# Privilege separation

## - UNIX security model -

Myrto Arapinis
School of Informatics
University of Edinburgh

# Privilege separation

Modern computers are

1. multi-users
2. multi-tasking

Goal: Prevent potentially misbehaving users and/or applications from harming the rest of the system

Permissions system: mechanisms for achieving separation between components

# Central question

“Who is allowed to access what and how?”

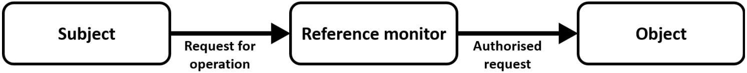
The subject (who) - *eg.* user, application, process

The object (what) - protected resource, *eg.* hardware device, network socket, memory, files, directories, *etc.*

The access operation (how) - *eg.* read, write, execute

# Key assumptions for separation

1. The system know who the user is - user has authenticated, *e.g.* using username / password
2. Complete mediation - all requests are mediated - all requests go to the reference monitor that enforces specified access control policies

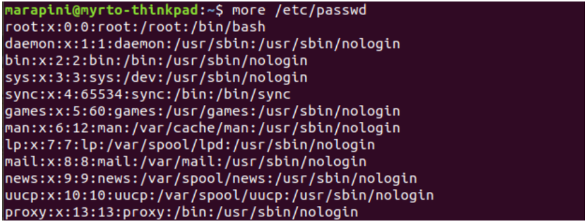


The **reference monitor** grants permission to users to apply certain operations to a given resource

# Users

Two types of accounts each with a unique identifier, the user ID (`uid`):

1. User accounts - associated with humans
2. Service accounts - associated with background processes



```
marapini@myrto-thinkpad:~$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

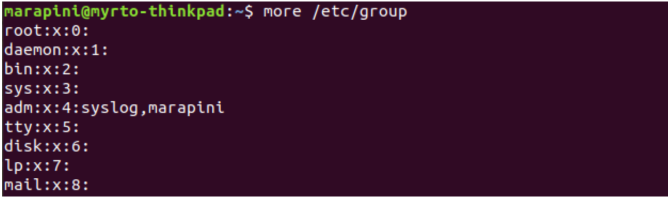
- One entry in the `/etc/passwd` per account with the fields:

  `username:password:uid:gid:uid_info:home:shell`

- `uid 0` - user root `uid`

# Groups

- Groups are sets of users that share resources
- Every group has a name and a unique identifier, the group ID (gid)
- Allow for easier users management and monitoring



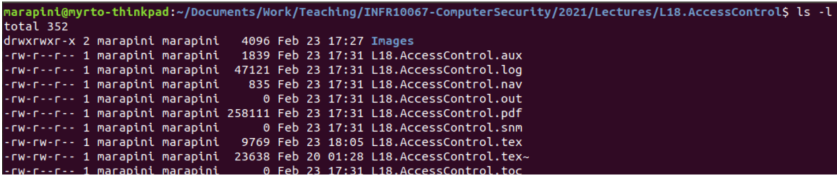
```
marapini@myrto-thinkpad:~$ more /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,marapini
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
```

- One entry in the /etc/group per group with the fields:

```
group_name:password:gid:group_list
```

# File permissions

- All resources (sockets, directories, files) are managed as files
- 3 defined permissions: read (r), write (w), execute (x)
- Permissions are defined for the owner, the owner's group, and other users
- Root and owner can change file permissions
- Only root can change file ownership



