would be overkill for this course (and the advertised man-page does a very decent job if you need those), but the first one is an important one to be aware of.

bash

```
cat /etc/systemd/journald.conf

# This file is part of systemd.
# ...
# See journald.conf(5) for details.

[Journal]
#Storage=auto
#Compress=yes
#Seal=yes
...
```

Journald can keep its logfiles entirely in the `/run/log/journal` folder when this setting is set to `volatile`. You won't find this folder anywhere on your hard drive, it is a folder created when your system boots that is kept entirely in your computer's working memory. Like anything else that is stored in RAM it is disposed of as the operating system shuts down. This means logfiles are only kept as long as your system is running. Every time your OS boots you start with a clean log. When this setting is set to `persistent` log files are kept in the `/var/log/journal` folder, a location that exists on your hard disk drive. The journal is kept even when the system reboots. The `auto` setting which is Ubuntu's default means it will act in the persistent manner when the /var/log /journal folder exists, and act volatile when it doesn't. If you notice your system isn't keeping its log-files across boots, check if this folder exists and create it if necessary.

Using the Journal

Using `systemctl status ...` you only get to see the last few log messages a

particular service generated. If you want to view the complete log, you need to use the `journalctl` command. This will open the complete journal with the pager `less` . As this journal contains everything logged in the system starting from its first boot (or however long your system is configured to keep old log events around) it quickly becomes too big to just go through it line by line. See it for yourself how many events there are by pressing the `End` -key to jump to the last line. The strength of Journald isbeing able to filter the information so you only see the events you care about.

ℹ The events shown depend on the user running journalctl. The root-user and users in the `adm` -group will see all events generated by the system. A regular user who is not a member of `adm` sees only the events generated by the processes he owns. When making a new user with an administrator role add the user to both the `sudo` -group, so he can run commands as root, and to the `adm` -group so he can see the entire journal.

A log line contains the following information:

```bash
dec 10 18:12:05 linux-ess gnome-shell[1784]: DING: Detected async api for
```

dec 10 18:12:05 : A timestamp signalling when the event occured.
linux-ess: The hostname of the computer where the event was generated.
gnome-shell[1784]: The name of the process generating the event, and its process ID.
DING: Detected async api for thumbnails: The actual message.

ℹ Because journalctl default behaviour is to open the journal with the pager less, you can use the keys you learned in chapter 5 to navigate or search inside the journal.

The first `journalctl` option you should know, is `-e` or `--pager-end` . This will jump straight to the last page of the journal. As this is usually the part of the log you care about, you'll almost always use this option when using

`journalctl`. Another practical option is `-x` or `--catalog`. This will add explanation to log lines when available.

bash

```
student@linux-ess:~$ journalctl -xe
dec 09 23:57:27 linux-ess systemd[1]: Stopped LSB: Start busybox udhcpd a
    Subject: A stop job for unit udhcpd.service has finished
    Defined-By: systemd
    Support: http://www.ubuntu.com/support
    A stop job for unit udhcpd.service has finished.
    The job identifier is 4877 and the job result is done.
dec 09 23:57:27 linux-ess sudo[4795]: pam_unix(sudo:session): session clo

student@linux-ess:~$ journalctl -e
dec 09 23:57:27 linux-ess systemd[1]: Stopped LSB: Start busybox udhcpd a
dec 09 23:57:27 linux-ess sudo[4795]: pam_unix(sudo:session): session clo
```

In the above example you see the same two events, with and without the -x option. It may not mean much now as the extra information isn't the most useful, but this option will save you once you start configuring things like Web Servers on your Linux machine. When you write your own configuration files it will tell you why a particular server will not start with your configuration, pointing to things like invalid options or typos. You won't find this extra information in traditional text-based logs.

