Introduction to Computer Security Lecture Slides

Access Control

Associate Prof. Dr. Mihai Chiroiu

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 (e.g., Ring 3), cannot access the data in another more privileged region (e.g. Ring 0).
- Firewalls: If a packet matches with certain conditions, it will be dropped.

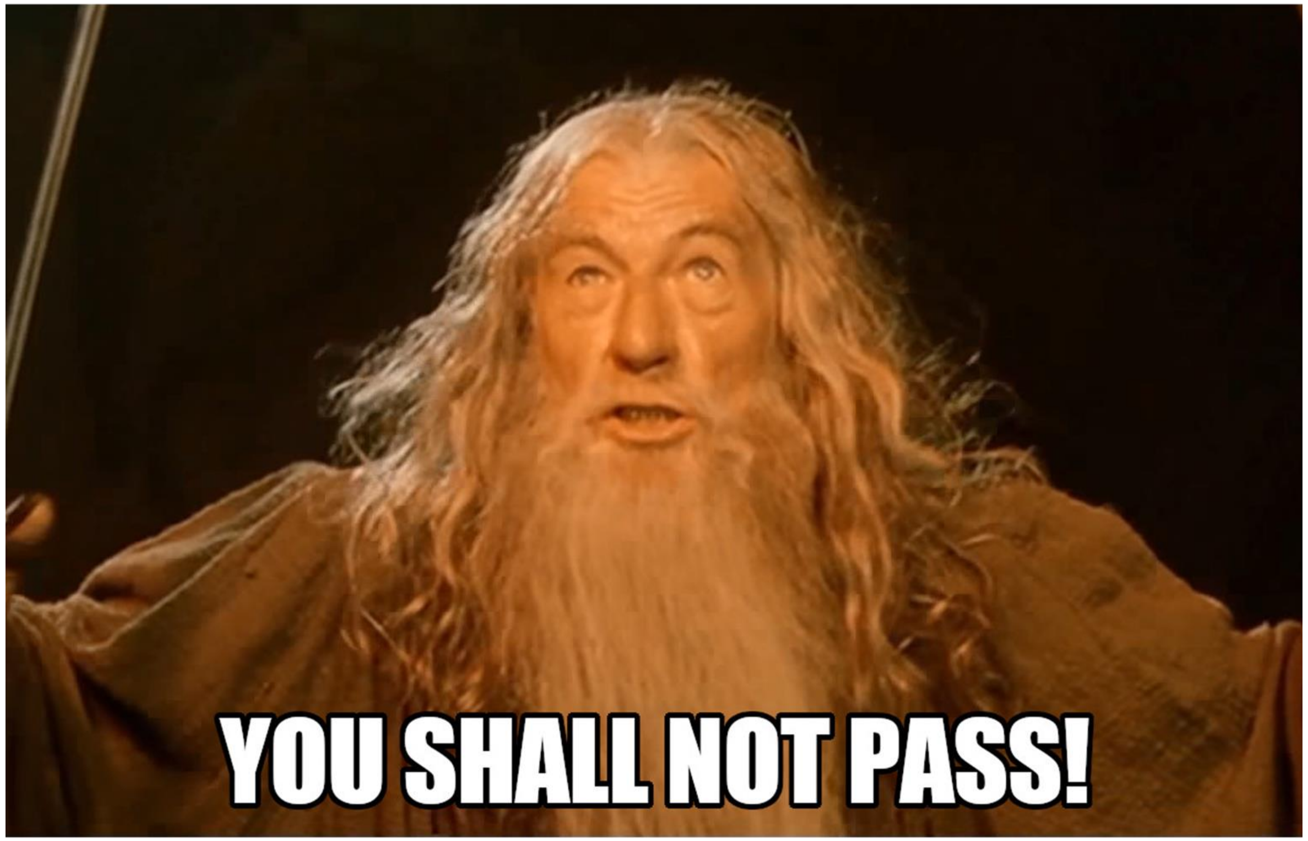
Access control

(1) Identify

(2) Authenticate

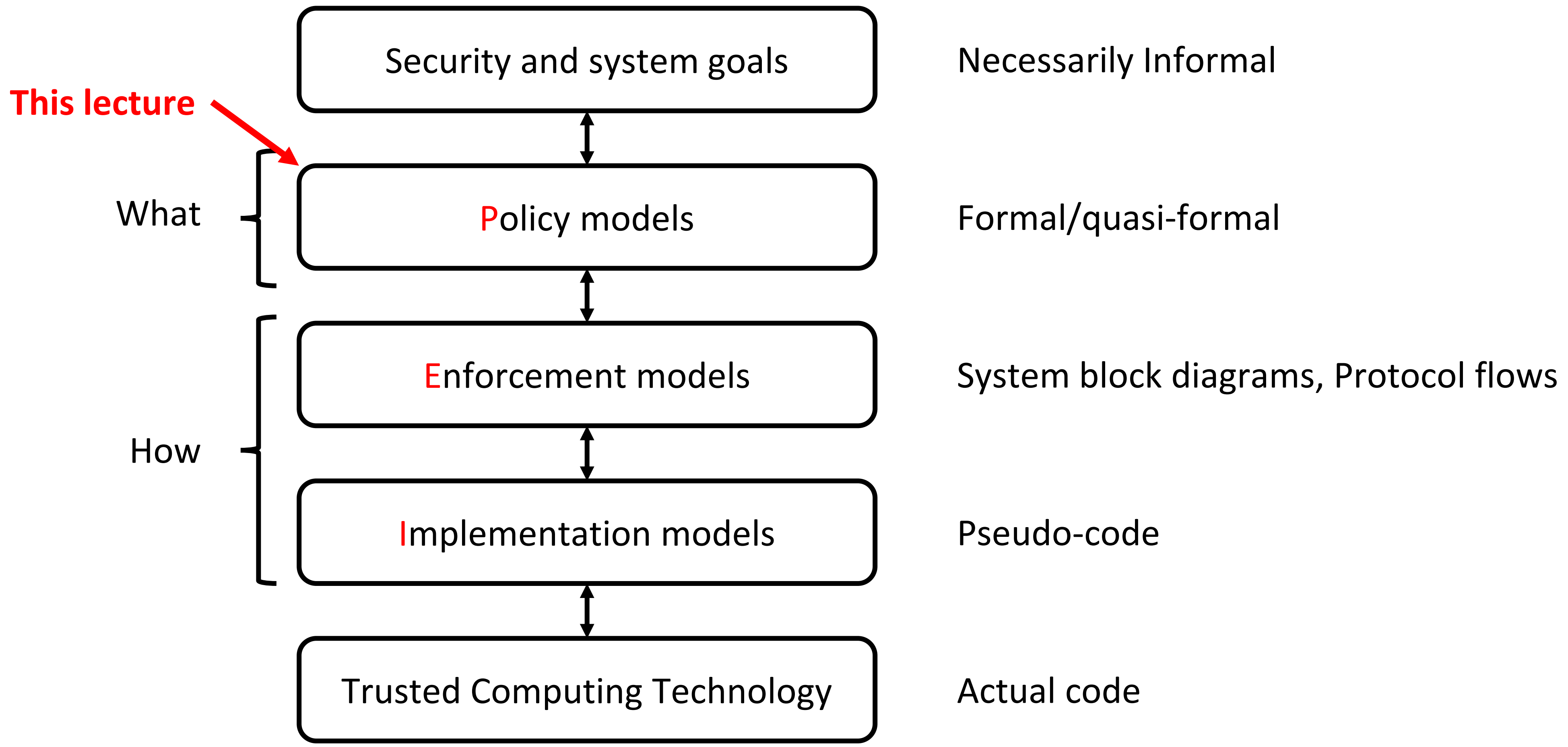
(3) **Authorize**

(4) Audit



https://knowyourmeme.com/memes/you
shall-not-pass

PEI Model [1]



