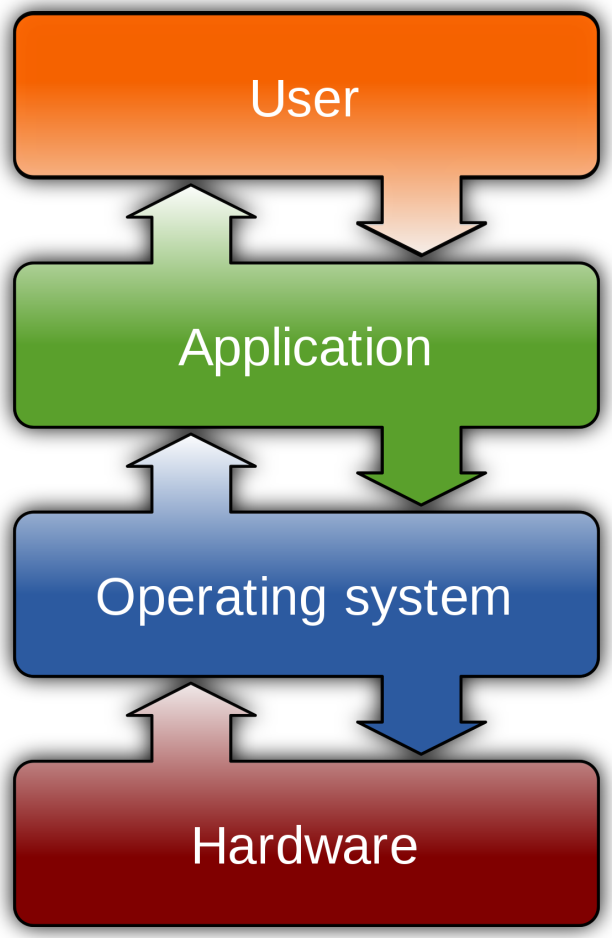
Operating Systems

- key concepts & security principles -

Myrto Arapinis
School of Informatics
University of Edinburgh

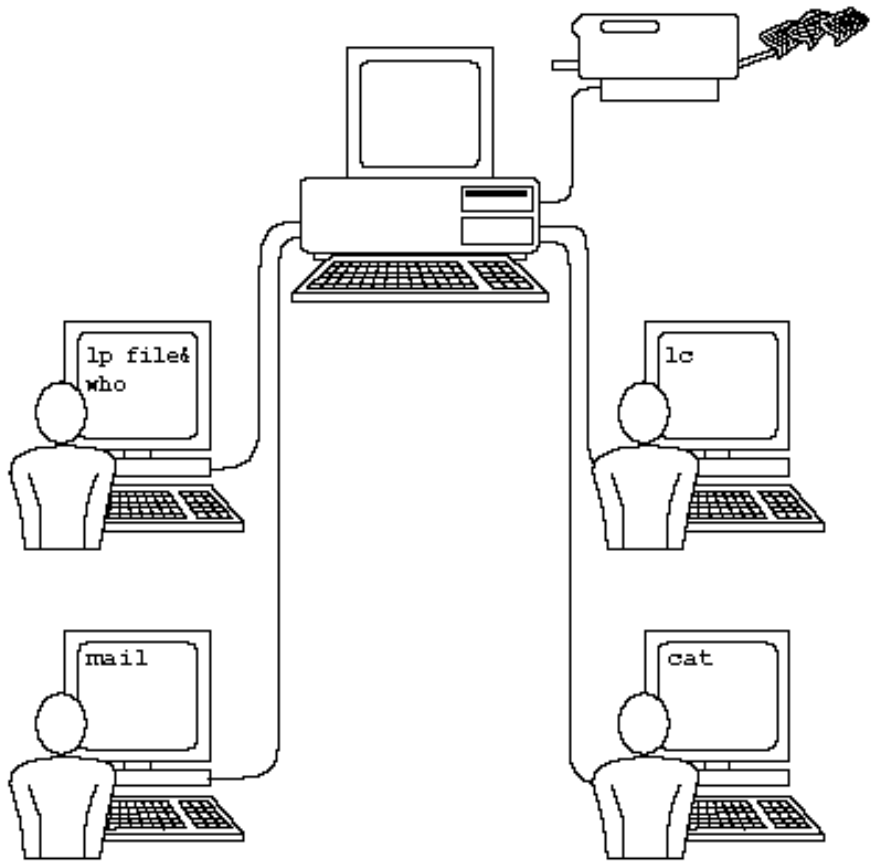
Operating systems

- An OS provides the interface between the users of a computer and that computer's hardware.
- The OS handles the management of low-level hardware resources:
  - disk drives,
  - CPU,
  - RAM,
  - I/O devices, and
  - network interfaces.



