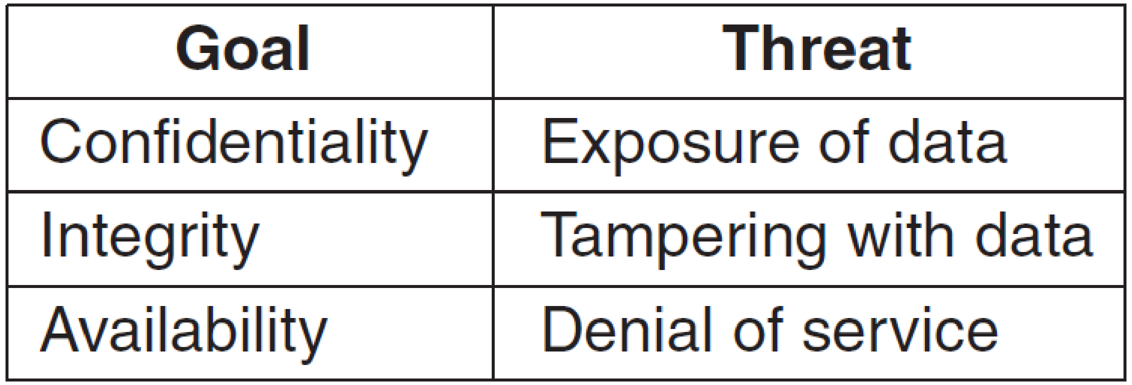
Security (Part 1)

Operating Systems

The Security Environment

- **Confidentiality** - if the owner of data decides to make available only to certain people and no others, the system should guarantee that release of the data to unauthorised people never occurs
- **Integrity** - unauthorised users should not be able to modify any data (changing the data, removing data and adding false data) without the owner's permission
- **Availability** - nobody can disturb the system to make it unusable, such as in the form of **denial-of-service** attacks that are increasingly common



Goal	Threat
Confidentiality	Exposure of data
Integrity	Tampering with data
Availability	Denial of service

Intruders

- Common categories:
  - Casual prying by nontechnical users.
  - Snooping by insiders.
  - Determined attempts to make money.
  - Commercial or military espionage.

Operating System Security

- Often the ways to compromise the security of a computer system are not very sophisticated.
  - E.g. easy to guess passwords, writing down passwords
- Exploiting such behaviours of humans, social engineering, is a significant challenge.
  - E.g. requirement to frequent password change vs. writing down passwords
- However, operating systems should also account for targeted attacks that are more sophisticated in nature, targeting the security framework of operating systems.

Operating System Security

- Passive attacks
  - try to steal information passively
  - sniff the network traffic and tries to break the encryption to get to the data
- Active attacks
  - try to make a computer program misbehave
  - take control of a user’s Web browser to make it execute malicious code