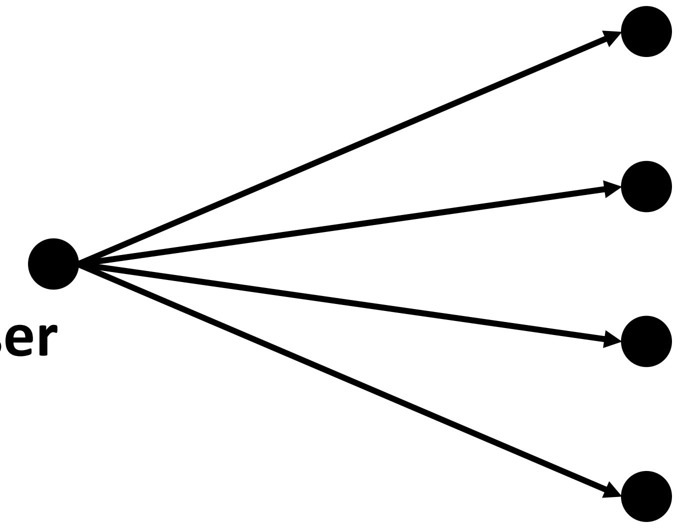
This lecture
Security and system goals
Policy models
Enforcement models
Implementation models
Trusted Computing Technology
What
How
Necessarily Informal
Formal/quasi-formal
System block diagrams, Protocol flows
Pseudo-code
Actual code

Vocabulary

- Basic abstractions:
  - Subjects
  - Objects
  - Rights

- A **subject** is an entity who wishes to access a certain **object**, which is a resource (e.g., a file or a network packet). The different modes of access (e.g., reading, writing) are called **permissions**.

