

# Hardening Maintenance

Fortificación de S.O.  
Master en Seguridad Informática. 2023/24  
Universidade da Coruña  
Universidade de Vigo

Antonio Yáñez Izquierdo  
José Rodríguez Pereira

# Contents I

- 1 Introduction
- 2 logs, logfiles and syslogd
  - log configuration
- 3 rotating of logs
  - linux logrotate
- 4 lynis

# Introduction

# Maintaining a system

- once we have hardened our system, the task ahead is to maintain it that way
- security, as said many times, is not a goal but a continuing process
- to keep our system safe we must
  - keep it up day with the corresponding system and/or application patches
  - keep ourselves informed about possible system/application vulnerabilities not already patched so we can uninstall/disable them
  - monitor the system in search of vulnerabilities, attacks or suspicious activities

## Maintaining a system

- the primary source of information on one system is the log subsystem
- everything that happens can be logged, should we want to
- **authentication logs and system critical logs should always be also sent to another (or various) machines**
- in addition we can have external programs check our system for known vulnerabilities or misconfigurations, to make them more secure, such as lynis, openvas . . .

# logs, logfiles and syslogd

# logs

- a *log* is a description of an event that happened to a process in the system
- although some programs can use and maintain their particular log files it is usual the log daemon in the system (typically named *syslogd*) takes care of the logs in a centralized way. (linux usually replaces syslog with another "more advanced" utility like *syslog-ng* or *rsyslog*)
- usually a *log* is a single line of text containing
  - time and date of the event
  - the machine and service where it has originated,
  - the type and severity of event

# logfiles

- a *logfile* is a file where the system stores the logs
- typically is a plain text file containing one line per event
- there can exist different files for different services
- instead of logging to files, logs can also be sent to some device (for example a *terminal*), to users on the system or even to other systems on the network



## location of logs files

- the location of the log files varies from system to system. Nearly every system has them under the `/var` directory
- the location of the files can also be defined by the system administrator.
- most of linux distributions store the logs directly under directory `/var/log/` (or one of its subdirectories)

# syslogd

- *syslogd* is the daemon that takes care of the logs on the system
  - although in most linux distributions this daemon has been substituted by *rsyslogd* or *syslog-ng*
- applications submit messages to *syslogd*
- *syslogd* reads its configuration file when it starts and decides what to do with the messages it receives

# logs, logfiles and syslogd

## → log configuration

# log configuration

- for syslogd (or any of its alternatives) to know what to do with the messages, it must be specified in its configuration file.
- this file is typically **/etc/syslog.conf** (**/etc/rsyslog.conf** if rsyslog is being used ...)
- a log message is classified according to
  - its *facility*: which service has generated the log. One of a predefined list on the system.
  - its *severity*: how important the log is. One of a predefined list on the system.

## syslog facilities

- this are the more usual facilities on syslog

**auth** security/authorization messages

**authpriv** security/authorization messages (private)

**cron** cron and at

**daemon** system daemons without separate facility value

**ftp** ftp daemon

**kernel** kernel messages

**lpr** line printer subsystem

**mail** mail subsystem

**news** USENET news subsystem

**syslog** messages generated internally by syslogd(8)

**user** generic user-level messages

**uucp** uucp subsystem (obsolete)

## syslog severities

- this are the more usual severities on syslog

**emerg** system is unusable

**alert** action must be taken immediately

**crit** critical conditions

**err** error conditions

**warning** warning conditions

**notice** normal, but significant, condition

**info** informational message

**debug** debug-level message

## syslog file format

- each line of the file specifies what to do with some logs. Lines starting with `#` are treated as comments
- the format of one lines is

```
selector <tab> action
```

- selector selects logs based on the facility and severity. It has the form `facility.severity`.
  - some systems accept the `*` as a wildcard for facility and/or severity
  - some systems also accept the format `facility1,facility2.severity` or `facility1.severity1; facility2.severity2`

## syslog file format

- action represents what must be done with the log selected by '*selector*'. It can be one of the following
  - write the log to a file. This is represented by the name of the file (starting with /, if we precede the / with a -, the file is not synced after logging). A log can also get sent to a device (for example a terminal) using the device name as the logfile
  - notify users. In this case, action is a comma separated list of users that would get the log provided they are logged in the system. Usually the symbol \* stands for all users



## syslog file format

- send the log to another machine running *syslogd*. If action starts with **@** the log is sent to the machine specified after the character **@** (name or ip). logs coming from another machine do not get resent to another

```
cron.emrg;cron.alert @192.168.1.5
cron.alert root,cronmaster
cron.err /var/log/cron-errors.log
cron.* /var/log/cron.log
```

- modern syslog alternatives allow us to send the logs to a named pipe (|, discard the log (*stop*) specify another protocol (tcp) or port ...

## extensions

- for logs coming from other machine to be accepted, we need to tell syslog (or its substitute) about. Depending on the *syslog* alternative we are starting, this can be done as a parameter when we start syslog, or as an option in its configuration file.
  - we have to explicitly allow that connection in the firewall, should we have one
- there are a number of functionalities that, although not standard, can be found on many systems, (specially on linux systems, where a great number of *syslogd* alternatives are available)
  - the existence a directory (typically `/etc/syslog.d` where different software packages can place their particular log configuration
  - the possibility of, instead writing the logs to a file (or sending them to another machine), start a program and pass the log to its standard input

# rotating of logs

## rotating of logs

- the problem with log files is that they keep growing in time. Large files use up a lot of disk space and are difficult to manage
- the solution is to *rotate* the logs: create a new file once the log file has a certain size or a certain age.
- in linux **logrotate** is the standard log rotating program

# linux logrotate

- logrotate takes care of rotating, compressing, removing, . . . of log files in linux systems
- it is usually run daily through cron
- *logrotate* has its configuration file `/etc/logrotate.conf`
  - it has some global options which can be overridden by per-file options
  - specific options for some logfile can be specified in the format

```
logfile {
    options
}
```
  - additional specific file configurations can be put in the *logrotate* configuration directory, specified in *logrotate* configuration file (typically `/etc/logrotate.d`)

# sample linux logrotateconfiguration file

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0660 root utmp
    rotate 1
}
```

## sample /etc/logrotate.d/apache

```
/var/log/apache2/*.log {
    weekly
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    create 640 root adm
    sharedscripts
    postrotate
        /etc/init.d/apache2 reload > /dev/null
    endscript
    prerotate
        if [ -d /etc/logrotate.d/httpd-prerotate ]; then \
            run-parts /etc/logrotate.d/httpd-prerotate; \
        fi; \
    endscript
}
```

# lynis



## lynix and openvas

- there are multiple tools to help on auditing systems security. Two of the most common in linux world are

### lynis

- available as a package on most linux distros
- focussed on checking the local system for vulnerable unpatched packages, misconfigurations . . .

### openvas

- Open Vulnerability Assesment System focusses on network vulnerabilities
- can also be executed through a web interface

# lynis

- available through the repositories
- we install the package (as well as its dependencies) with

```
# apt-get install lynis
```
- we execute the program `lynis` from the command line, specifying what we want it to check as arguments
- it writes to the standard output a summary of what it has checked and what it has found