```
marapini@myrto-thinkpad:~/Documents/Work/Teaching/INFR10067-ComputerSecurity/2021/Lectures/L18.AccessControl$ ls -l
total 352
drwxrwxr-x 2 marapini marapini   4096 Feb 23 17:27 Images
-rw-r--r-- 1 marapini marapini   1839 Feb 23 17:31 L18.AccessControl.aux
-rw-r--r-- 1 marapini marapini  47121 Feb 23 17:31 L18.AccessControl.log
-rw-r--r-- 1 marapini marapini    835 Feb 23 17:31 L18.AccessControl.nav
-rw-r--r-- 1 marapini marapini      0 Feb 23 17:31 L18.AccessControl.out
-rw-r--r-- 1 marapini marapini 258111 Feb 23 17:31 L18.AccessControl.pdf
-rw-r--r-- 1 marapini marapini      0 Feb 23 17:31 L18.AccessControl.snm
-rw-rw-r-- 1 marapini marapini   9769 Feb 23 18:05 L18.AccessControl.tex
-rw-rw-r-- 1 marapini marapini  23638 Feb 20 01:28 L18.AccessControl.tex~
-rw-r--r-- 1 marapini marapini      0 Feb 23 17:31 L18.AccessControl.toc
```

## Directory permissions

- Execute permission on a directory allows traversing it
- Read permission on a directory allows lookup

# Directory permissions

- Execute permission on a directory allows traversing it
- Read permission on a directory allows lookup

Quizz: Imagine you have the following groups:

- `infr10067` - for any user involved with the Computer Security course
- `tas` - for all Informatics TAs

How can you have a folder only for Computer Security TAs?

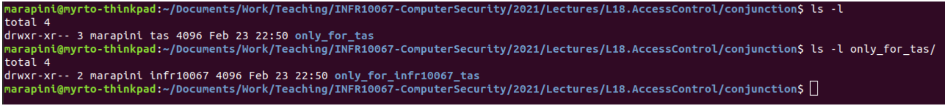
# Directory permissions

- Execute permission on a directory allows traversing it
- Read permission on a directory allows lookup

Quizz: Imagine you have the following groups:

- `infr10067` - for any user involved with the Computer Security course
- `tas` - for all Informatics TAs

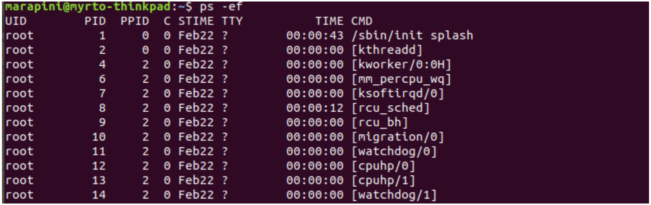
How can you have a folder only for Computer Security TAs?



```
marapini@myrto-thinkpad:~/Documents/Work/Teaching/INFR10067-ComputerSecurity/2021/Lectures/L18.AccessControl/conjunction$ ls -l
total 4
drwxr-xr-- 3 marapini tas 4096 Feb 23 22:50 only_for_tas
marapini@myrto-thinkpad:~/Documents/Work/Teaching/INFR10067-ComputerSecurity/2021/Lectures/L18.AccessControl/conjunction$ ls -l only_for_tas/
total 4
drwxr-xr-- 2 marapini infr10067 4096 Feb 23 22:50 only_for_infr10067_tas
marapini@myrto-thinkpad:~/Documents/Work/Teaching/INFR10067-ComputerSecurity/2021/Lectures/L18.AccessControl/conjunction$ 
```

# Processes

- Each process has a unique identifier, the process ID (`pid`)
- Each process is associated with the user that spanned it



```
marapini@myrto-thinkpad:~$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Feb22 ?        00:00:43 /sbin/init splash
root         2     0  0 Feb22 ?        00:00:00 [kthreadd]
root         4     2  0 Feb22 ?        00:00:00 [kworker/0:0H]
root         6     2  0 Feb22 ?        00:00:00 [mm_percpu_wq]
root         7     2  0 Feb22 ?        00:00:00 [ksoftirqd/0]
root         8     2  0 Feb22 ?        00:00:12 [rcu_sched]
root         9     2  0 Feb22 ?        00:00:00 [rcu_bh]
root        10     2  0 Feb22 ?        00:00:00 [migration/0]
root        11     2  0 Feb22 ?        00:00:00 [watchdog/0]
root        12     2  0 Feb22 ?        00:00:00 [cpuhp/0]
root        13     2  0 Feb22 ?        00:00:00 [cpuhp/1]
root        14     2  0 Feb22 ?        00:00:00 [watchdog/1]
```

- When a user runs a process, it runs with that user's privileges, *i.e.* they can access any resource that user has permissions for
- By default, a child process inherits its parent's privileges
- Processes are isolated in memory

# Process user IDs

Every process has:

- Real user ID (`uid`) - the user ID that started that process
- Effective user ID (`euid`) - the user ID that determines the process' privileges
- Saved user ID (`suid`) - the effective user ID before the last modification

# Process user IDs

Every process has:

- Real user ID (uid) - the user ID that started that process
- Effective user ID (euid) - the user ID that determines the process' privileges
- Saved user ID (suid) - the effective user ID before the last modification

Users can change a process' IDs:

```
setuid(x)        seteuid(x)
  uid ← x          uid ← uid
  euid ← x         euid ← x
  suid ← x         suid ← suid
```

- Root can change euid/uid to arbitrary values x:
- Unprivileged users can only change euid to uid or suid:

# Dropping privileges with `setuid`

Imagine a program that runs as root and wants to fork a process with lower privileges using the following code:

```
if (auth(uid, pwd) == SUCCESS) {
  if (fork() == 0) {
    seteuid(uid);
    exec("/bin/bash");
  }
}
```

# Dropping privileges with setuid

Imagine a program that runs as root and wants to fork a process with lower privileges using the following code:

```
if (auth(uid, pwd) == SUCCESS) {
  if (fork() == 0) {
    seteuid(uid);
    exec("/bin/bash");
  }
}
```

← the user can call
seteuid(0)
and become root!!

# Dropping privileges with `setuid`

Imagine a program that runs as root and wants to fork a process with lower privileges using the following code:

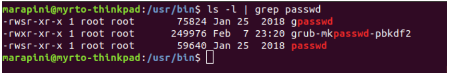
```
if (auth(uid, pwd) == SUCCESS) {
  if (fork() == 0) {
    setuid(uid);
    exec("/bin/bash");
  }
}
```

← the user cannot change uid

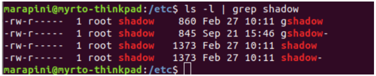
# Elevating privileges - setuid programs

- An executable file can have the set-user-ID property (setuid) enabled
- If A executes a setuid file owned by B, then the euid of the process is B and not A
- Writing secure setuid programs is tricky because vulnerabilities may be exploited by malicious user actions

- Some programs that access system resources are owned by root and have the setuid bit set (setuid programs)



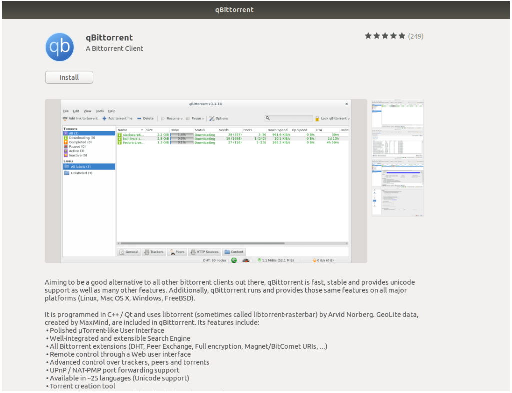
```
marapini@myrto-thinkpad:/usr/bin$ ls -l | grep passwd
-rwsr-xr-x 1 root root       75824 Jan 25  2018 gpasswd
-rwxr-xr-x 1 root root      249976 Feb  7 23:20 grub-mkpasswd-pbkdf2
-rwsr-xr-x 1 root root       59640 Jan 25  2018 passwd
marapini@myrto-thinkpad:/usr/bin$ 
```