Another powerful option is `-u` or `--unit`. This allows you to only view the events generated by a particular 'unit', generally a service running on your system.

bash

```
student@linux-ess:~$ sudo journalctl -u cron.service -e -n 5 --no-pager
dec 09 23:17:01 linux-ess CRON[4117]: pam_unix(cron:session): session clo
dec 09 23:30:01 linux-ess CRON[4218]: pam_unix(cron:session): session ope
dec 09 23:30:01 linux-ess CRON[4218]: pam_unix(cron:session): session clo
dec 10 00:17:01 linux-ess CRON[5161]: pam_unix(cron:session): session ope
dec 10 00:17:01 linux-ess CRON[5161]: pam_unix(cron:session): session clo
```

ℹ Tab-completion works with journalctl. Press Tab twice after -u to list the services you can filter by.

ℹ The options -n 5 (show 5 lines) and --no-pager (do not open with less) are used to make these examples clearer.

Use `--dmesg` to see only the events generated by the kernel.

bash

```
student@linux-ess:~$ journalctl --dmesg -n 2 -e --no-pager
dec 10 00:00:03 linux-ess kernel: audit: type=1400 audit(1670626803.896:1
dec 10 00:00:03 linux-ess kernel: audit: type=1400 audit(1670626803.976:1
```

As this log can go back months you'll also need to be able to specify a timeframe when filtering the journal. To only see the events generated since the last time you started your system, use the `-b` or `--boot` flag. To see the messages of a previous boot, provide an offset: -1 for the previous time your system started, -2 for the time before that, ...

You can ofcourse combine all options to further filter. The example below show the last 5 events the cron service generated, three boots ago.

bash

```
student@linux-ess:~$ journalctl -b -2 -u cron.service -e -n 5 --no-pager
nov 05 10:30:01 linux-ess CRON[4256]: pam_unix(cron:session): session clo
nov 05 11:17:01 linux-ess CRON[4311]: pam_unix(cron:session): session ope
nov 05 11:17:01 linux-ess CRON[4311]: pam_unix(cron:session): session clo
nov 05 11:30:01 linux-ess CRON[4321]: pam_unix(cron:session): session ope
nov 05 11:30:01 linux-ess CRON[4321]: pam_unix(cron:session): session clo
```

ℹ `journalctl --list-boots` gives you an overview of when your system was booted with a timestamp, which is useful information by itself. It also includes the offsets you can use.

You can use `--since` and `--until` to specify a time range

```bash
student@linux-ess:~$ journalctl --since '5 min ago' -n 2 --no-pager
dec 10 00:48:12 linux-ess gnome-shell[1784]: Window manager warning: last
dec 10 00:48:12 linux-ess gnome-shell[1784]: Window manager warning: W2 a

student@linux-ess:~$ journalctl --since '2022-12-09 23:00' -n 2 --no-page
dec 09 23:02:09 linux-ess update-notifier[2468]: gtk_widget_get_scale_fac
dec 09 23:02:09 linux-ess update-notifier[2468]: gtk_widget_get_scale_fac

student@linux-ess:~$ journalctl --since 'yesterday' -n 2 --no-pager
dec 09 22:43:07 linux-ess kernel: Linux version 5.15.0-53-generic (buildd
dec 09 22:43:07 linux-ess kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-

student@linux-ess:~$ journalctl --since '20 day ago' --until '5 day ago'
nov 27 13:49:04 linux-ess kernel: Linux version 5.15.0-50-generic (buildd
nov 27 13:49:04 linux-ess kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-
```

ℹ Examples of valid timestamps are "2022-11-23 23:02:15", "2022-09-01", "12:00", now, today, yesterday, "5 hour ago" (min, hour, day, week, month, year), "-5 min"

Priority levels

As you have probably noticed, not everything that is logged is necessarily important or useful information. This is why every event that is logged also has a priority level. These are, from not important to very important:

Displayed in white in the journal:

- debug: The most verbose setting. Everything that a process can log is logged. This is only useful when debugging a specific problem as you will be able to follow every step a process takes. These messages are usually discarded. If you want to see debug events, you'll have to enable it in the journald configuration.

