Unix – Chapters 3, 4, 8

Bjarte Wang-Kileng

HVL

January 16, 2025

Western Norway University of Applied Sciences

Outline

1. Chapter 3 – Access Control and Rootly Powers

2. Chapter 4 – Process Control

3. Chapter 8 – User Management

Outline

1. Chapter 3 – Access Control and Rootly Powers

2. Chapter 4 – Process Control

3. Chapter 8 – User Management

Standard UNIX Access control

- Login as a user.
- Users are member in one or several groups of users.
- Processes and files have owners.
- User *root* has special privileges.
- Linux kernel allows for user namespaces.
  - Not all of this chapter apply if using user namespaces.

UID and GID

- System track users and groups by a numeric id.
- User names are mapped to UIDs through the file “/etc/passwd”.
- Group names are mapped to GIDs through the file “/etc/group”.

File system access control

- A file belongs to one owner and one group.
- File owner can set permissions on file.
- File owner can specify group of file.
  - File owner must be member of this group.
  - User *root* can assign any group to a file.
- Access to file can be specified for owner, group and others.

Process ownership

RUID and RGID: Real UID and Real GID
The real owner. Can send signals to the process and change scheduling policy.

EUID and EGID: Effective UID and Effective GID.
Used for access checks.
Needed since processes can be run with privileges different from that of the owner.

SUID and SGID: Saved UID and Saved GID.
Needed when a process with elevated EUID temporarily change its EUID to RUID.

FUID and FGID: File system UID and File system GID.
Used for access control to the file system. Usually equal to EUID.

The root account

- Has UID and GID equal to 0.
- Can perform valid operations on any file or process.
  - Not necessarily true on a network filesystem
- Processes run as root can change its UID and GID.

setuid and setgid

- Programs with setuid or setgid permissions set will run with EUID or EGID equal to that of the program file.
- Used by e.g. the **passwd** program.
- Using capabilities (later), subset of root permissions can be given to programs.

Issue commands as root

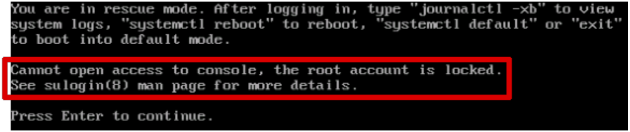
- Login as root.
- Using command **su**.
- Using command **sudo**.

Login as root

- No log or record of operations.
- Not recommended for a production system.
- Can disable the root account:

```
passwd -l root
```

  - Observe that the emergency and rescue boot modes require root login.



```
You are in rescue mode. After logging in, type "journalctl -xb" to view
system logs, "systemctl reboot" to reboot, "systemctl default" or "exit"
to boot into default mode.

Cannot open access to console, the root account is locked.
See sulogin(8) man page for more details.

Press Enter to continue.
```

- Enable root login:

```
sudo passwd -u root
```

Using command **su**

- Log entry of who run the **su** command, but not of the operations.
- Switch *--login* gives an environment similar to a real login.

```
su --login # or su -
```

- Root can **su** to any user without a password.

```
su --login username
```

Using command **sudo**

- Run one command as root (or another user).
- Creates a log entry of command and user.
- Configurable through the file "/etc/sudoers" or directory "/etc/sudoers.d".
  - Use the command **visudo** to edit "/etc/sudoers".
  - If several lines apply, the last matching line applies.
- Can give a user root access to perform specific tasks.
- Switch --*login* or -i gives an environment similar to a real login.
  - Similar to **su** with switch --*login*.
- Switch --*list* or -*l* to list sudo rights of invoking user.

Unattended use of sudo

- E.g. cron jobs.
- Must use the *NOPASSWORD:* parameter.

```
%MYSQL_ADMINS ALL = (mysql) NOPASSWORD: /usr/local/bin/mysqlbackup
```

  - Only use *NOPASSWORD* for specific commands.

- Must allow in sudoers file for sudo without a terminal.

```
Defaults !requiretty
```

Should be OK by default configuration.

System accounts

- Systems typically include many system users and groups.
- Uses low values for UID and GID.
  - See the file "/etc/login.defs".
- Usually no login.
  - Shell set to "/bin/nologin"
- Usually have no elongated access rights, but own files and processes.
  - MariaDB runs as user and group *mysql*.
  - Directory structure "/var/lib/mysql" belongs toa user and group *mysql*.

PAM (Pluggable Authentication Module)

- Authentication framework used on modern Linux systems.
- Pluggable system for user authentication.
- See e.g. the manual page **pam.conf(5)**.
- More later.

File system access lists (ACL)

- Generalization of the user/group/other permissions.
- Can set file system permissions for specific users and groups.
- Part of the file system – Supported by all major UNIX and Linux file systems.

