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".

Safety

A system is secure if it meets its specification.

Liveness

Intended behavior.
We want to preserve

* Confidentiality: only be viewed/learned by trusted parties

* Integrity: data can only be modified/enhanced by trusted parties

* Availability: system can be accessed by trusted parties

Integrity affects 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 & Q has the ability to violate the security of P (i.e., P's specification) if it does not violate P's

- **TCED should be:**
  - Small
  - Simple, easy to use and operate
  - Cost-effective
  - Ensure the security of the system

- More investment is needed for wider adoption

- The security of widely used systems is widely trusted

- TRUSTWORTY

- User control

- Secure systems

- Can make attacks more difficult
"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., kernel)
- can perform actions
- can trust or be trusted
- objects (e.g., files, network connections, pages,...)
- operations (e.g., read/write, delete,...)

Authorization: grant request for operation on object by principal?

Access control matrix

<table>
<thead>
<tr>
<th></th>
<th>obj0</th>
<th>obj1</th>
<th>obj2</th>
<th>obj3</th>
</tr>
</thead>
<tbody>
<tr>
<td>user0</td>
<td>r,w</td>
<td>r,w</td>
<td>r,w</td>
<td>r,x</td>
</tr>
<tr>
<td>user1</td>
<td>r,w</td>
<td>r,w</td>
<td>r,x</td>
<td>r,x</td>
</tr>
<tr>
<td>user2</td>
<td>r,w</td>
<td>r,x</td>
<td>r,x</td>
<td>r,x</td>
</tr>
</tbody>
</table>

allowed operations

principle of least privilege
- principle should be given as little power as necessary to do their job.
- here: access control matrix
  store efficiently:
  row by row: each principal has a list of allowed operations
  column by column each object has a list of who can access/own

Capabilities
- easy to look at principal see what it has control
  - hard to determine access
  - easy to revoke when has control
  - hard to look at user, revoke access across the board

Can manage with Crypto, Transferable
Unix file system:
  each file has an owner, group.
  permissions: owner group public:  
                rw- rw- rw- ...
  (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)
Discriminatory access control (DAC)
- user decides permission, 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 prepend secret public directory: copy data from A to B.
- also want to avoid
  if secret & 0x1
  output "x" (implies flow to output y).

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