Operating System Security

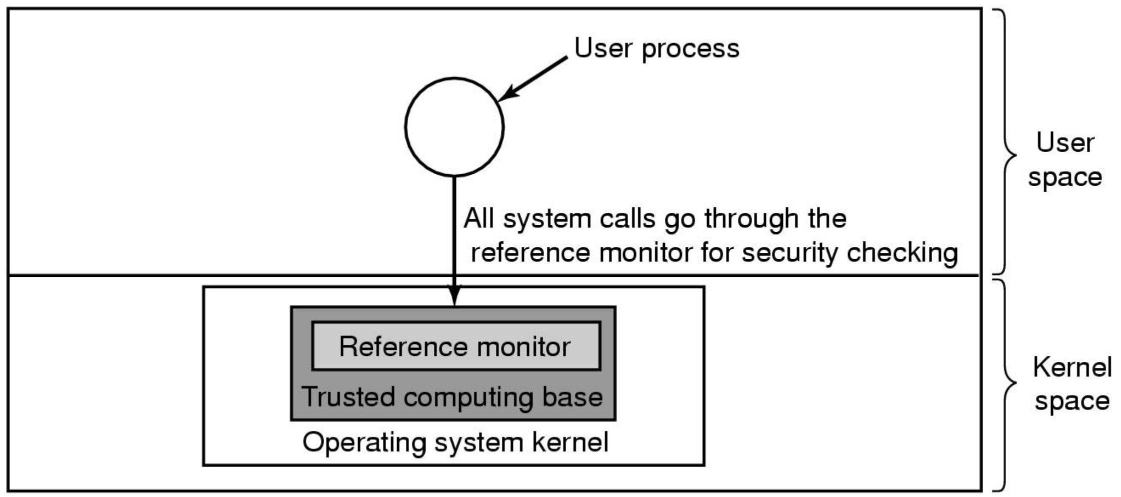
- **Cryptography**
  - shuffling a message or file in such a way that it becomes hard to recover the original data unless you have the key
  - to transmit data securely over the network, to store files securely on disk, to scramble the passwords in a password file, etc.
- Software **hardening**
  - adds protection mechanisms to programs to make it hard for attackers to make them misbehave
  - to prevent attackers from injecting new code into running software, to make sure that each process has exactly those privileges it needs to do what it is supposed to do and no more, etc.

Can we build secure systems?

- Is it possible to build a secure computer system?
  - In principle, software can be free of bugs and we can even verify that it is secure—as long as that software is not too large or complicated.
  - Unfortunately, computer systems today are horrendously complicated.
- If so, why is it not done?
  - People are not willing to leave what they are using, even if it's not secure
  - The only known way to build a secure system is to keep it simple. Features are the enemy of security.
  - But, today's feature-rich software have more complexity, more code, more bugs, and more security errors.

Trusted Computing Base (TCB)

- In the security world, people often talk about **trusted systems** rather than secure systems.
- These are systems that have formally stated security requirements and meet these requirements.
- At the heart of every trusted system is a minimal **TCB** (**Trusted Computing Base**) consisting of the hardware and software necessary for enforcing all the security rules.
- If the trusted computing base is working to specification, the system security cannot be compromised, no matter what else is wrong.