Vocabulary – Users and Principals

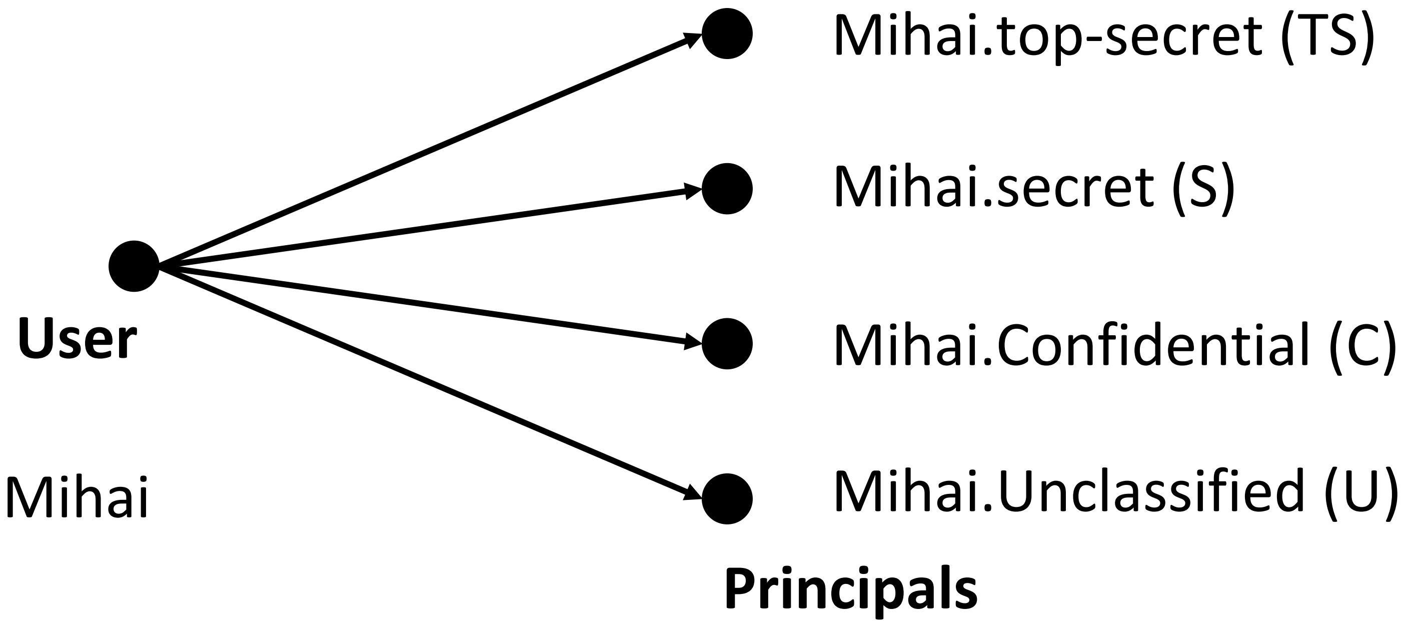


Real World **User**

Principals - Unit of Access Control and Authorization

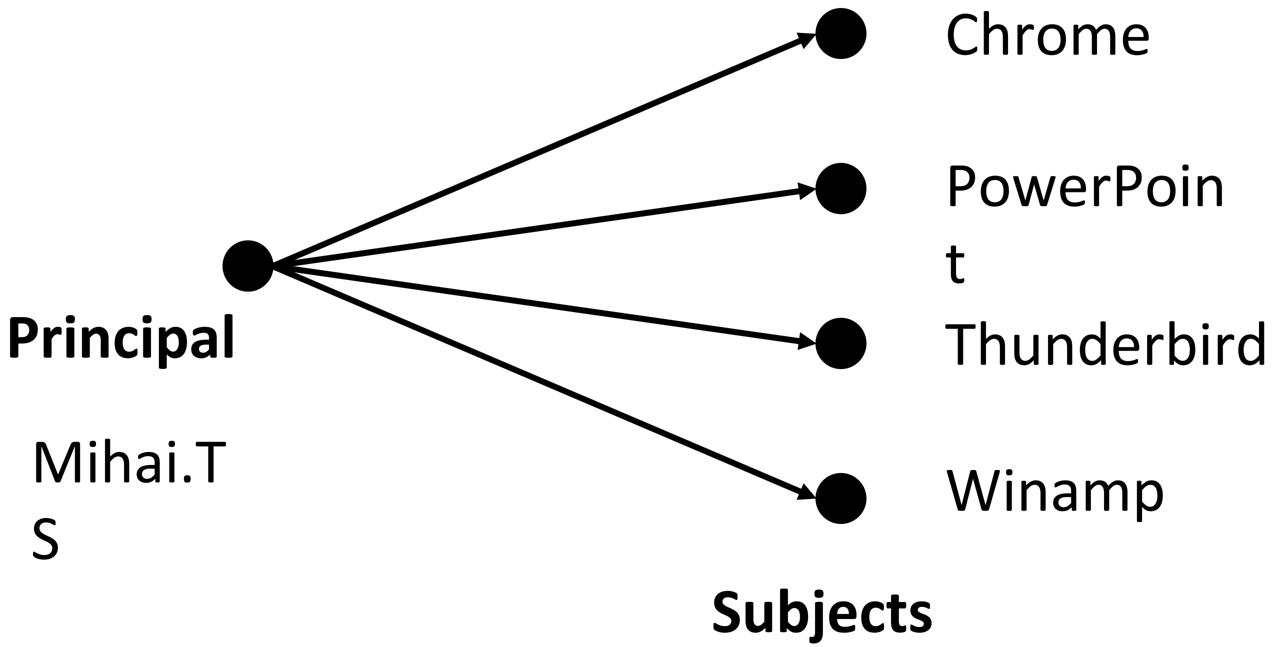
- A Principal is an User authenticated in a context