Multi-users

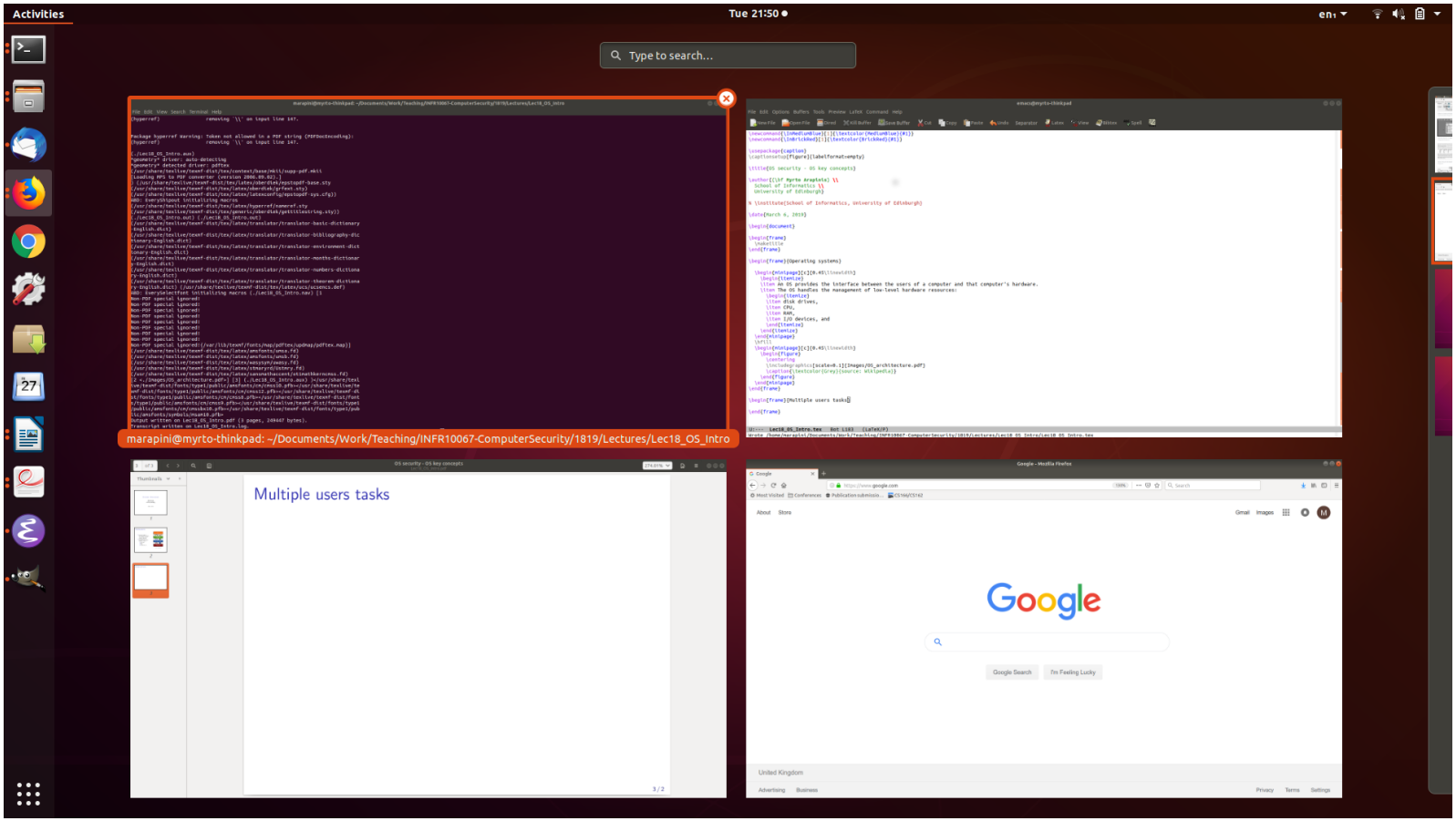
OSes must allow for multiple users with potentially different levels of access to the same computer.



