Announcements

- Assignment #2
  - Due today 11:59pm
- Midterm Oct.3rd (Tuesday)
  - 9:30 – 11:30 am
  - I will in the classroom (010) 10:00-11:15am, No lecture
- Thursday office hours
  - In person: Woodward Hall 330D
  - Zoom meeting link: https://charlotte-edu.zoom.us/my/jxiang1
- UNCC security symposium
  - Extra credit: 1%
  - Go there without registration, don’t eat the food

Access Control

Slides from Prof. Dr. Mihai Chiroiu

Today's plan: Access Control

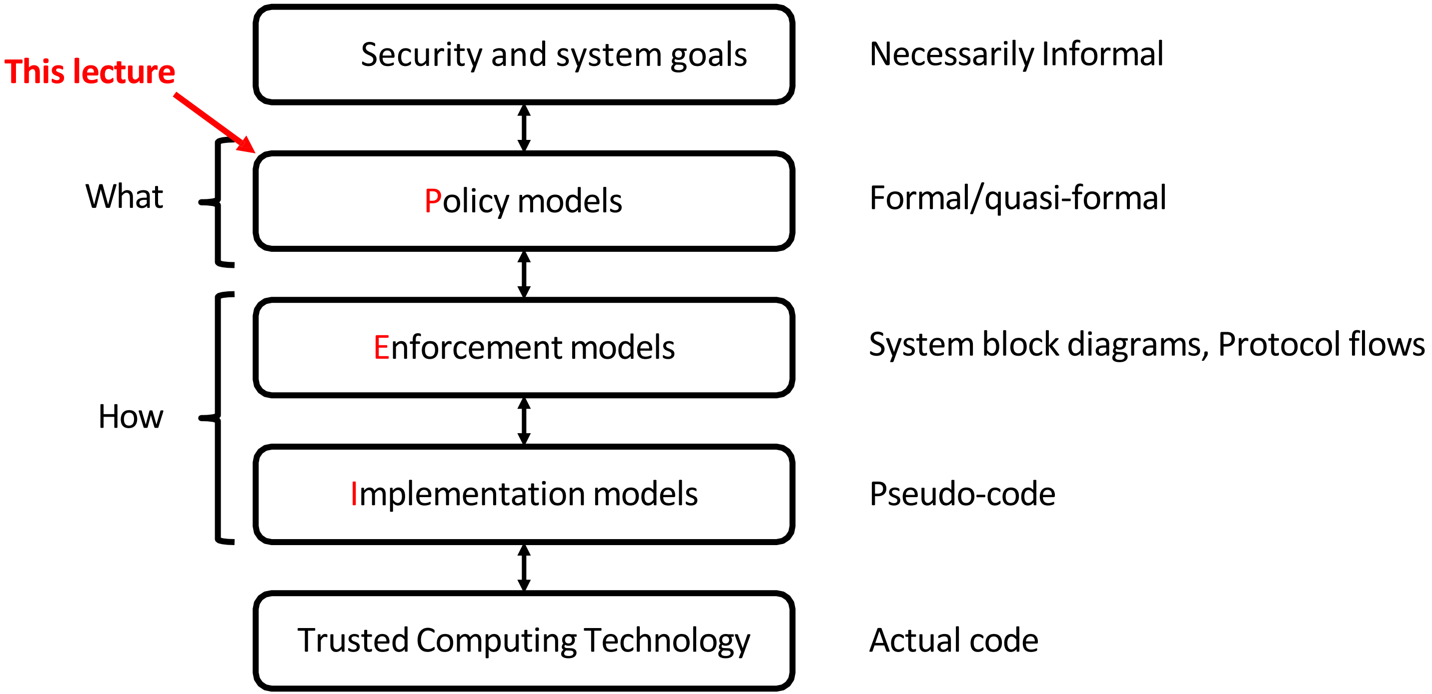
- Vocabulary
- Discretionary access controls (DAC)
- Mandatory access controls (MAC)
  - Access control models
- Role-Based Access Control (RBAC)

Examples of Access Control

- Social Networks: Access to personal information.
- Web Browsers: Access only to a website (same origin policy).
- Operating Systems: One user cannot arbitrarily access/kill another user’s files/processes.
- Memory Protection: Code in one region, cannot access the data in another more privileged region.
- Firewalls: If a packet matches with certain conditions, it will be dropped.