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 models

Access control enforcement

- **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.

Access control enforcement

- **Attribute-Based Access Control (ABAC)** – properties of an object are used when usage decision are made**.**
- **Usage Control (UCON)** – generalization of access control to include authorization, obligations, conditions, continuity and mutability of attributes.

Discretionary access controls

DAC

- No precise definition.
- 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			parent	
U4		r		

Access Control Lists (ACL)

- An access control list is a set {Ao | o ∈ 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: parent

D
U2: Kill

Capabilities

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

```
int fd = open("/etc/passwd", O_RDWR);
```

=> fork() + exec(), new process inherits fd (the authorization "token")

- Windows: Security Identifier (SID) in Active Directory

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

ACL vs. Capabilities

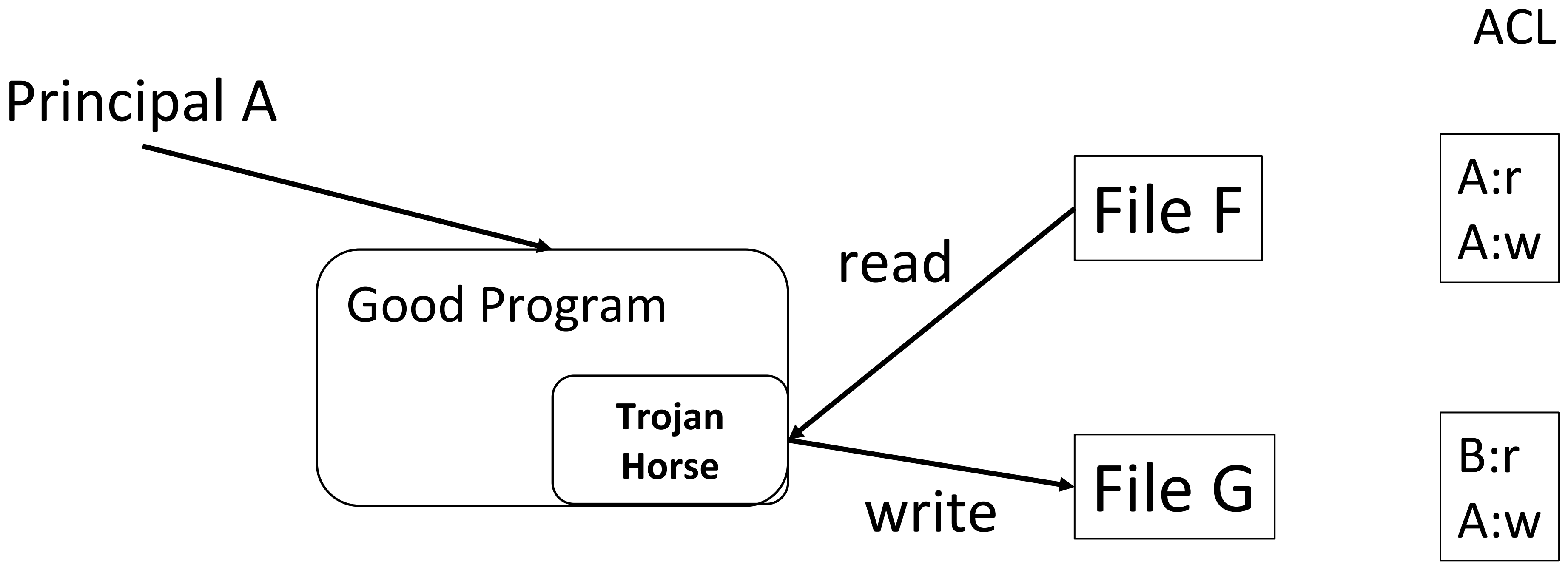
- ACL require authentication of subjects
- Capabilities do not require authentication of subjects, but do require unforgeability and control of propagation of capabilities. Usually implemented through cryptography.

