Today

• Security policies
• Enforcement
• Authenticating people
• Passwords
Security policy

• *Security policies* prescribe what must be done and what must not be done by *principals* (i.e., people, computers, executing programs).

• Security policies are typically formulated in terms of the three basic kinds of *security properties*:
  – **Confidentiality**. Which principals are allowed to learn what information.
  – **Integrity**. What changes to the system (stored information and resource usage) and to its environment (outputs) are allowed.
  – **Availability**. When must inputs be read or outputs produced.

These classes are not completely independent.
Confidentiality

• An operating system restricts which files and directories each principal can read.
• Reading an object is only one way to learn information about that object.
• Inference is another.
  – Through *information flow*, a principal might learn the value of one variable by reading another.

  ```
  if sec>0 then x=1 else x=2;
  pub=x
  ```

• Another way to learn information is by measuring some aspect of system behavior, called a *covert channel*, known to be correlated with secret information.
Privacy

• The right of an individual to determine what personal information is communicated to which others, when, and for what reason.

• For computing systems, privacy often is concerned with *personally identifiable information* (PII).
  - PII encompasses information that potentially can be used to identify a person.
  - Examples: name, social security number, telephone number, address.
Integrity

• Integrity properties proscribe "bad things" from occurring during execution.
• Integrity properties can be used to convey proscriptions about data and how it is changed.
• To enforce such properties, operating systems provide control over write and execute access to files and memory regions.
• This control is not always enough to prevent low-integrity data from contaminating high-integrity data.
• Alternative: information flow control. It can
  – defend against malicious code downloaded from the Internet,
  – defend against buffer-overflow attacks.
Availability

• A “good thing” should happen during execution.

• Examples: program correctness, responsiveness

• Needed for:
  – Business through web,
  – Critical infrastructures.
Enforcement

Strategies for enforcing security policies:

• Isolation
  – Examples: Virtual Machines, Sandboxes, Processes, Firewalls

• Monitoring
  – Complete Mediation. The monitor intercepts every access to every object.
  – Least Privilege. A principal should be only accorded the minimum privileges it needs to accomplish its task.
  – Separation of Privilege. Different accesses should require different privileges.

• Recovery
Security through Accountability

Complete Mediation and:

- **Authorization.** An authorization mechanism governs whether requested actions are allowed to proceed.

- **Authentication.** An authentication mechanism associates a principal with actions.

- **Audit.** An audit mechanism records system activity, attributing each action to some responsible principal.
Authentication for People

• **Something you know.** You demonstrate knowledge of a secret or fact(s) unlikely to become known to impersonators.

• **Something you have.** You demonstrate possession of some distinctive object that is difficult for an impersonator to obtain or fabricate.

• **Something you are.** You allow certain of your physical attributes to be measured, believing that corresponding measurements will not be similar for impersonators.
Storing Passwords

• The obvious scheme for storing passwords is to use a file that contains the set of pairs <user, pwd>.

• What if the password file is compromised?
• Compute a cryptographic hash function H(pwd) for each password pwd and store the set of pairs <user ,H(pwd)> as the password file.

• Vulnerable to offline attack.
  – A program computes the hashes of passwords that people are likely to pick and compares them with the hashes in the password file.

• Salt
  – Store with each user name a nonce n, called salt, and combine that nonce with pwd before computing cryptographic hash function H().
  – The password file now stores a set of triples, <user, n, H(pwd n)>.  
  – Early versions of Unix used 12-bit numbers for salt; the nonce for a given user was obtained by reading the real-time system clock when creating the account for user.

• Pepper
  – We might keep the salt secret by storing a set of pairs <user, H(pwd n)>, where nonce n, now called the pepper, is not stored elsewhere in the tuple for user.
  – Pepper n is picked from a standard enumeration of possible pepper values.
Today

• Security policies
• Enforcement
• Authenticating people
• Passwords
Coming up...

• Next lecture: Security (2)
• HW5: due tonight
• Review on Friday
• No class on Monday
• Final exam on Tuesday