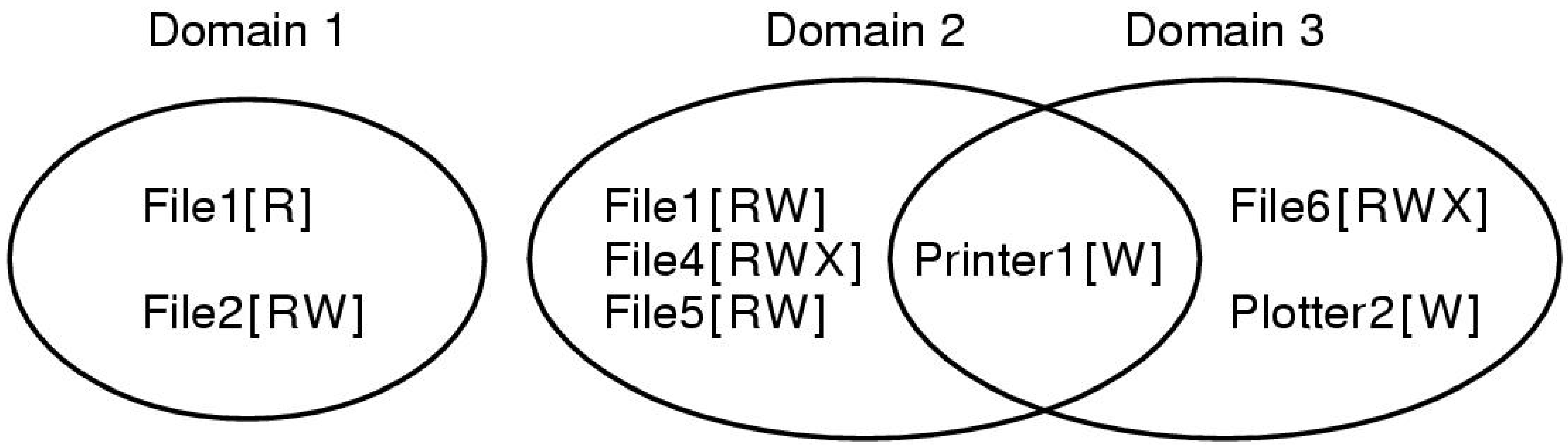


Controlling Access to Resources

- A computer system contains many resources, or ‘‘objects,’’ that need to be protected.
- These objects can be hardware (e.g., CPUs, memory pages, disk drives, or printers) or software (e.g., processes, files, databases, or semaphores).
- A model of what is to be protected and who is allowed to do what is necessary for the operating system.
- There are various models for doing this,
    1. Protection Domains
    2. Access Control Lists
    3. Capabilities