PEI Model

ITIS 6200 / 8200

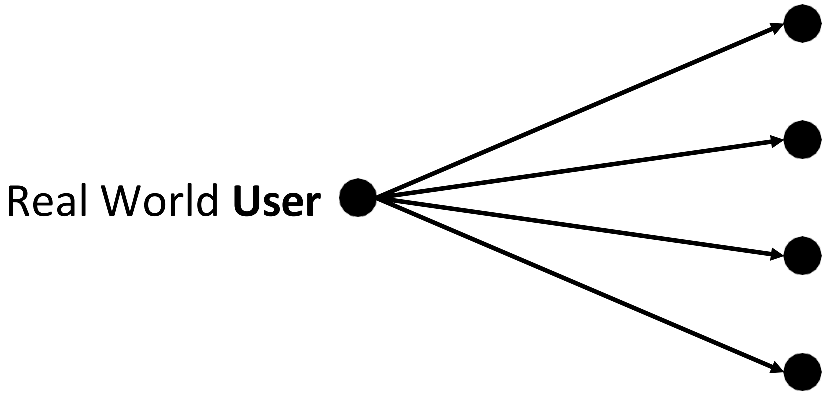


Vocabulary

- Basic abstractions:
  - **User**: human
  - **Object**: a piece of data or a resource (e.g., a file or a network packet).
  - **Subject**: an entity who wishes to access a certain **object** (e.g., a process executing on behalf of a user)
  - **Rights (permissions)**: different modes of access (e.g., reading, writing)

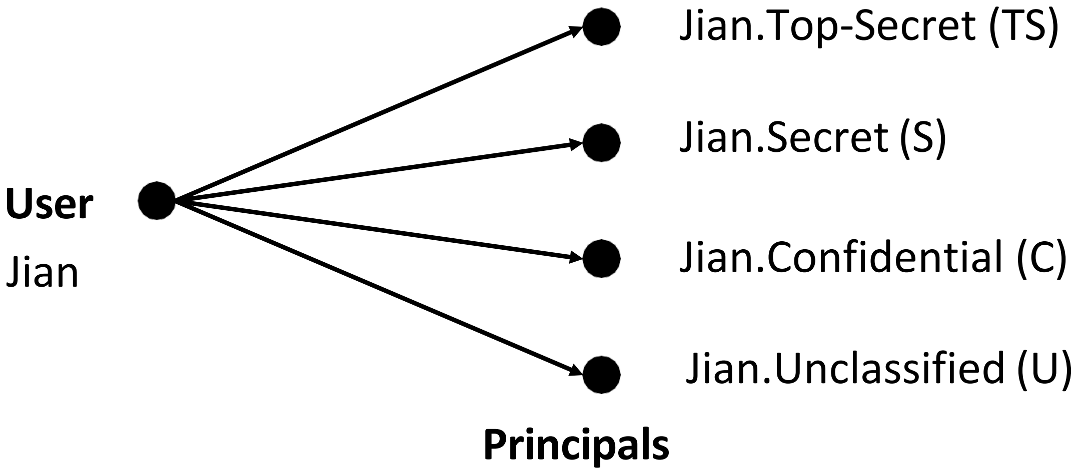
Vocabulary – Users and Principals

ITIS 6200 / 8200



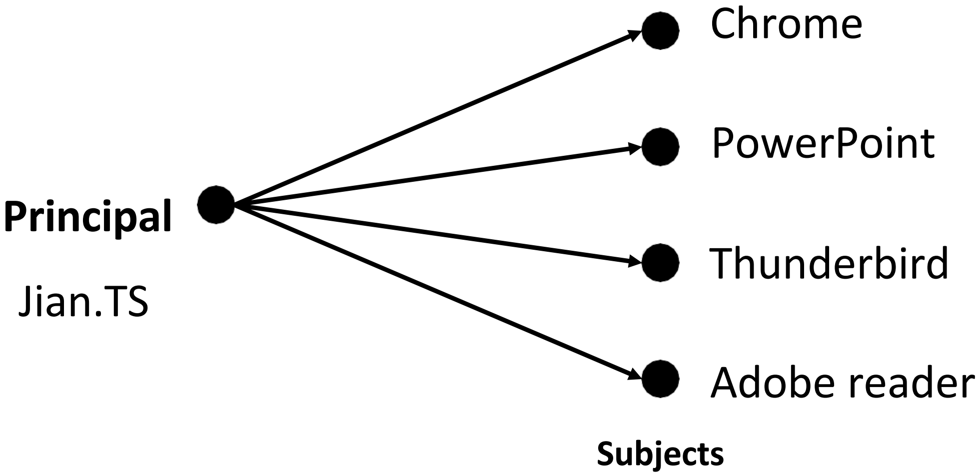
- A Principal is an User authenticated in a context

Vocabulary – Users and Principals



Example: the user generates multiple API keys

Vocabulary – Principals and subjects



A subject is a program executing on behalf of a principal

Vocabulary

- The relation between Users and Principals is One-To-Many
  - Allows accountability of user's actions, use least privileges required for a task
  - E.g., API keys: don't share your password
- For simplicity, a principal and subject can be treated as identical concepts

Vocabulary - Objects

- An object is anything on which a subject can perform operations (mediated by rights)
- Usually objects are passive, for example:
  - File
  - Directory (or Folder)
  - Memory segment
- But, subjects (e.g., processes) can also be objects, with operations
  - kill
  - suspend
  - resume

Access Control Policies

Access Control Policies

- **Discretionary access controls (DAC)** – the access of objects (or subjects) can be propagated from one subject to another. Possession of an access right by a subject is sufficient to allow access to the object.
- **Mandatory access controls (MAC)** – the access of subjects to objects is based on a system-wide policies (based on security labels) that can be changed only by the administrator.
- **Role-Based access Control (RBAC)** – can be configured as both MAC or DAC, access to objects is based on roles.