The OS needs to have in place mechanisms to isolate different users.

Multi-tasking

OSes must allow multiple application programs to run at the same time



The OS needs to have in place mechanisms to isolate different applications running.

Essential Unix architecture

Kernel

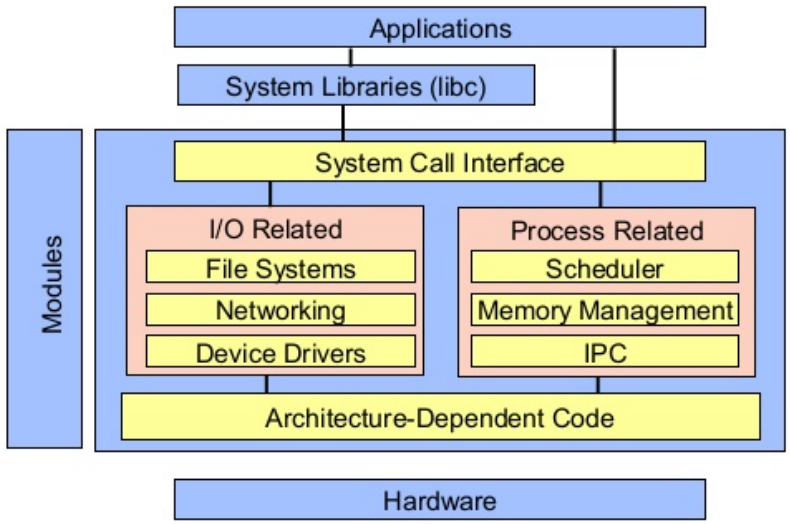
- key component of the OS
- supports secure sharing of low-level resources between users/applications
- kernel limits how applications access computer resources

Execution modes

- User mode - access to resources through syscall to kernel
- Kernel mode - direct access to resources

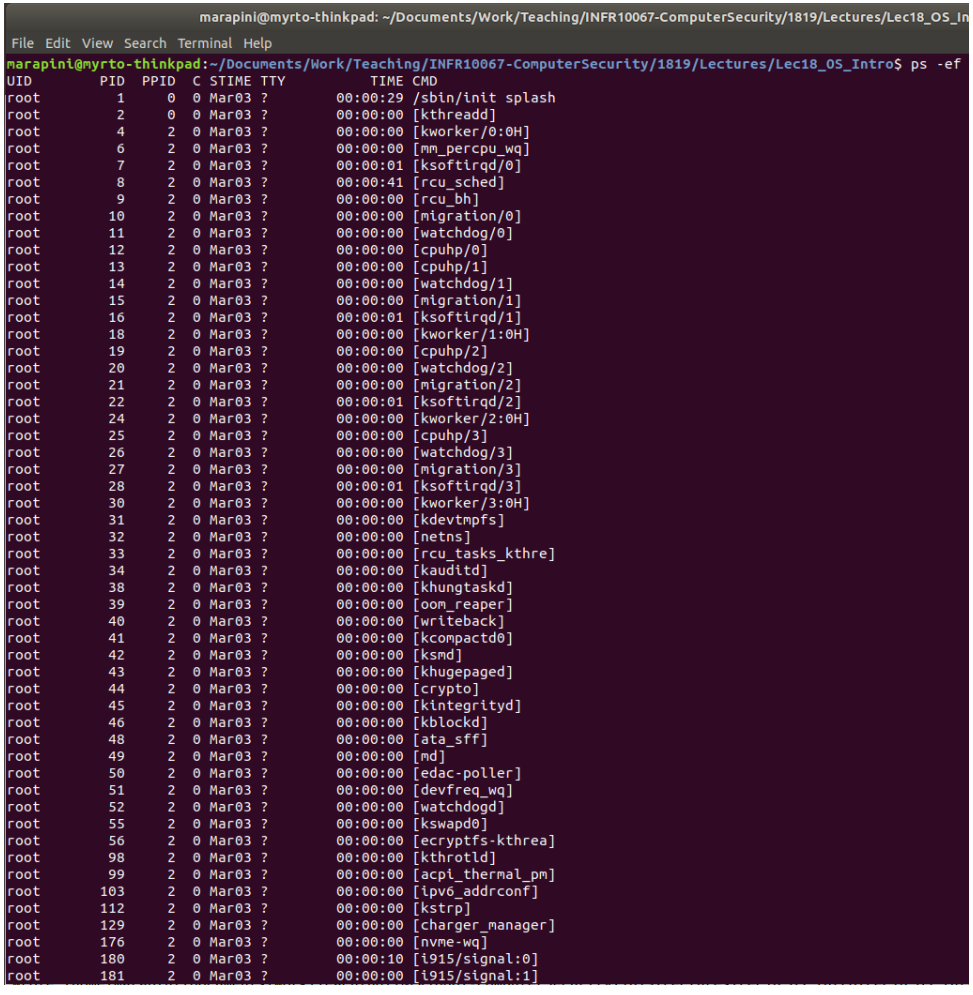
System calls are usually contained in a collection of programs, *eg.* a library such as the C library `libc`:

- `open() close() read() write()`
- `wait() fork() exec() exit()`

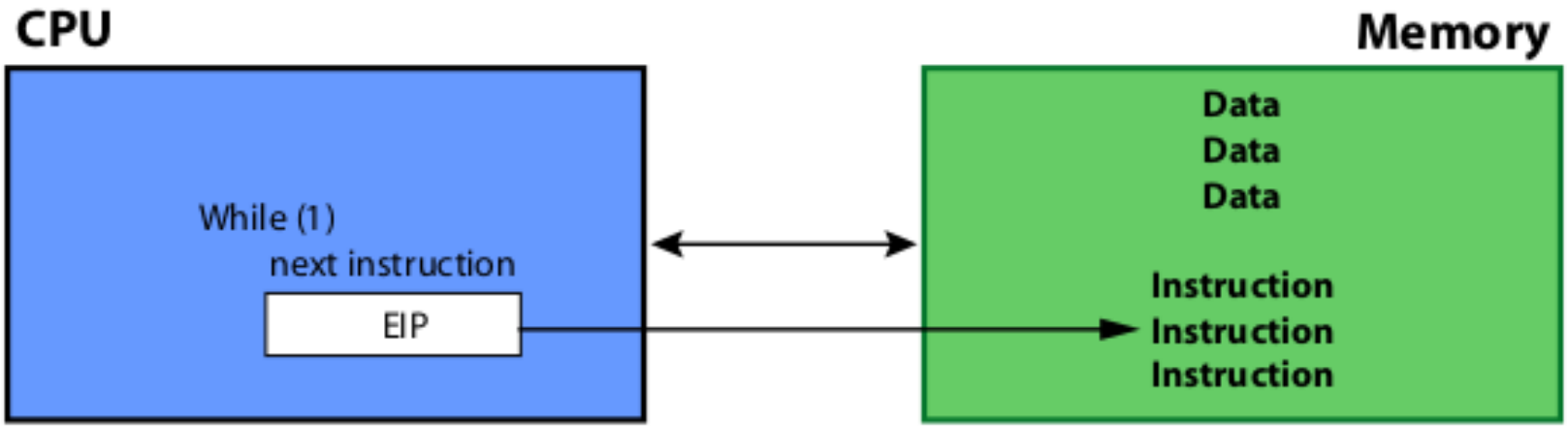


Processes and process management

- A process is an instance of a program that is currently executing
- To actually be executed the program must be loaded into RAM and uniquely identified
- Each process running is identified by a unique process ID (`pid`)
- To a `pid`, we can associate its CPU time, memory usage, user ID (`uid`), program name, etc
- A process might control other processes (fork)
- Child process inherits context from parent process