Linux Capabilities

- Traditionally, Linux authorization uses two levels only:
  - Full root access.
  - Normal user access.
- Linux Capabilities divide root powers into approximately 30 separate permissions.
- For a overview of all capabilities, see e.g. the manual **capabilities(7)**.
- Capabilities can be given to program files.
  - Similar to **chmod u+s**, but with less privileges.
  - Processes started from the program file gets the capabilities.
- Capabilities can be given to processes.
  - Process can start child processes and give of its capabilities.

Linux namespaces

- System that wrap global resources into sandboxed environments.
- Isolated containers for resources.
- Users and processes only see the sandboxed environment.
- Mount namespace is an exception:
  - Namespace see the mounts of its parent, but
  - the parent does not see the mounts of child mount namespaces.
  - Can together with **chroot** create isolated islands of file systems.

The Linux namespaces

- PID (processes),
- network (network interfaces, routing),
- UTS (hostname),
- user (UIDs),
- mount (mount points, file systems),
- IPC (System V IPC).

LSM – Linux Security Modules API

- API that allow security modules as loadable kernel modules.
- E.g. SELinux, AppArmour, Smack, TOMOYO, Yama.
- Current LSMs do not cooperate – Use only one.

MAC – Mandatory Access Control

- Access control policies supplement or override traditional model, e.g.:
  - Web documents must reside in “/var/www/html”
  - Only the user can access his home directory.
- Both SELinux and AppArmour are MAC systems.

SELinux

- Created by NSA – Open source.
- Used by Fedora and Redhat with clones, but available also for other distributions.
- Both MAC and role based access control.
- Level of control can be set in file **/etc/selinux/config**.
- Processes are monitored, and only actions consistent with the polices are allowed.
- Security context of files are stored as extended attributes.
- Security context is compared with policies encoded in policy files.
- Details in lab assignment.

AppArmor

- By Canonical (Ubuntu).
- MAC system to supplement the traditional access control system.
- Goal is to limit damage from services.
- Action must be allowed by both the traditional model and AppArmor.

Outline

1. Chapter 3 – Access Control and Rootly Powers

2. Chapter 4 – Process Control

3. Chapter 8 – User Management

Process parameters

- PID: Unique number that identifies the process.
  - Uniqueness only within the same PID namespace.
- PPID: The parent process.
  - Processes are created in a tree structure.
- RUID, EUID, RGID, EGID: See chapter 3.
- Niceness: Scheduling parameter.
  - Low nice values give more CPU time.
- Control terminal: Terminal of STDIN, STDOUT and STDERR.

Signals to processes

- The command **kill** can send signal to a process.

```
kill [-s signal] pid
```

- See book and man page **signal(7)** for more signals:
  - SIGTERM (15): Default signal sent by kill. Terminates the process.
  - SIGKILL (9): Signal can not be caught. Process is killed.
  - SIGINT (2): Signal sent by Ctrl-C.
  - SIGSTOP (19): Freeze process.
  - SIGCONT (18): Continue process stopped with SIGSTOP.
- Always try SIGTERM (default) before SIGKILL.
- Sometimes even SIGKILL will fail (e.g. disk problem).
  - Only reboot will remove process.
- Also **pkill** and **killall**:

```
killall httpd # All httpd processes
pkill -u bki,jon emacs # All emacs processes of bki and jon
```

Processes commands

- Monitoring – **ps**, **pidof**, **pgrep**.

```
ps -C emacs
pidof /usr/bin/emacs
pgrep -u bki emacs
```

- Interactive monitoring – **top**.
  - “q” to quit, “?” for help.
- Change the nice value – **nice**, **renice**.
  - Non-root user can only increase the value.
    - Non-root user can not reset the value.
  - Root can set the value arbitrarily.
- Trace system calls and signals – **strace**.

Process scheduling in Linux

- Uses multi-level queue scheduling with preemption – 101 queues.
- Kernel uses two different scheduling systems:
  - 100 real time queues – Multi-level queue scheduling.
  - One queue for user- and interactive processes – Organized as a self balancing binary tree (a *red-black tree*).
- The 101 queues have priorities from 0 (lowest) to 100 (highest).
  - A process belonging to a higher priority queue will preempt a running lower priority process.

The real time queues

- Uses 100 real time queues with priorities from 1 to 100.
- Within each queue a process can belong to scheduling class **SCHED_FIFO** or **SCHED_RR**.
- A **SCHED_FIFO** process is not preempted unless a higher priority process needs the CPU.
- **SCHED_RR** uses a time quantum of 100ms.

User- and interactive processes

- The queue of user- and interactive processes has priority 0 (lowest).
- Processes usually belong to scheduling class **SCHED_OTHER**, but can also be **SCHED_IDLE** or **SCHED_BATCH**.
- The importance of a process is determined by its scheduling class and a value called the *nice* value.
    - *nice* values go from -20 (most favorable) to +19 (least favorable).
- The scheduling algorithm is named Earliest Eligible Virtual Deadline first (EEVDF).
- EEVDF improves on the older *Completely Fair Scheduler* (CFS):

  CFS always tries to split up CPU time between runnable tasks as close to "ideal multitasking hardware" as possible.