- The Confused Deputy Problem [1986]
  - Example: Cross-Site Scripting (XSS), *setuid* privilege escalation (e.g., sudo)
  - Solution: Use Capabilities Implementation

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

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 controls

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
- TOCTTOU (Time Of Check To Time of Use) problem:
  - authority checks access to an object
  - *unknowingly to him, attacker replaces object with another one*
  - privileged subject operates on attacker controlled object!

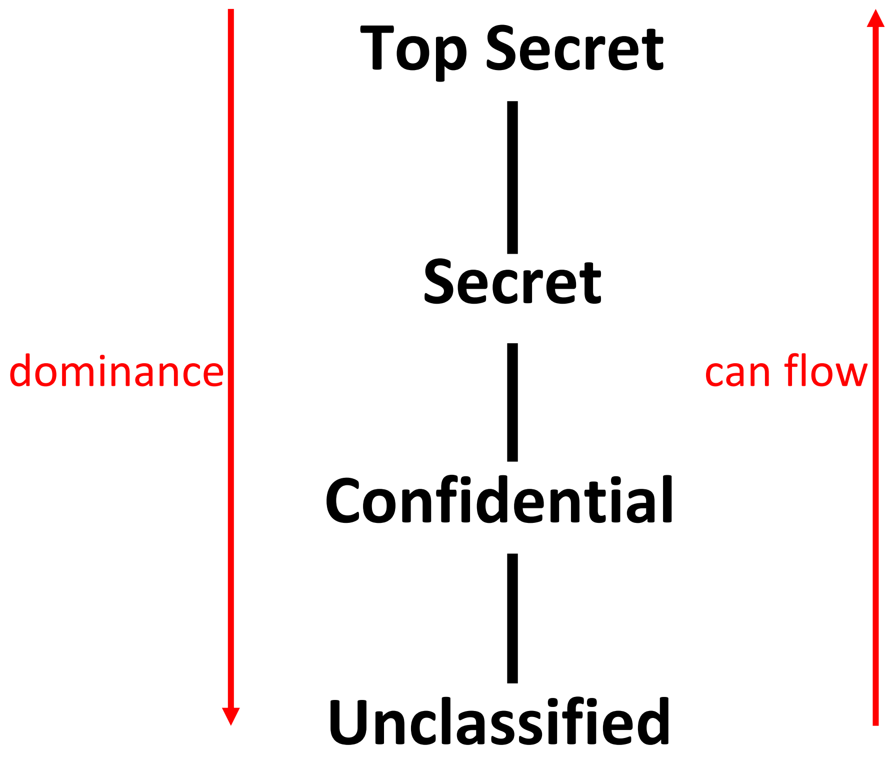
Modeling Access Control

- Multi-level security (MLS)
  - Bell-LaPadula (BLP)
  - Biba Model
- 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

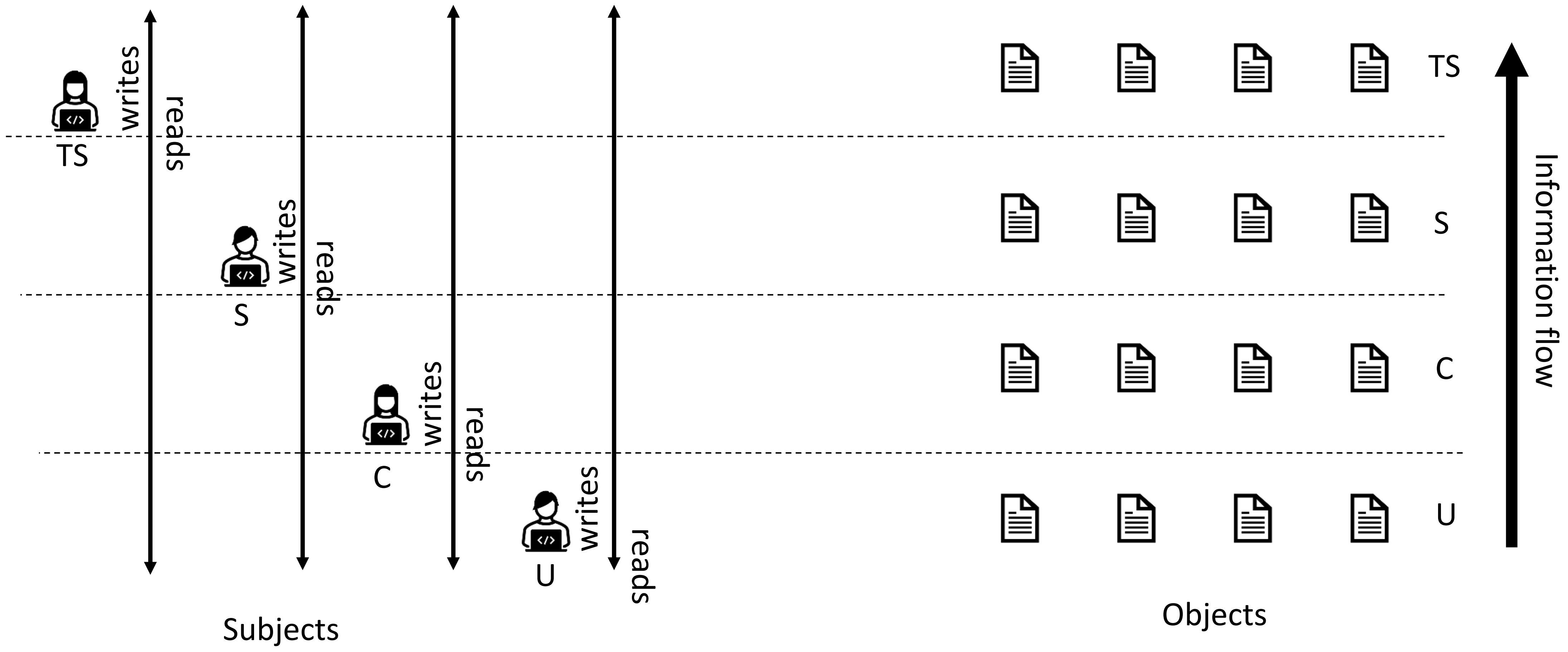
- Aims to capture confidentiality (read) requirements only
- The system is modelled as transitions through a set of states, starting from an initial state.
  - State = Object, access matrix, current access information
- State transition rules describe how a system can go from one state to another
- Each 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:
  - A) Simple Security Property (SS): no subject may read data at a higher level
  - B) The *(Star)-Property (SP): no subject may write data at a lower level
    - (due to the fear of Trojan Horse)