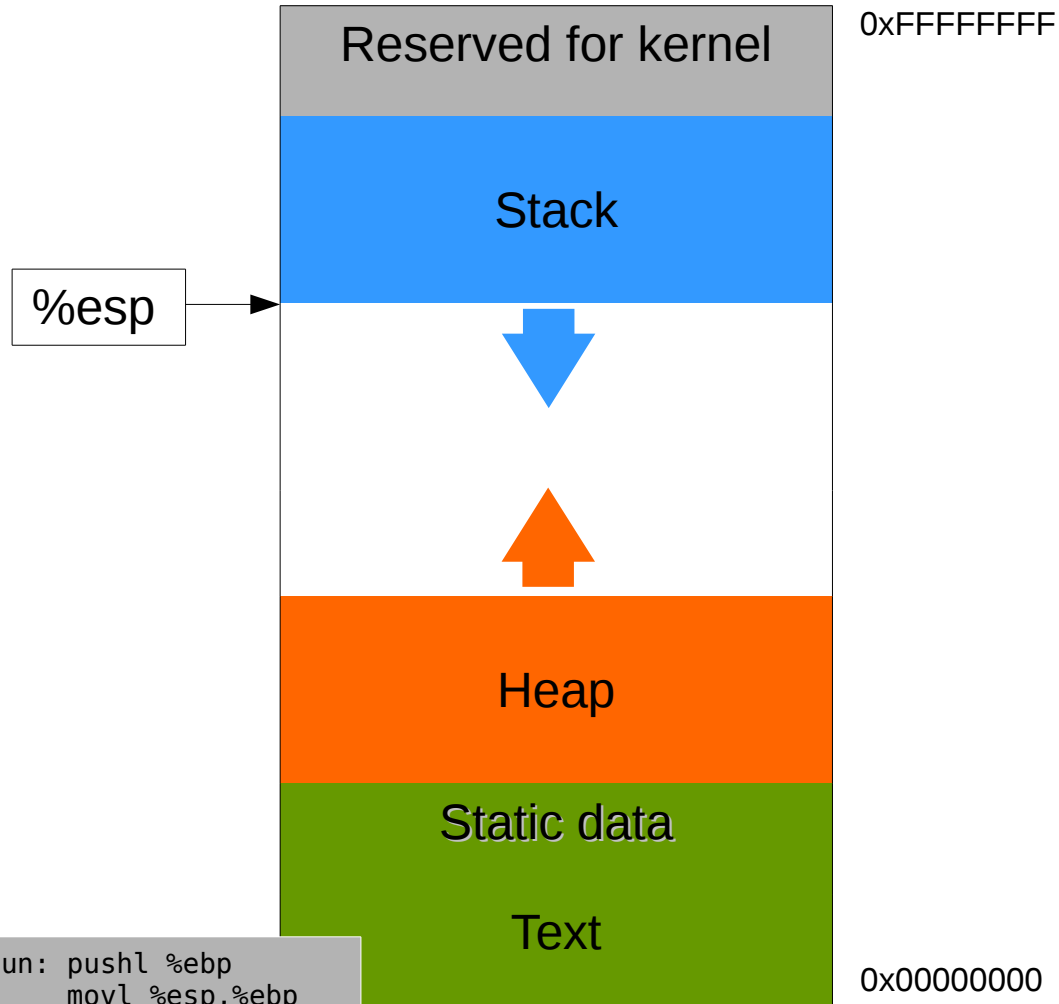


x86 CPU/Memory



- To actually be executed the program must be loaded into RAM and uniquely identified
- The RAM memory allocated to a process is its address space
- It contains both the code for the running program, its input data, and its working memory
- CPU interprets instructions - `%eip` points to next instruction

x86 process memory layout (simplified)