Discretionary Access Control

DAC

- No precise definition.
- The underlying philosophy in DAC is that subjects can determine who has access to their objects.
- Basically, DAC allows access rights to be propagated at subject's discretion
  - often has the notion of owner of an object
  - used in UNIX, Windows, etc.

DAC Implementation

- Let *S* be the set of all subjects, *O* the set of all objects, and P the set of all permissions. The description of access control can be given by a set $A \subseteq S \times O \times P$.

- When new permissions are added, triplets are added to A; when they are removed (revoked), triplets are deleted.

Access Control – Representation

- An access control matrix is a matrix (Ms,o) whose rows are subjects and columns are objects. Element (Ms,o) ⊆ P is the set of permissions that subject **S** is authorized for object o.

Objects (and Subjects) →

Subjects ↓

	A	B	C	D
U1				
U2		rw		kill
U3			r	
U4		r		

Access Control Lists (ACL)

An access control list is a set $\{Ao \mid o \in O\}$, one element for each **object**. The elements of the list are the pairs (s, p) of **subjects** s who have **permission** p to that object.

B
U2: rw
U4: r

C
U3: r

D
U2: Kill

Capabilities

Storing capabilities means giving to each subject tokens which give them access to the permissions they are entitled.

U1	
U2	B/rw, D/kill
U3	C/r
U4	B/r

ACL vs. Capabilities

- ACL require authentication of subjects
- Capabilities do not require authentication of subjects, but do require unforgeability and control of propagation of capabilities. Usually implemented through cryptography.

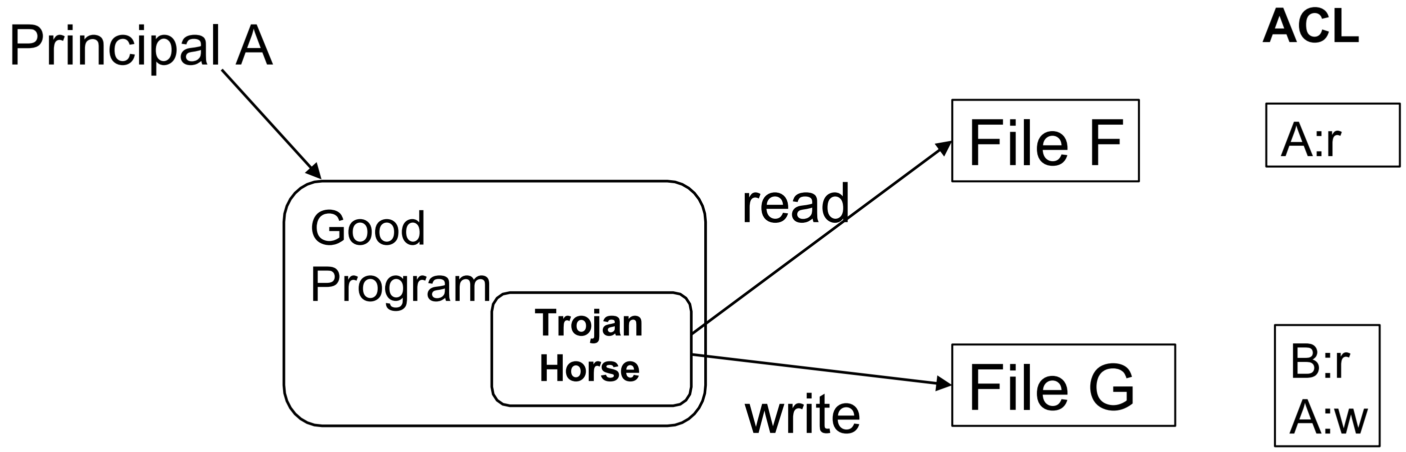
ACL vs. Capabilities Example

- Scenario:
  - ➢ Bob wishes to store valuable items in a safe box maintained by a bank. In some cases, he wants his trustworthy relatives to access the box. The bank can regulate access to Bob's box in two ways:
    - ❖ Maintain a list of persons, or
    - ❖ Issue one or multiple access keys to the box.
- ACL approach
  - ➢ Bank's role: the financial institution must have a list of account holders, verify users, and define permissions. The entity needs to maintain the list's integrity and authenticate access.
  - ➢ Adding new users: Bob must pay a visit to the bank's branch to add more users
  - ➢ Delegation: the approved third parties cannot delegate their access rights to other parties.
  - ➢ Removing users: Bob and the bank can delete names from the list.
- Capability approach
  - ➢ Bank's role: the bank is not involved
  - ➢ Adding new users: Bob can assign a key to a thirty-party
  - ➢ Delegation: key can be passed to others
  - ➢ Revoke: Bob can recall his key from the thirty-party, but it may be challenging to establish whether they made a copy.