- A system is secure if and only if every reachable state is secure.

BLP Model


writes
reads
TS
writes
reads
S
writes
reads
C
writes
reads
U
Subjects
TS
S
C
U
Objects
Information flow

BLP Problems

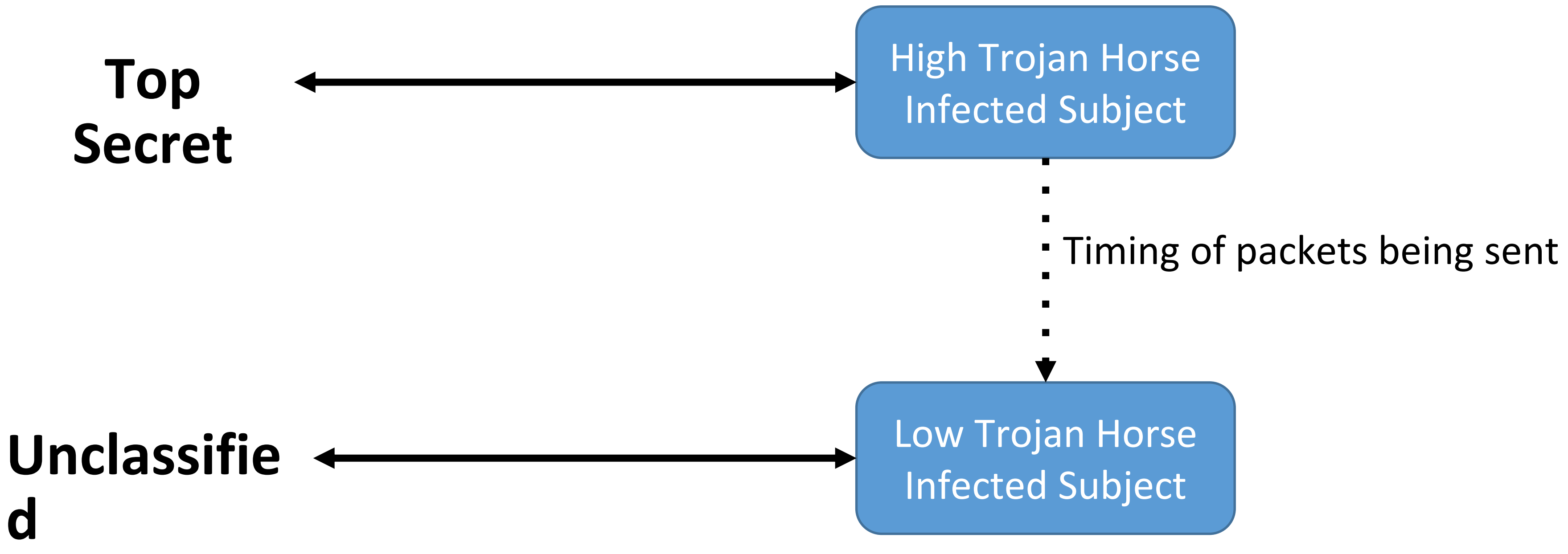
- Consider a system with subjects s1, s2, and objects o1, o2
  - Lm(s1) = Lc(s1) = L(o1) = high
  - Lm(s2) = Lc(s2) = L(o2) = low
- And the following execution
  - s1 gets access to o1, reads something, releases access, then changes current level to low, gets write access to o2, writes to o2
- Every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