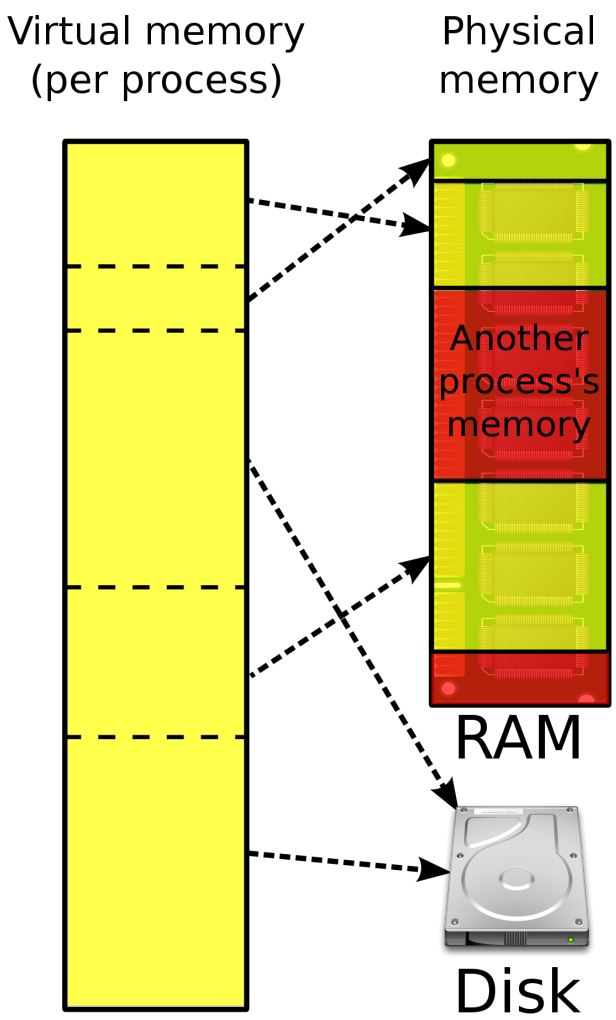



```
void function(int a, int b, int c) {
     char buffer1[5];
     char buffer2[10];
}
void main() {
     function(1,2,3);
}
```

```
fun: pushl %ebp
     movl %esp,%ebp
     subl $20,%esp

main: pushl $3
      pushl $2
      pushl $1
      call fun
```

Virtual memory



- Common technique used in a computer's OS
- Sometimes available RAM is not enough to run several programs at one time. This is where virtual memory comes in.
- A system using virtual memory uses a section of the hard drive to emulate RAM - secondary memory treated as though it were main memory
- A memory management unit (MMU) maps a logical address space to a corresponding physical address.

Live CD attacks on memory

The attack:

1. Attacker with physical access to computer powers off the computer (without properly shutting down)
2. Attacker boots to different OS via external media
3. Attacker retrieves the `Pagefile.sys`, `Swapfile.sys`, `Hiberfile.sys` files
4. Attacker gains access to passwords and sensitive information that were stored in memory

Live CD attacks on memory

The attack:

1. Attacker with physical access to computer powers off the computer (without properly shutting down)
2. Attacker boots to different OS via external media
3. Attacker retrieves the `Pagefile.sys`, `Swapfile.sys`, `Hiberfile.sys` files
4. Attacker gains access to passwords and sensitive information that were stored in memory

Mitigation:

Hard disk encryption must be used !

Security principles

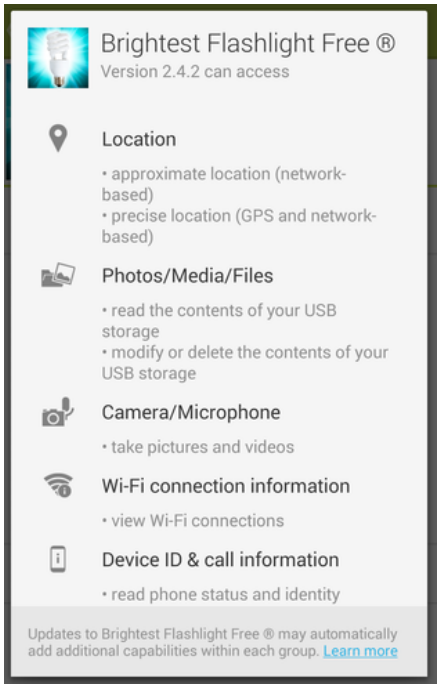
Defence-in-depth

- Security protections built in multiple layers of the system: if one mechanism fails, another steps up immediately behind to thwart attacks
- Firewalls, intrusion detection and protection systems, network segmentation, anti-virus, least privilege, strong passwords, patch management