DAC Problems

- The underlying philosophy in DAC is that subjects can determine who has access to their objects.
  - There is a difference, though, between trusting a person and trusting a program.
- The copies of file are not controlled
- The Trojan Horse attack [1970]
  - Solution: use MAC

Trojan Horse attack



Principal B cannot read file F

What does Trojan Horse do?

- Create a new object G
- Grant A write access to G
- Grant B read access to G
- Copy F to G
- Find a way to interest A, so it runs the Torjan Horse program

Buggy software can become Trojan Horses

- When a buggy software is exploited, it executes the code/intention of the attacker, while using the privileges of the user who started it

- This means that computers with only DAC cannot be trusted to process information classified at different levels

Mandatory Access Control

Modeling Access Control

- Assigning access rights based on regulations by a **central authority**

- Implemented using a *"reference monitor"*
  - Small Trusted Computing Base (TCB) [John Rushby, 1981, OSP]

- Implemented using Virtualization

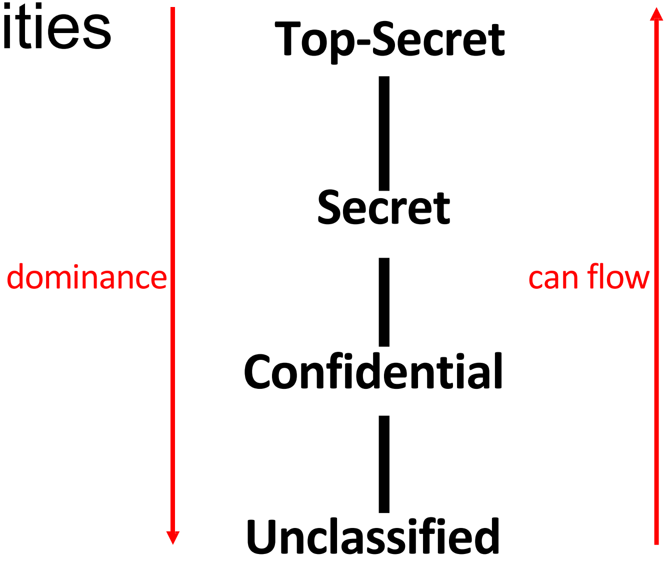
Modeling Access Control

- Multi-level security (MLS)
  - Bell-LaPadula (BLP) (Confidentiality)
  - Biba Model (Integrity)
- Chinese Wall

Multi-level security (MLS)

- The capability of a computer system to carry information with different sensitivities

- Bell-LaPadula (BLP) Model [1973]
- Biba Model



BLP Model

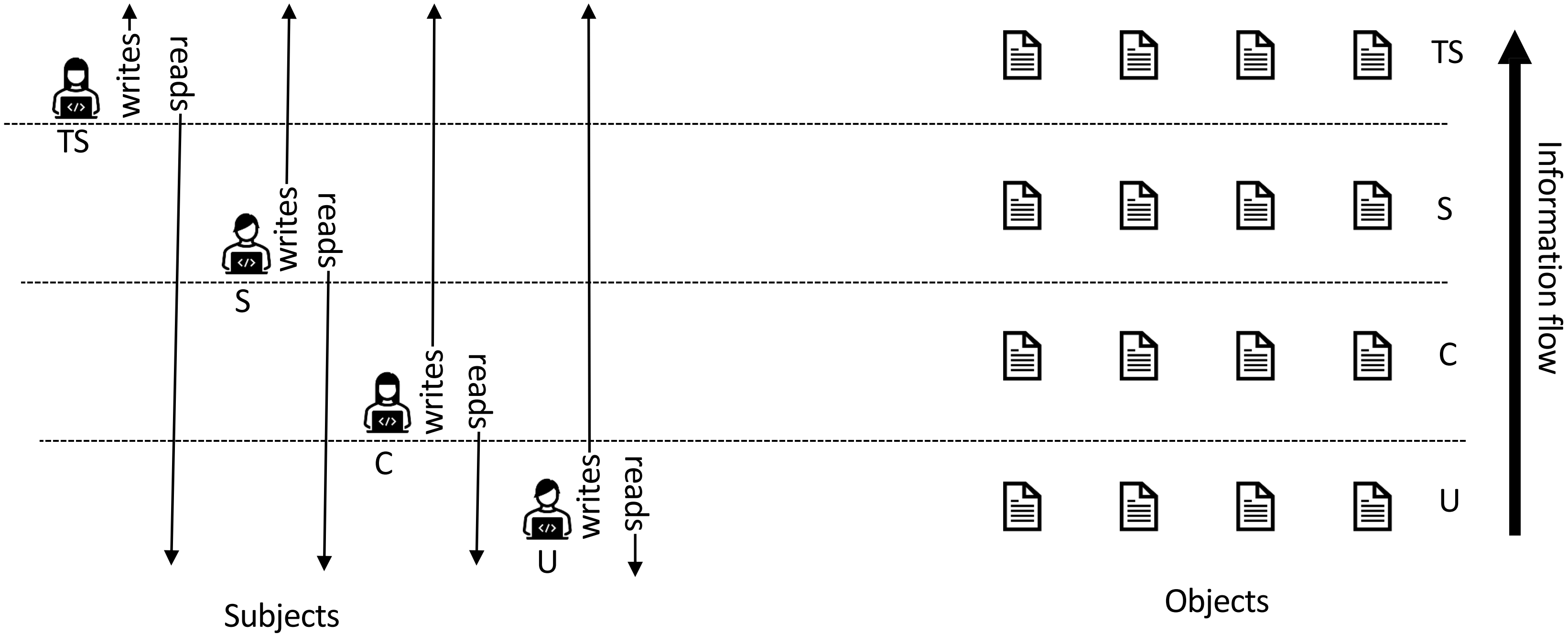
- Aims to capture confidentiality (read) requirements only
- The system is modelled as transitions through a set of states, starting from an initial state.
- State transition rules describe how a system can go from one state to another
- Each **subject** s has a maximal security level **Lm(s)**, and a current security level **Lc(s)**
- Each **object** has a classification level