- info: Information part of normal operation.
- notice: Something unusual happened, but it is not an error. An example of this is a user that is allowed to, using sudo.

Displayed in yellow in the journal:

- warning: If no action is taken an error can be expected. You'll get a warning when your system is close to running out of storage space. Displayed in yellow.

Displayed in red in the journal:

- err: (short for error) A non-fatal error has occured. Meaning a service has encountered an error, but is still able to continue running.
- crit: (short for critical) A process has crashed, or a failure in a primary system component.
- alert: A vital component stops working, expect data loss. Important security errors will also show up as crit. Action needs to be taken immediately.
- emerg: (short for emergency) Only used by the kernel. The system has become unusable. If possible this message will be communicated to all logged in users. As the logging-service is also likely to be affected, you wont see this priority in a log file.

Remember the ominous 'this incident will be reported' when trying to use sudo when you are not allowed to?

bash

```
jacob@linux-ess:/home/student$ sudo useradd jef
[sudo] password for jacob:
jacob is not in the sudoers file.  This incident will be reported.

student@linux-ess:~$ journalctl -b -p alert --no-pager
dec 10 01:54:59 linux-ess sudo[5753]:    jacob : user NOT in sudoers ; TT
```

You can filter by priority level using the `-p` option in combination with the levels mentioned above. You will see all the events with at least that priority.

ℹ The options given here are just some of the more common options to filter the journal. See the manpage for `journalctl` to see the complete list.

rsyslogd

Rsyslogd is the service that logs events in the traditional way. While journald-output is easier to filter for only showing the events you are interested in, there are still use cases where a syslog-based logging system will be more practical. One example is a non-booting system: because of the text-based files you can just grab them off the hard drive and read them, which is harder to do with journald's binary files. A competent Linux administrator will need at least some knowledge of both systems to pick the system apropriate for the current job.

Using rsyslog files

As mentioned in the introduction, rsyslogd takes the events passed down from the journald-service. It processes them and stores the events in text-files. You'll find these files in the `/var/log` folder. Outside of the rsyslog-logs, you'll find a number of other logs created by different programs that implement their own logging. You can get an overview of the different logs by using `ls` or `tree` on this folder.

bash

```
student@linux-ess:~$ ls -l /var/log
total 1600
-rw-r--r--  1 root             root                0 nov  5 00:27 alter
-rw-r--r--  1 root             root            25743 okt  2 19:40 alter
-rw-r-----  1 root             adm                 0 okt  4 02:23 appor
-rw-r-----  1 root             adm               918 okt  2 23:10 appor
drwxr-xr-x  2 root             root             4096 dec 10 18:36 apt
-rw-r-----  1 syslog           adm             16996 dec 10 18:36 auth.
-rw-r-----  1 syslog           adm             31316 dec  9 22:43 auth.
...
```

There are a few files here that you should be aware of: `/var/log/syslog` is the file that captures most rsyslog-events, it is basically the default file, when another file is not specified. `auth.log` is another important one as it contains all security related events (like a failed sing-on attempt, a user using sudo,...). As you can probably guess, boot.log contains events generated during a system boot and kern.log has all the messages the kernel generates. These files are owned by root and the `adm` -group, and the rsyslog-logs are not world readable. This means that just like with journald, you'll need to be a member of the `adm` -group to read them.

To find events in the files you can use the filters you use on any text-file. As those are explained in Chapter 8, we won't go very deep here.

bash

```
student@linux-ess:/var/log$ cat auth.log | grep sudo | tail -n 5   #searc
Dec 10 00:32:20 linux-ess sudo: pam_unix(sudo:session): session closed fo
Dec 10 01:54:59 linux-ess sudo:    jacob : user NOT in sudoers ; TTY=pts/
Dec 10 18:36:26 linux-ess sudo:  student : TTY=pts/0 ; PWD=/home/student
Dec 10 18:36:26 linux-ess sudo: pam_unix(sudo:session): session opened fo
Dec 10 18:36:38 linux-ess sudo: pam_unix(sudo:session): session closed fo

student@linux-ess:/var/log$ cat syslog | grep error | tail -n 5
Dec  9 22:43:37 linux-ess gnome-session[1106]: gnome-session-binary[1106]
Dec  9 22:43:37 linux-ess gnome-session[1106]: gnome-session-binary[1106]
Dec  9 22:43:37 linux-ess gnome-session-binary[1106]: GLib-GIO-CRITICAL:
Dec  9 22:43:37 linux-ess gnome-session-binary[1106]: GLib-GIO-CRITICAL:
```