BLP Problems

- There is no ACK from High to Low
- 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


Top Secret
High Trojan Horse Infected Subject
Timing of packets being sent
Unclassified
Low Trojan Horse Infected Subject

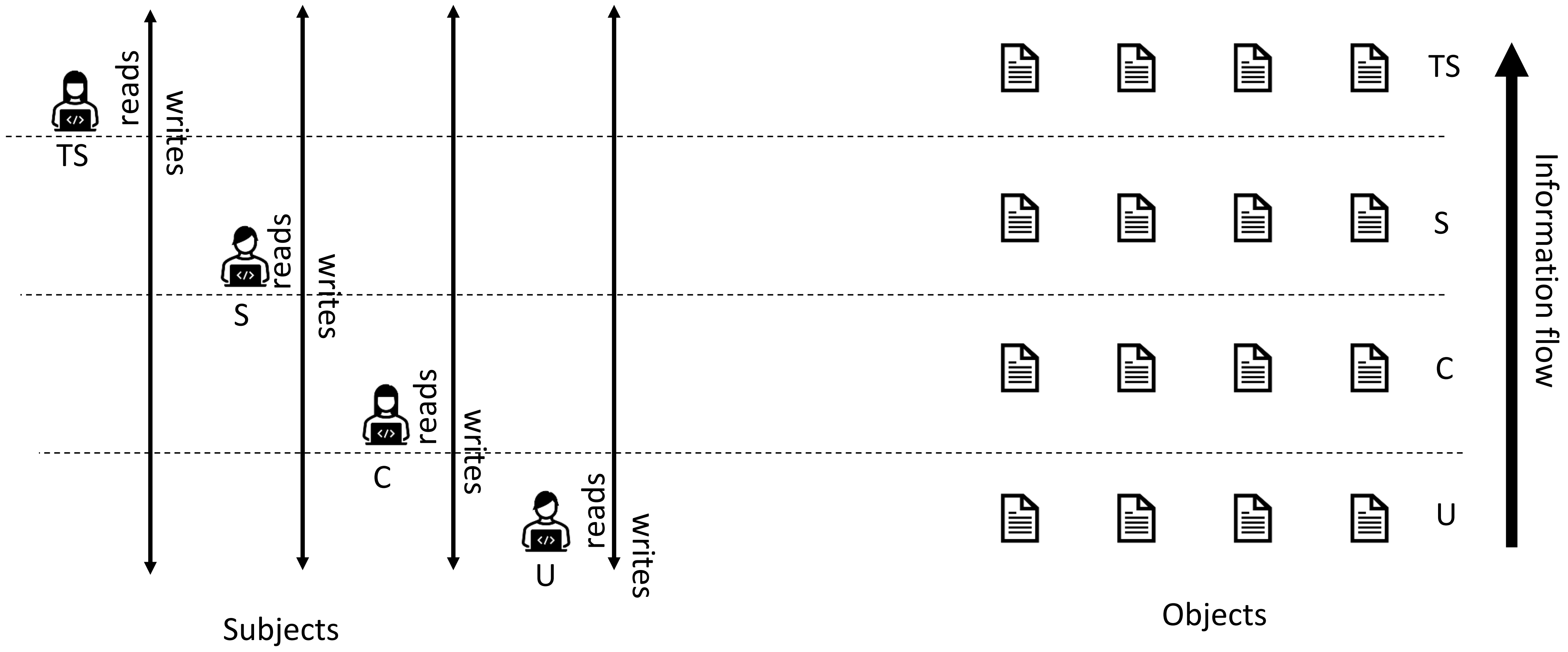
Biba Model

- 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

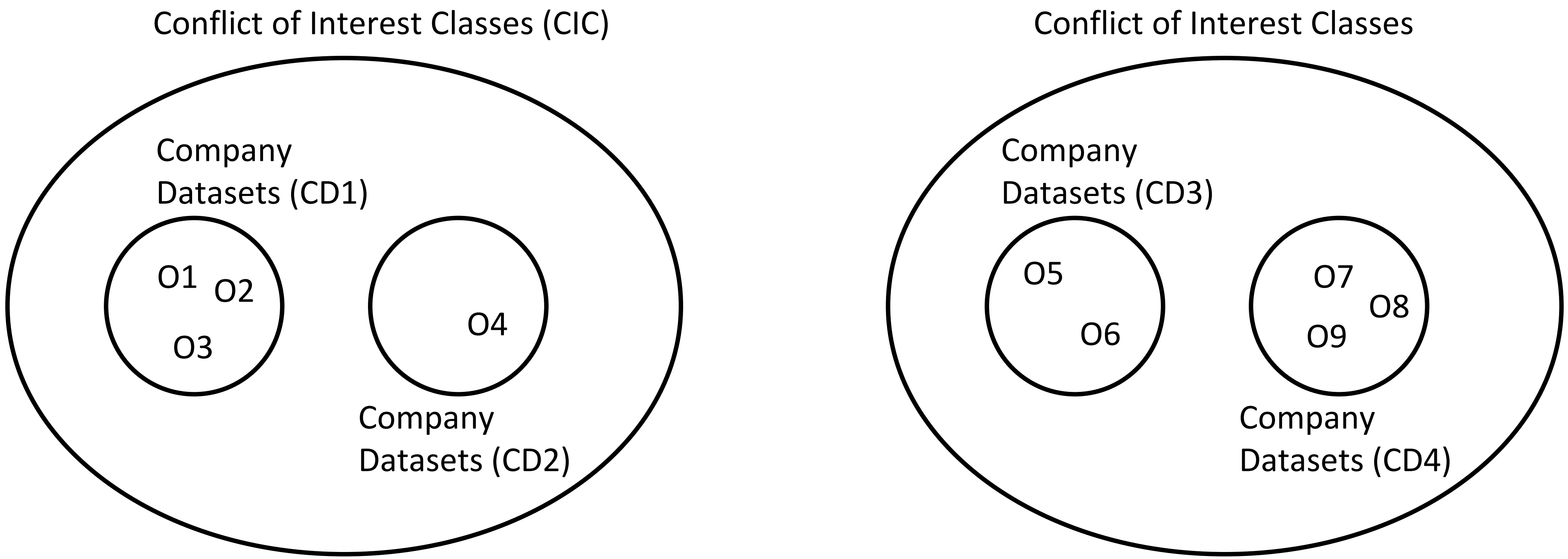

reads
writes
TS
reads
writes
S
reads
writes
C
reads
writes
U
Subjects
TS
S
C
U
Information flow
Objects

Biba Model

- Used by Windows

- E.g., A Internet Explorer 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]



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 by the simple security rule

  and