BLP Model

- A state is secure if:
  - Simple Security Property (SS): no subject may read data at a higher level (NO read up)
  - The *(Star)-Property (SP): no subject may write data at a lower level (NO write down)
    - (due to the fear of Trojan Horse)
- A system is secure if and only if every reachable state is secure.

BLP Model

ITIS 6200 / 8200



No Read Up, No Write Down

BLP Problems

- Consider a system with subjects s1, s2, and objects o1, o2
  - Lm(s1) = Lc(s1) = L(o1) = Secret
  - Lm(s2) = Lc(s2) = L(o2) = Unclassified
- And the following execution
  - s1 (Secret) gets access to o1 (Secret), reads something, releases access
  - s1 changes current level to Unclassified
  - s1 gets write access to o2 (Unclassified), writes to o2
- Every state is secure, yet illegal information exists
- Solution: subject cannot change current levels, or cannot drop to below the **highest level read so far**
  - s1 cannot drop to Unclassified after reading Secret

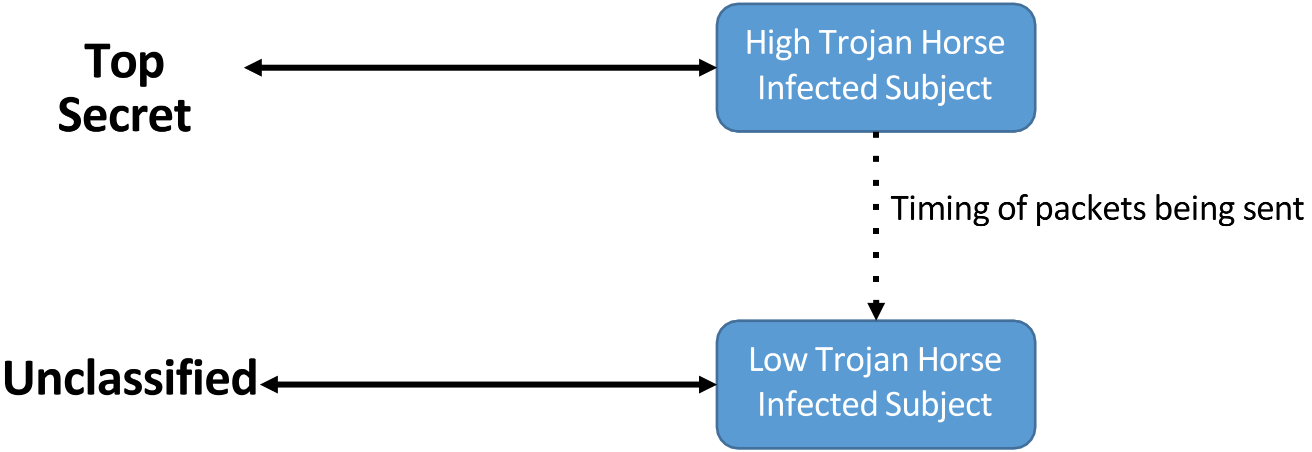
BLP Problems

- Not all system components can be enforced by BLP, e.g., memory management must have access to all levels
  - Called *“trusted subjects”*