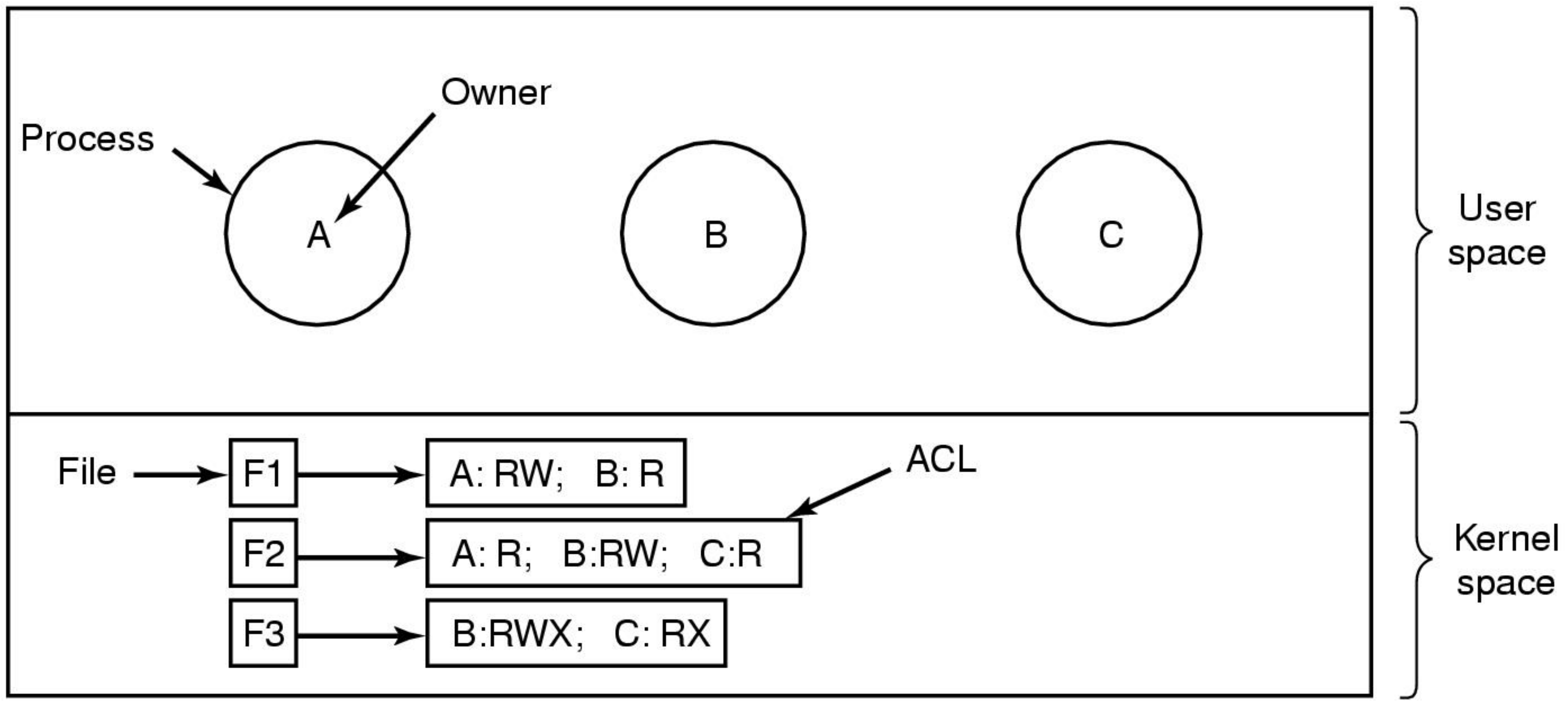
Protection Domains

- A **domain** is a set of (object, rights) pairs.
- Each pair specifies an object and some subset of the operations that can be performed on it.
- A **right** in this context means permission to perform one of the operations.
- E.g. Unix/Linux file permissions with UID/GID