CFS and the *nice* value

- Each process is assigned a “virtual” run time.
- The “virtual” run time of a process increases monotonic with its CPU time, normalized to the minimum run time of the queue.
  - How does the Completely Fair Scheduler prevent starvation . . .
- The “virtual” run time clock ticks slower with lower *nice* values.
- The “virtual” run time clock ticks slower for processes of **SCHED_OTHER** compared to processes of **SCHED_IDLE** and **SCHED_BATCH**.
- CFS will run the process with the shortest “virtual” run time.
  - Low *nice* values give more CPU time.

The “/proc” file system

- Linux uses several pseudo file systems with “files” that are dynamically created by the kernel.
    - Not real files, but
    - entry points to kernel data.
- All process information is stored in “/proc”, e.g. info on *systemd* below “/proc/1/”.
- Commands like **ps** and **top** get their information from “/proc”.

Periodic processes

- Cron
- Anacron
- Systemd timers

Cron

- Crontab files configure tasks to be run at certain times.
- Main crontab file – “/etc/crontab”
- Scripts in directories “/etc/cron.d”, “/etc/cron.hourly”, “/etc/cron.daily”, “/etc/cron.weekly”, “/etc/cron.monthly”.
- Users must use command **crontab** to create crontab file.
  - Users crontab files in “/var/spool/cron”.
- For format of crontab file:

```
man 5 crontab
```

- Logging:
  - If syslog, in file “/var/log/cron”.
  - Systemd journal, e.g.:

```
journalctl --unit=crond.service
```

Anacron

- Similar to Cron, but does not assume machine to be on at all times.
  - Will run missed job when computer is turned on.
- Main configuration file – “/etc/anacrontab”
- Cron and Anacron are working togheter on RedHat 9 and clones.
  - The *crond.service* is running *anacron* one minute past every hour, then
  - Anacron is running the daily, weekly and monthly Cron jobs.

systemd timers

- Absolute and relative times, e.g. 30 seconds after boot:

```
OnBootSec=30
```

  - See book for more parameters.

- Can be started and enabled as other systemd units.

Outline

1. Chapter 3 – Access Control and Rootly Powers
2. Chapter 4 – Process Control
3. Chapter 8 – User Management

Account mechanism

- Login information can have many sources, see the file "/etc/nsswitch.conf".
  - E.g. local files, SSSD, NIS.
  - The SSSD daemon can use eg. LDAP, Kerberos and Active Directory for authentication.
    - More details later and in lab assignment.
- Traditionally, account information is found in the file "/etc/passwd".

Files

“/etc/passwd”: User name, password, uid, gid, gecos, home, shell.

- GECOS – Readable personal info.
- Password usually in shadow file.

“/etc/shadow”: User name, encrypted password, password lifetime fields.

“/etc/group”: Group name, password, gid, list of member uids.

- Group password is rarely used.
- Group password can allow non members to enter group.

“/etc/gshadow”: As “/etc/shadow”, but for groups.

- Rarely used.

Passwords

- Stored encrypted:

```
$prefix$options$salt$password-hash
```

  - The prefix field specifies the hash method.
  - Value **6** specifies SHA-512, and was default on e.g. RHEL 7 and 8.
    - No *options* field.
  - Value **y** specifies yescrypt, and is default on RHEL 9.

- Configuration of password strength and number of login attempts through PAM modules.
  - E.g. **pam_pwquality**, **pam_faillock**.

Working with users and groups

- GUI programs.
  - Not suitable for bulk processing and scripting.
- Using commands.
- Working with the account system files.
  - The commands **vipw** and **vigr** will lock files before opening an editor.

Commands

User commands: **passwd**, **useradd**, **usermod**, **userdel**, **chfn**, **chage**, **chsh**, **newusers**.

- **passwd**, **chfn** and **chsh** also by normal users.

Group commands: **gpasswd**, **groupadd**, **groupmod**, **groupdel**.

- Command **gpasswd** also by group administrator.

Information: **getent**, **id**, **whoami**, **finger**, **pinky**.

- Also normal users.

Work as another: **su**, **sg**, **runuser**, **newgrp**, **sudo**.

Access: **chown**, **chmod**, **setfacl**, **chgrp**.

Configuration files

“/etc/default/useradd”: Defaults for **useradd**.

“/etc/skel”: Content to fill in HOME with **useradd**.

“/etc/login.defs”: Shadow password suite configuration.

“/etc/security”: Configuration files for login.

Remove users

- Accounts can be locked and unlocked.
  - `usermod -L` and `usermod -U`.
- After removing user, e.g. using `userdel`, check for remnant files.

```
find filesystem -xdev -nouser
```

- Remember also databases, phone lists, crontab, pending jobs, processes, mail spool etc.