```
marapini@myrto-thinkpad:/etc$ ls -l | grep shadow
-rw-r-----  1 root shadow    860 Feb 27 10:11 gshadow
-rw-r-----  1 root shadow    845 Sep 21 15:46 gshadow-
-rw-r-----  1 root shadow   1373 Feb 27 10:11 shadow
-rw-r-----  1 root shadow   1373 Feb 27 10:11 shadow-
marapini@myrto-thinkpad:/etc$ 
```

# UNIX permissions are too coarse-grained

All application installed by a single user account have the same privileges!



**qBittorrent**

A Bittorrent Client

★★★★★ (249)

| Name | Size | Done | Status | Seeds | Peers | Down Speed | Up Speed | ETA | Ratio |
|---|---|---|---|---|---|---|---|---|---|
| slackware6... | 2.2 GiB | 1.4% | Downloading | 39 (357) | 3 (9) | 961.6 KiB/s | 0 B/s | 39m | |
| kali-linux-1.... | 2.8 GiB | 0.0% | Downloading | 19 (1698) | 1 (242) | 10.1 KiB/s | 0 B/s | 1d 13h | |
| Fedora-Live... | 1.3 GiB | 0.1% | Downloading | 27 (116) | 5 (13) | 166.2 KiB/s | 0 B/s | 4h 59m | |

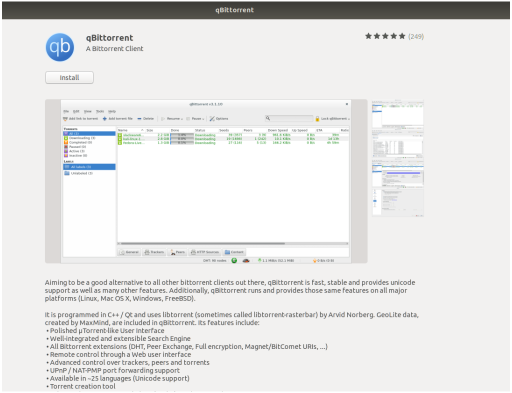
Aiming to be a good alternative to all other bittorrent clients out there, qBittorrent is fast, stable and provides unicode support as well as many other features. Additionally, qBittorrent runs and provides those same features on all major platforms (Linux, Mac OS X, Windows, FreeBSD).

It is programmed in C++ / Qt and uses libtorrent (sometimes called libtorrent-rasterbar) by Arvid Norberg. GeoLite data, created by MaxMind, are included in qBittorrent. Its features include:

- Polished µTorrent-like User Interface
- Well-integrated and extensible Search Engine
- All Bittorrent extensions (DHT, Peer Exchange, Full encryption, Magnet/BitComet URIs, ...)
- Remote control through a Web user interface
- Advanced control over trackers, peers and torrents
- UPnP / NAT-PMP port forwarding support
- Available in ~25 languages (Unicode support)
- Torrent creation tool

# UNIX permissions are too coarse-grained

All application installed by a single user account have the same privileges!



| Name | Size | Done | Status | Seeds | Peers | Down Speed | Up Speed | ETA | Ratio |
|---|---|---|---|---|---|---|---|---|---|
| slackware6... | 2.2 GiB | 1.4% | Downloading | 39 (357) | 3 (9) | 961.6 KiB/s | 0 B/s | 39m | |
| kali-linux-1.... | 2.8 GiB | 0.0% | Downloading | 19 (1698) | 1 (242) | 10.1 KiB/s | 0 B/s | 1d 13h | |
| Fedora-Live... | 1.3 GiB | 0.1% | Downloading | 27 (116) | 5 (13) | 166.2 KiB/s | 0 B/s | 4h 59m | |

Aiming to be a good alternative to all other bittorrent clients out there, qBittorrent is fast, stable and provides unicode support as well as many other features. Additionally, qBittorrent runs and provides those same features on all major platforms (Linux, Mac OS X, Windows, FreeBSD).

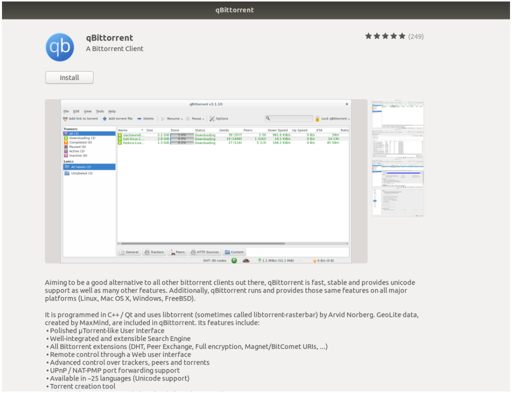
It is programmed in C++ / Qt and uses libtorrent (sometimes called libtorrent-rasterbar) by Arvid Norberg. GeoLite data, created by MaxMind, are included in qBittorrent. Its features include:

- Polished µTorrent-like User Interface