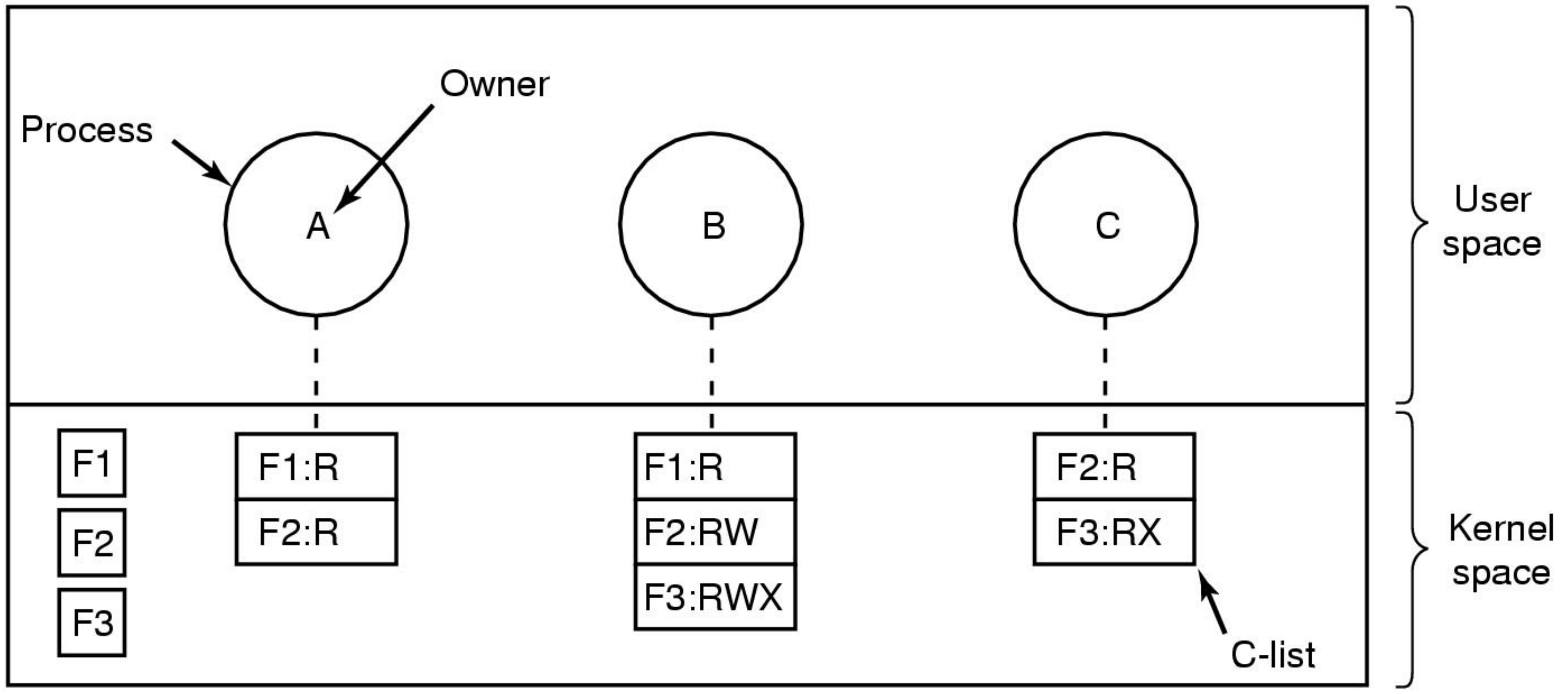
Access Control Lists

- An **Access Control List (ACL)** consists of associating with each object an (ordered) list containing all the domains that may access the object, and how.



Capabilities

- A **capability list** (or **C-list**) is a list of objects associated with each process that may be accessed, along with an indication of which operations are permitted on each, in other words, its domain.

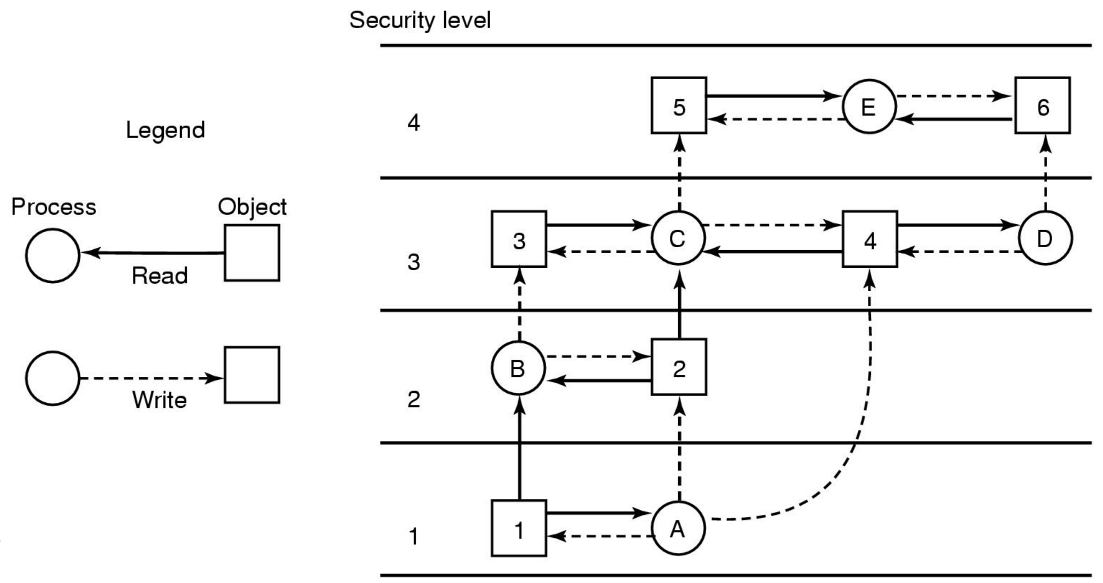


Formal Models of Secure Systems

- **Multilevel Security**
  - **discretionary access control**
    - operating systems allow individual users to determine who may read and write their files and other objects
  - **mandatory access controls**
    - the organization has stated rules about who can see what, and these may not be modified by individuals
  - operating systems must enforce the stated security policies, in addition to the standard discretionary access controls

Formal Models of Secure Systems

- **The Bell-La Padula Model**
- Rules for the Bell-La Padula model:
  - **The simple security property**:
    A process running at security level k can read only objects at its level or lower.
  - **The * property**: A process running at security level k can write only objects at its level or higher.



Formal Models of Secure Systems

- **The Biba Model**
- Rules for the Biba model:
  - **The simple integrity principle**: A process running at security level k can write only objects at its level or lower (no write up).
  - **The integrity * property**: A process running at security level k can read only objects at its level or higher (no read down).