U1 – Chapter 3, 4, 8

Bjarte Kileng

HVL

January 10, 2018

Standard UNIX Access control

- Login as a user.
- Users are member in one or several groups of users.
- Processes and files has owners.
- User *root* has special privileges.
- Linux kernel allow 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

- File has 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.
- 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.

FSUID 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.
- 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 the program file.
- Used by e.g. the **passwd** program.
- Using capabilities (later), specific 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 though that emergency and rescue boot modes require root login.

- Enable root login on Ubuntu:

```
sudo passwd -u root
```

Using command **su**

- Log entry of who run the **su** command, but not of the operations.
- Root can **su** to any user without a password.
- Use `su - 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».
  - Use the command **visudo** to edit the file.
  - If several lines apply, the last matching line is used.
- Can restrict root access to specific tasks.
- Only allow sudo access to users that you can trust with administrative tasks.

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
```

  Off by default.

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.
  - E.g. user and group *mysql*.
  - The MySQL (or MariaDB) run as user and group *mysql*.
  - Directory structure «/var/lib/mysql» has user and group *mysql*.

PAM (Pluggable Authentication Module)

- Authentication framework used on modern Linux systems.
- Pluggable system for user authentication.
- More later.

Kerberos

- PAM can use Kerberos for authentication.
- Network cryptographic authentication.
- Uses a third party server for authentication.
- Used also by Microsoft Active Directory.
- 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.
- More later.

Linux Capabilities

- Traditionally, Linux authorization uses two levels only:
  - Full root access.
  - Normal user access.
- Linux Capabilities divide root powers into approx. 30 separate permissions.
- For a overview of all capabilities, see the man page *capabilities*.
- Capabilities can be given to program files.
  - Similar to **chmod u+s**, but with less privileges.
  - Processes started from the program file get 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.
  - Except for mount namespaces.

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 and Yama.
- Current LSMs do not cooperate – Use only one.

MAC – Mandatory Access Control

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

SELinux

- Created by NSA – Open source.
- Used by RedHat, CentOS, Fedora, 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.

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.

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.
- UID, EUID, GID, 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 **ps** can send a signal to a process.

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

- See book for more signals:
  - 15 (TERM): Default signal sent by kill. Terminates the process.
  - 9 (KILL): Signal can not be caught. Guarantees that process is killed.
  - 2 (INT): Signal sent by Ctrl-C.
  - 17 (STOP): Freeze process.
  - 19 (CONT): Continue process stopped with STOP.
- Always try TERM (default) before KILL.
- Sometimes even KILL will fail (e.g. disk problem).
  - Only reboot will remove process.
- Also **pkill** and **killall**:

```
killall httpd # All httpd processes
pskill -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**.
  - If increased, can not be returned to its previous 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 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 have 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 *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 total CPU time.
- 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.
- All process information is stored in «/proc», e.g. systemd below «/proc/1/».
- Commands like **ps** and **top** get their information from «/proc».

Periodic processes

- cron
- Systemd timers

Cron

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

```
man 5 crontab
```

- Logging in «/var/log/cron».

systemd timers

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

```
OnBootSec=30
```

  - See book for more types.
- Can be started and enabled as other systemd units.

Account mechanism

- Login information can have many sources, see the file «/etc/nsswitch.conf».
  - E.g. local files, LDAP, NIS.
  - LDAP (and perhaps NIS) will be covered later
- 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 group.

- Rarely used.

Passwords

- Stored encrypted:

```
$id$salt$encrypted password
```

  - The id field specify the hash method.
  - Value 6 means SHA-512.
- Configuration of password strength and number of login attempts through PAM modules.
  - E.g. **pam_tally2**, **pam_pwquality**.

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** and **chfn** also by normal users.

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

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

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

- Also normal users.

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

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

Configuration files

«/etc/default/useradd»:	Defaults for **useradd**.
«/etc/skel»:	HOME content 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, check for remnant files.

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

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