- Well-integrated and extensible Search Engine
- All Bittorrent extensions (DHT, Peer Exchange, Full encryption, Magnet/BitComet URIs, ...)
- Remote control through a Web user interface
- Advanced control over trackers, peers and torrents
- UPnP / NAT-PMP port forwarding support
- Available in ~25 languages (Unicode support)
- Torrent creation tool

!!? What if qBittorent is malware ?!!

# UNIX permissions are too coarse-grained

All application installed by a single user account have the same privileges!



| Name | Size | Done | Status | Seeds | Peers | Down Speed | Up Speed | ETA | Ratio |
|---|---|---|---|---|---|---|---|---|---|
| slackware6... | 2.2 GiB | 1.4% | Downloading | 39 (357) | 3 (9) | 961.6 KiB/s | 0 B/s | 39m | |
| kali-linux-1.... | 2.8 GiB | 0.0% | Downloading | 19 (1698) | 1 (242) | 10.1 KiB/s | 0 B/s | 1d 13h | |
| Fedora-Live... | 1.3 GiB | 0.1% | Downloading | 27 (116) | 5 (13) | 166.2 KiB/s | 0 B/s | 4h 59m | |

Aiming to be a good alternative to all other bittorrent clients out there, qBittorrent is fast, stable and provides unicode support as well as many other features. Additionally, qBittorrent runs and provides those same features on all major platforms (Linux, Mac OS X, Windows, FreeBSD).

It is programmed in C++ / Qt and uses libtorrent (sometimes called libtorrent-rasterbar) by Arvid Norberg. GeoLite data, created by MaxMind, are included in qBittorrent. Its features include:

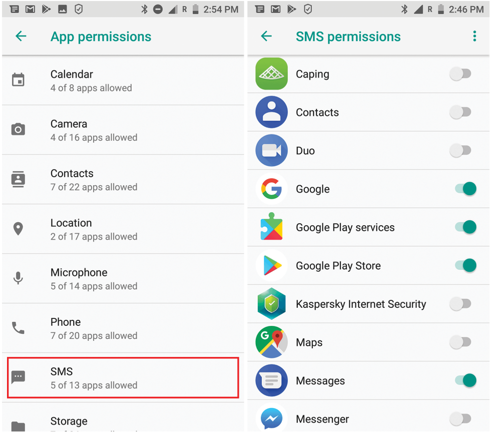
- Polished μTorrent-like User Interface
- Well-integrated and extensible Search Engine
- All Bittorrent extensions (DHT, Peer Exchange, Full encryption, Magnet/BitComet URIs, ...)
- Remote control through a Web user interface
- Advanced control over trackers, peers and torrents
- UPnP / NAT-PMP port forwarding support
- Available in ~25 languages (Unicode support)
- Torrent creation tool

!!? What if qBittorent is malware ?!!

- Better delegate capabilities associated with specific root powers

# Android permissions

- Each app runs with a different user ID
- Apps do not interact
- Permissions are set per app



2:54 PM

App permissions

Calendar
4 of 8 apps allowed

Camera
4 of 16 apps allowed

Contacts
7 of 22 apps allowed

Location
2 of 17 apps allowed

Microphone
5 of 14 apps allowed

Phone
7 of 20 apps allowed

SMS
5 of 13 apps allowed

Storage

2:46 PM

SMS permissions

| | |
|---|---|
| Caping | ○ |
| Contacts | ○ |
| Duo | ○ |
| Google | ● |
| Google Play services | ● |
| Google Play Store | ● |
| Kaspersky Internet Security | ○ |
| Maps | ○ |
| Messages | ● |
| Messenger | ○ |

# Take aways

The UNIX security model provides a simple and flexible model, **but permissions are too coarse-grained**:

- same permissions for all applications ran under a single user account
- many utilities have the `setuid` bit enabled

→ many opportunities for privilege escalation attacks

→ better use capabilities when delegating privileges