- Can overwrite high and more important files

BLP Problems

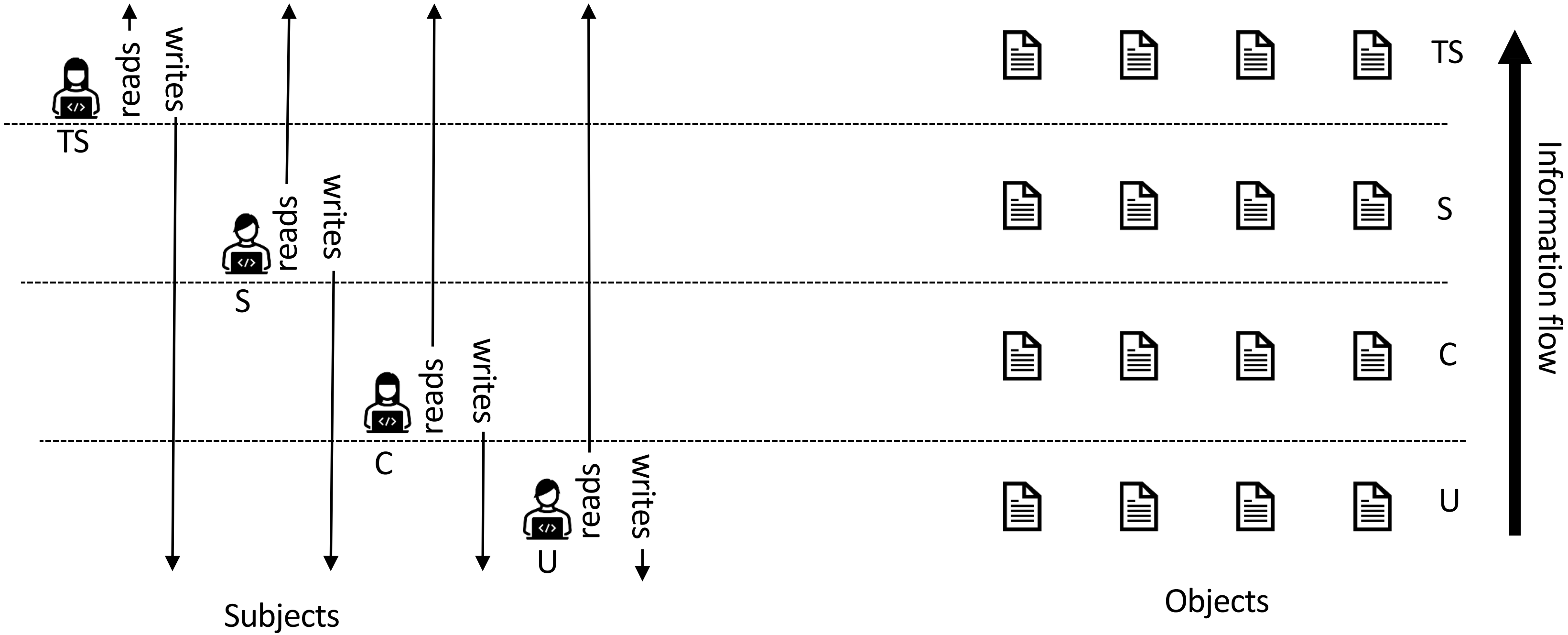
- Covert channels cannot be blocked by star-property



Biba Model

- Integrity is also very important
- Each subject (process) has an integrity level; each object has an integrity level; Integrity levels are totally ordered
- NO read down; NO write up
  - BLP upside down
- The integrity of an object is the lowest level of all the objects that contributed to its creation

