Access Control and Operating System Security

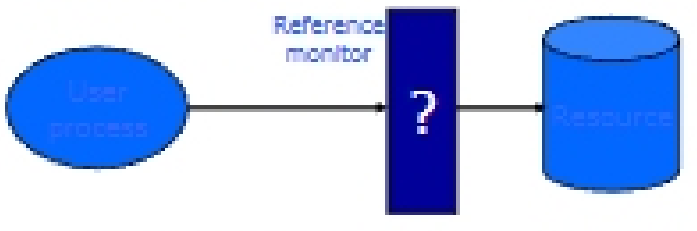
John Mitchell

Outline (may not finish in one lecture)

- Access Control Concepts
  - Matrix, ACL, Capabilities
  - Multi-level security (MLS)
- OS Mechanisms
  - Multics
    - Ring structure
  - Amoeba
    - Distributed, capabilities
  - Unix
    - File system, Setuid
  - Windows
    - File system, Tokens, EFS
  - SE Linux
    - Role-based, Domain type enforcement
- Secure OS
  - Methods for resisting stronger attacks
- Assurance
  - Orange Book, TCSEC
  - Common Criteria
  - Windows 2000 certification
- Some Limitations
  - Information flow
  - Covert channels

Access control

- Common Assumption
  - System knows who the user is
    - User has entered a name and password, or other info
  - Access requests pass through gatekeeper
    - OS must be designed monitor cannot be bypassed



Decide whether user can apply operation to resource

Access control matrix [Lampson]

Objects

Subjects	File 1	File 2	File 3	...	File n
User 1	read	write	-	-	read
User 2	write	write	write	-	-
User 3	-	-	-	read	read
...					
User m	read	write	read	write	read

Two implementation concepts

- Access control list (ACL)
  - Store column of matrix with the resource
- Capability
  - User holds a "ticket" for each resource
  - Two variations
    - store row of matrix with user
    - unforgeable ticket in user space

	File 1	File 2	...
User 1	read	write	-
User 2	write	write	-
User 3	-	-	read
...			
User m	read	write	write

Access control lists are widely used, often with groups

Some aspects of capability concept are used in Kerberos, ...

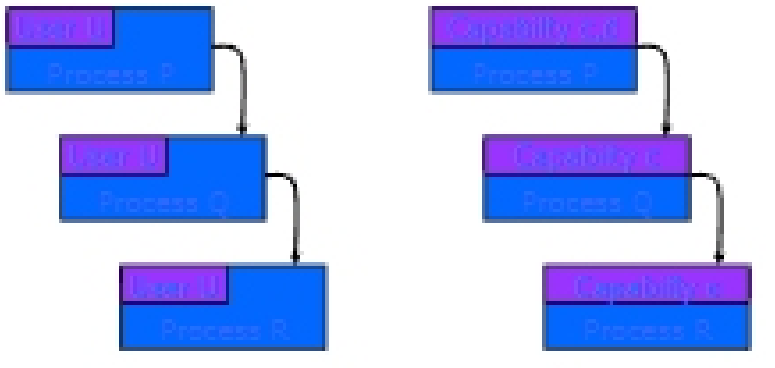
Capabilities

- Operating system concept
  - "... of the future and always will be ..."
- Examples
  - Dennis and van Horn, MIT PDP-1 Timesharing
  - Hydra, StarOS, Intel iAPX 432, Eros, ...
  - Amoeba: distributed, unforgeable tickets
- References
  - Henry Levy, Capability-based Computer Systems
    http://www.cs.washington.edu/homes/levy/capabook/
  - Tanenbaum, Amoeba papers

ACL vs Capabilities

- Access control list
  - Associate list with each object
  - Check user/group against list
  - Relies on authentication: need to know user
- Capabilities
  - Capability is unforgeable ticket
    - Random bit sequence, or managed by OS
    - Can be passed from one process to another
  - Reference monitor checks ticket
    - Does not need to know identify of user/process

ACL vs Capabilities

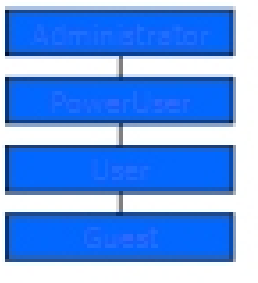


ACL vs Capabilities

- Delegation
  - Cap: Process can pass capability at run time
  - ACL: ????
- Revocation
  - ACL: Remove user or group from list
  - Cap: Try to get capability back from process?
    - Possible in some systems if appropriate bookkeeping
      - OS knows what data is capability
      - If capability is used for multiple resources, have to revoke all or none …
      - Other details …

Roles (also called Groups)

- Role = set of users
  - Administrator, PowerUser, User, Guest
  - Assign permissions to roles; each user gets permission
- Role hierarchy
  - Partial order of roles
  - Each role gets permissions of roles below
  - List only new permissions given to each role