As you can see, to use this effectively it helps that you have some idea what you are searching for. If you don't know what to search for it is hard to use `grep` effectively. Journald advanced filtering makes it easier to find relevant events when you don't know the right keywords to search for.

ℹ The specific names of the log files are determined by the Linux distribution. On a lot of distributions (of the redhat family) the /var/log /syslog file will for example be called /var/log/messages

Rsyslog configuration

In `/etc/rsyslog.conf` you'll find the default configuration of the rsyslog-service. Among other settings it includes the default file permissions for the log-files. But the part we want to draw attention to are the last few lines.

bash

```
#
# Include all config files in /etc/rsyslog.d/
#

$IncludeConfig /etc/rsyslog.d/*.conf

student@linux-ess:/var/log$ ls /etc/rsyslog.d/
20-ufw.conf  50-default.conf
```

This line says that every file in the rsyslog.d folder should be added at the end of the config-file. A configuration folder ending in '.d' is called a drop-in folder. It allows different programs to add their own configuration by dropping it in this folder. The setting that decides which log-file is used for different types of events is `50-default.conf` (the number decides the order in which the extra config files are added).

bash

```
# First some standard log files.  Log by facility.
#

auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
#daemon.*                       -/var/log/daemon.log
kern.*                          -/var/log/kern.log
#lpr.*                          -/var/log/lpr.log
mail.*                          -/var/log/mail.log
#user.*                         -/var/log/user.log
...
mail.err                        /var/log/mail.err
...
```

```
*.emerg                         :omusrmsg:*
```

Events are categorised by the "facility" that produces the events. For example, auth and authpriv are any event related to security. This is followed by a . and a priority level. None means these events will not be logged in the specified file. In the above example everything related to security is logged into the file /var/log/auth.log. Everything else (.) is added to the syslog file. To stop security messages from ending up in the general syslog, auth.none and authpriv.none are added. Events of the mail subsystem with a *priority of at least* error are added to mail.err. Events of the emergency type (the highest level) are displayed in every logged in user's terminal. If you want these messages to only go to the root-user, replace the line with
omusrmsg
root. You can use @ to send the messages to another networked computer. For example the line

```
*.err               @logger.pxl.be
```

would send all events with a priority of error to a computer with the hostname logger.pxl.be

logrotate

A log-file where no events are ever removed would eventually fill up the file system. It is logrotate's job to cycle the logs and remove the old ones. You'll find the configuration for this in `/etc/logrotate.conf`

bash

```
# rotate log files weekly
weekly
# use the adm group by default, since this is the owning group
# of /var/log/syslog.
su root adm
# keep 4 weeks worth of backlogs
rotate 4
# create new (empty) log files after rotating old ones
create
```

This specifies that every week a new log file is started, and four old log-files should be kept. This means that after five weeks the oldest back-up log file will be removed.

logger

logger is a command that allows you to generate log events yourself. This can be used to record actions that should be kept as part of the log-file. Just use logger and the message you want recorded. With the option -p you can set the priority level of the event.

bash

```
student@linux-ess:/var/log$ logger -p crit "system going down for mainten
student@linux-ess:/var/log$ journalctl -xe -n 2 --no-pager
dec 11 09:45:33 linux-ess sudo[33617]: pam_unix(sudo:session): session cl
dec 11 09:45:47 linux-ess student[34023]: system going down for maintenan
```

Previous
12 Managing drives and filesystems
Next
Assignment