Biba Model



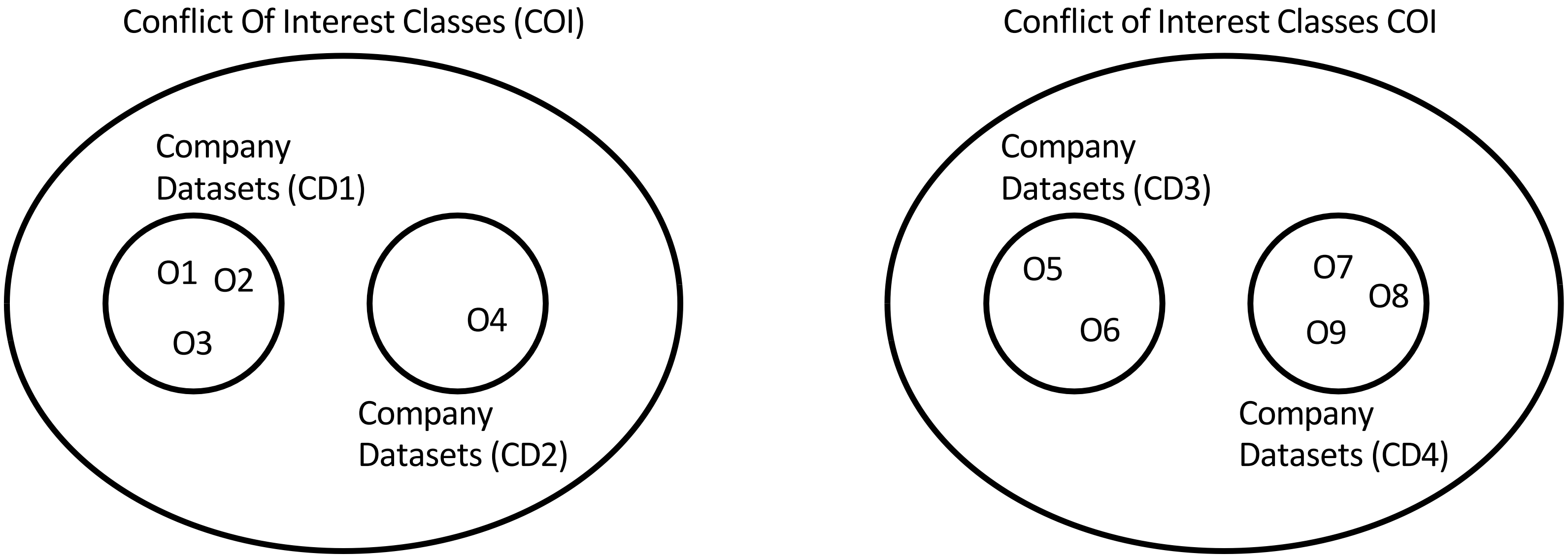
No Read Down, No Write Up

Biba Model

- Used by Windows

- E.g., A browser can download a file (created with a low integrity level) and read everything in the system. It cannot write to a higher level object.

Chinese Wall (Brewer and Nash model) [1989]

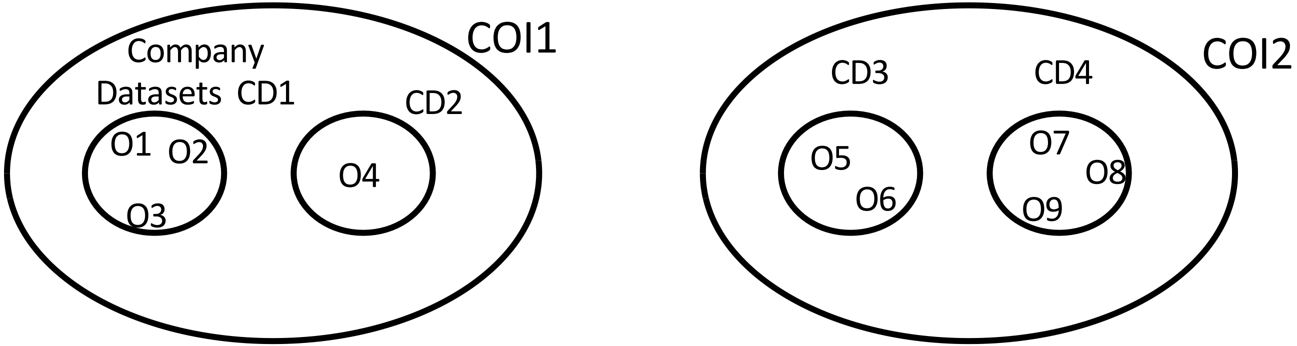


Example:

CD1 = Bank of America; CD2 = Wells Fargo;

CD3 = Ford; CD4 = GM

Chinese Wall



- S can read O only if
  - O is in the same company dataset as some object previously read by S (i.e., O is within the wall)

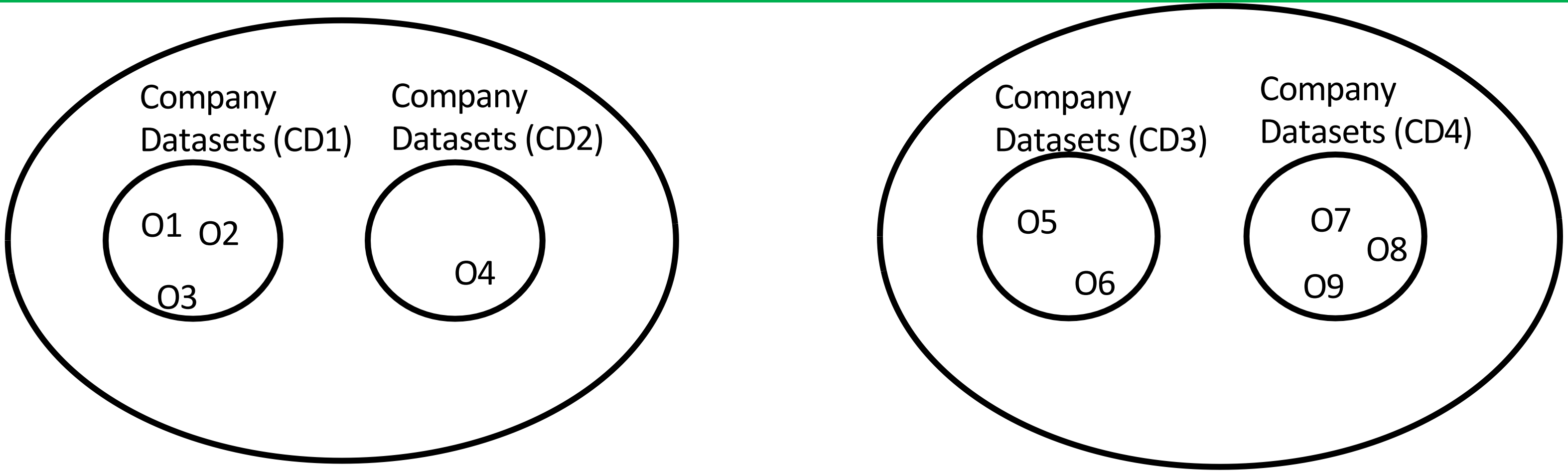
  or
  - O belongs to a conflict of interest class within which S has not read any object (i.e., O is in the open)
- S can write O only if
  - S can read O and
  - S has never read an object O' such that CD(O) ≠ CD(O')