Groups for resources, rights

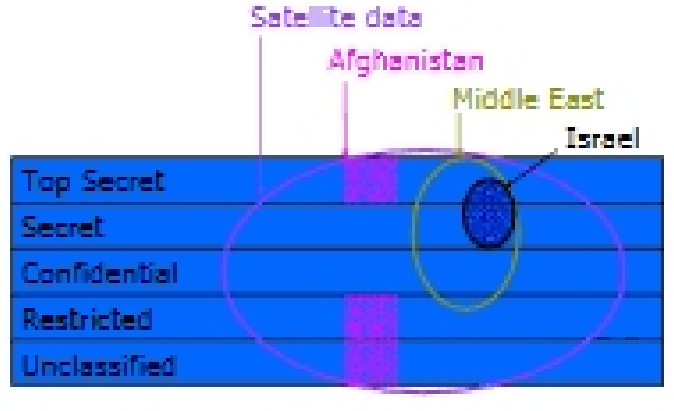
- Permission = ⟨right, resource⟩
- Permission hierarchies
  - If user has right r, and r>s, then user has right s
  - If user has read access to directory, user has read access to every file in directory
- Big problem in access control
  - Complex mechanisms require complex input
  - Difficult to configure and maintain
  - Roles, other organizing ideas try to simplify problem

Multi-Level Security (MLS) Concepts

- Military security policy
  - Classification involves sensitivity levels, compartments
  - Do not let classified information leak to unclassified files
- Group individuals and resources
  - Use some form of hierarchy to organize policy
- Other policy concepts
  - Separation of duty
  - "Chinese Wall" Policy

Military security policy

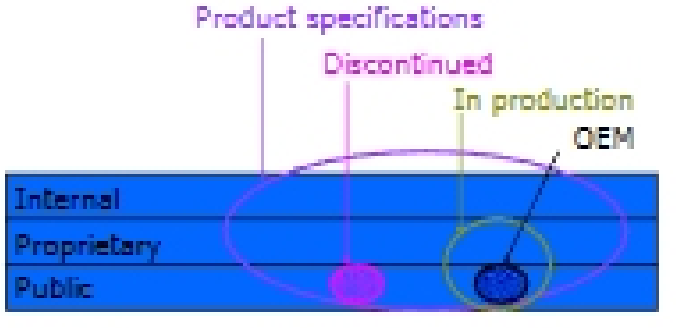
- Sensitivity levels
- Compartments



Military security policy

- Classification of personnel and data
  - Class = (rank, compartment)
- Dominance relation
  - $D_1 \leq D_2$ iff $rank_1 \leq rank_2$ and $compartment_1 \subseteq compartment_2$
  - Example: (Restricted, Israel) ≤ (Secret, Middle East)
- Applies to
  - Subjects – users or processes
  - Objects – documents or resources

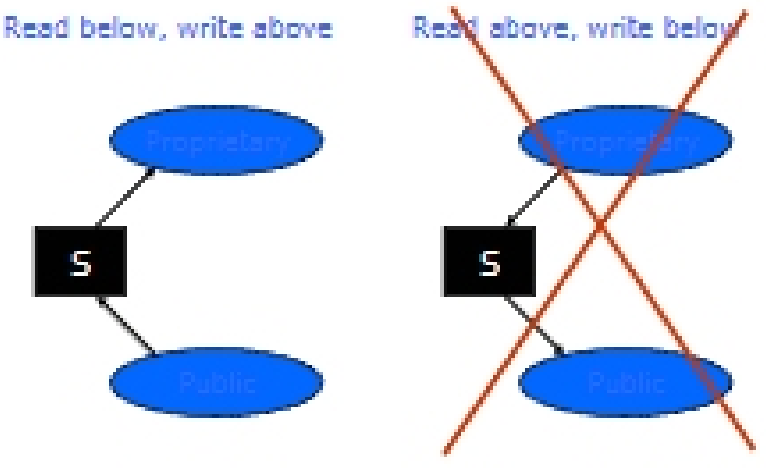
Commercial version



Bell-LaPadula Confidentiality Model

- When is it OK to release information?
- Two Properties (with silly names)
  - Simple security property
    - A subject S may read object O only if C(O) ≤ C(S)
  - *-Property
    - A subject S with read access to O may write object P only if C(O) ≤ C(P)
- In words,
  - You may only *read below* your classification and only *write above* your classification

Picture: Confidentiality



Biba Integrity Model

- Rules that preserve integrity of information
- Two Properties (with silly names)
  - Simple integrity property
    - A subject S may write object O only if C(S) ≥ C(O)
    (Only trust S to modify O if S has higher rank ...)
  - *-Property
    - A subject S with read access to O may write object P only if C(O) ≥ C(P)
    (Only move info from O to P if O is more trusted than P)
- In words,
  - You may only *write below* your classification and only *read above* your classification