Least privilege

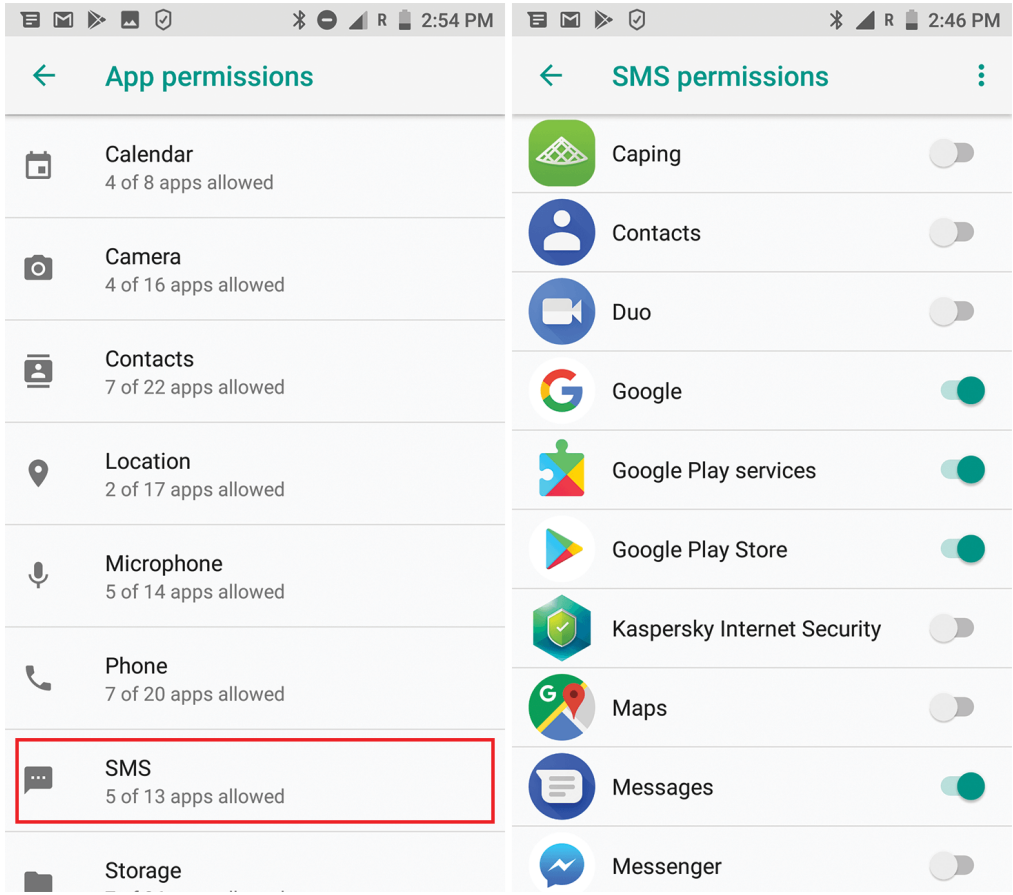
- Users and programs should only access the data and resources required to perform its function

Least privilege



- Users and programs should only access the data and resources required to perform its function
- A torch application does not need access to your location, photos, camera, microphone, wifi, device id, to perform its intended task!

Privilege separation



- Segment the system into components to which we can limit access
- Will limit the damage caused by a security break of any individual component

Open design

- The security of a mechanism should not depend on its secrecy
- The design and implementation details always get leaked (!)

Economy of mechanism

- When designing a security mechanism keep it simple!
- It will facilitate the job of security researchers and allow verification
- It will facilitate the task of developers and avoid bugs
- It will facilitate the life of users and avoid misuses

More security principles

Fail-safe defaults - default configuration should be conservative, *eg.* new user should be granted least privileges by default

Complete mediation – every access to a resource must be checked for compliance with security policy

Usable security – UIs and security mechanisms should be designed with the ordinary user in mind – the users should be supported in interacting in a secure way with the system – you can't blame users !

...

What we learned today

Many tasks handled by the OS relate to fundamental security problems.

1. OS concepts
   - basic tasks of the OS
   - security concerns arise from multiple users and multiple processes
   - processes and process management
   - x86 runtime memory

2. Security design principles
   - Defence-in-depth
   - Least privilege
   - Privilege separation
   - Open design
   - Economy of mechanism