Q: If s1 has read o1

…

Chinese Wall

ITIS 6200 / 8200



Once a subject reads two objects from different CDs, that subject may never write any object.

Consider the following scenario:

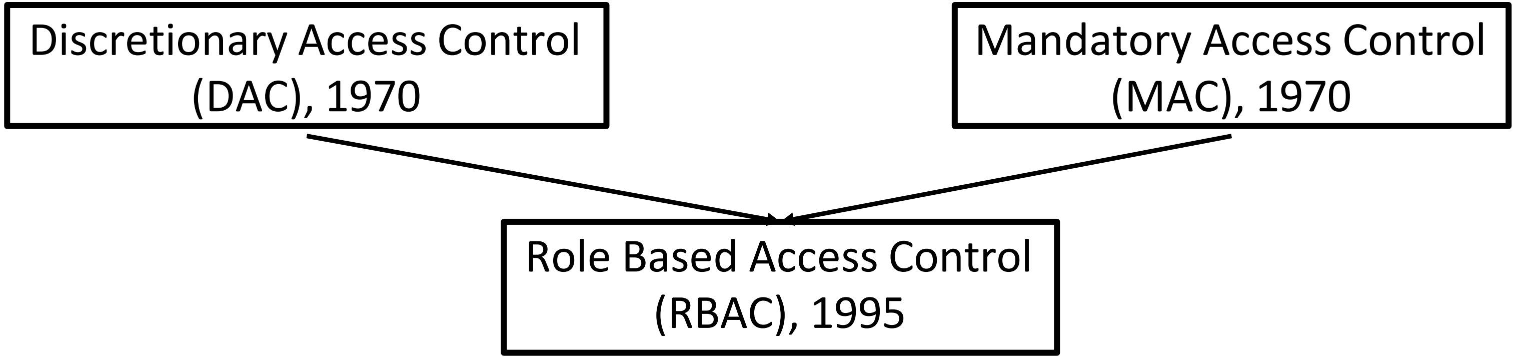
- S1 reads information from an object in CD1.
- S2 reads information from an object in CD2.
- S1 writes that information to object O6 in CD3.
- S2 reads that information from O6.

Without the security condition, S2 would have read information pertaining to both CD1 and CD2

Role-Based Access Control

Role-Based Access Control

- In the real world, security policies are dynamic.
- E.g., a user promotes at his job, therefore his rights must change (deleted, added, etc.)



Role-Based Access Control

- Can be configured to do DAC
  - roles simulate identity (RBAC98)

- Can be configured to do MAC
  - roles simulate clearances (ESORICS 96)

Role-Based Access Control

- Changes the underlying subject--object model
  - a policy is a relation on roles, objects, and rights
- Subjects are now assigned to roles;
  - *role assignment*
- Roles are hierarchical

Roles as policy

- A role brings together
  - a collection of users and
  - a collection of permissions
- These collections will vary over time
- A user can be a member of many roles
- Each role can have many users as each role can have many users as members

RBAC Shortcomings

- Role granularity is not adequate leading to role explosion
- Role design and engineering is difficult and expensive
- Assignment of users/permissions to roles is cumbersome
- Adjustment based on local/global situational factors is difficult

Resources

1 http://www.profsandhu.com/confrnc/asiaccs/asiaccs06-pei.pdf

2 http://www.cs.cornell.edu/courses/cs5430/2011sp/NL.accessControl.html

3 http://cnitarot.github.io/courses/cs526_Spring_2015/s2014_526_ac.pdf

4 https://people.cs.rutgers.edu/~pxk/419/notes/access.html