Lec 27: Security

- security, trust, TCB
- security properties: CIA
- security mechanism taxonomy: "Gold Standard"
- Authorization
  - ACL
  - Capabilities
  - DAC/MAC

What is security? What does "the system is secure" mean?

- only some users can access system.
- processes control their own behavior
- secrecy of data protected.

} Violations should be "impossible" or "infeasible"

A system is secure if it meets its (specification)
↳ intended behavior.

safety
liveness

We want to preserve

CIA

- Confidentiality: (data) only be viewed / learned by "trusted" parties
- Integrity: data can only be modified / influenced by trusted parties
- Availability: (data) system can be accessed by trusted parties

---

Integrity affect Conf.: - can change metadata that controls access to data, can learn data.
- can change code: change behavior of trusted program.

Conf. violations can reduce integrity:
- can learn password / private key: use that to influence trusted program.

Trust: p trusts q if q has the ability to violate the security of p (i.e. p's specification).
(trust is just an assumption)

- q is **trustworthy** if it doesn't violate p's expectations.

- **trusted computing base**: set of components you trust.

  - Prog. language
    - compiler/interp.
    - libraries
    - users/system admins: install properly, use properly.
    - Operating system
      - isolation
      - access control (e.g. read only files)
    - hardware
    - network?

  level of trust depends on
  - requirements
  - assumptions (e.g. don't assume network has Conf/Int. guarantees)

TCB should be:

- small
  - fewer components
  - more verifiable
- trustworthy
- simple
  - only cover things you need.

⇒ more investment in the security than in the attacks.

- widely used systems tend to be widely tested, have fewer bugs. } better
- obscure systems can make attackers job harder.

"Gold standard" of security mechanisms

- **Authentication**: verifying the identity of user.
- **Authorization**: checking whether requests are allowed.
- **Audit**: make it possible to review actions that system performed (take corrective action, revoke authorization).

Authorization in OS

- principal: person / user, process, component (e.g. keyboard)
  - can perform actions
  - can trust or be trusted.
- objects (e.g. files, network connections, pages, ...)
- operations (e.g. read/writing, deleting, ...)

Authorization: grant request for operation on object by principal?

Access control matrix

objects →

principals ↓	/	/home/mdg	/home/mdg/solutions.txt	/bin/sh
mdg	r,w	r,w	r,w	r,x
eqs			r,w	r,x
tyb			r	r,x

↑ allowed operations

principle of least privilege:
principals should be given as little power as necessary to do their job.

Here: access control matrix is sparse.

Store efficiently:

- row by row: each principal has a list of allowed operations — Capabilities.
  - easy to look at principal see what caps it has. revoke.
  - hard to determine access.
  - Can manage with crypto, transferrable.
- column by col: each object has a list of who can access / how — Access control list (ACL)
  - easy to review who has control
  - hard to look at user, revoke access across the board.

Unix file system:

each file has - an owner, group.

- permissions: owner group public (everyone)

r w x r w x r w x

(simple form of access control list)

each process has an owner, group

- can access any file according to

NTFS (windows file system):

each file has a list of users & perms.

- read, write, execute, delete, create new files (if dir)

Discressenary access control (DAC)
- user decides permissions, appropriate use of data.

- Mandatory Access Control (MAC)
  - data has specifications for correct use.
    e.g. government classification.
    just because you have access to data doesn't mean you can share it.

  - policies that say: if a process ever reads file A, never allowed to write to file B (or network).

    - could prevent (secret / public) direct flow: copy data from A to B.

    - also want to avoid

      if secret & 0x1 : output "x" ↩ implicit flow
      else output "y"

    Covert channel:
    - if secret & 0x1 ;
        sleep (1s)
      else
        sleep (0s)
      output x