  - no object can be read which is in a different company dataset to the one for which write access is request

Chinese Wall

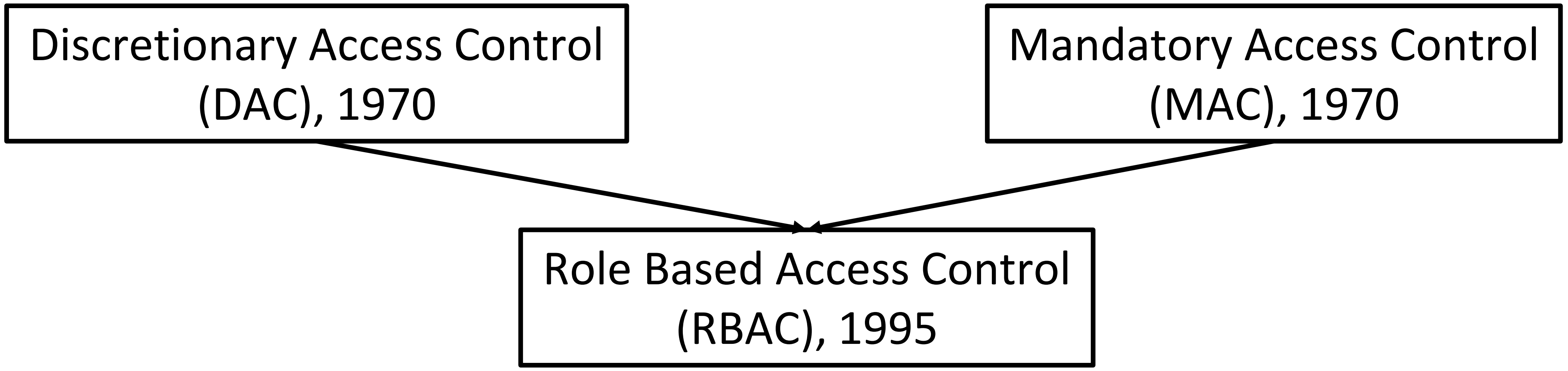
- Once a subject reads two objects from different CDs, that subject may never write any object.

- S1 reads information from an object in CD1.
- S1 writes that information to object O6 in CD3.
- S2 reads that information from O6.
- At the end of this sequence, S2 would have read information pertaining to both CD1 and CD2, which would violate the Chinese Wall policy since both CDs are in the same CIC.

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.)


Discretionary Access Control (DAC), 1970
Mandatory Access Control (MAC), 1970
Role Based Access Control (RBAC), 1995

Role-Based Access Control

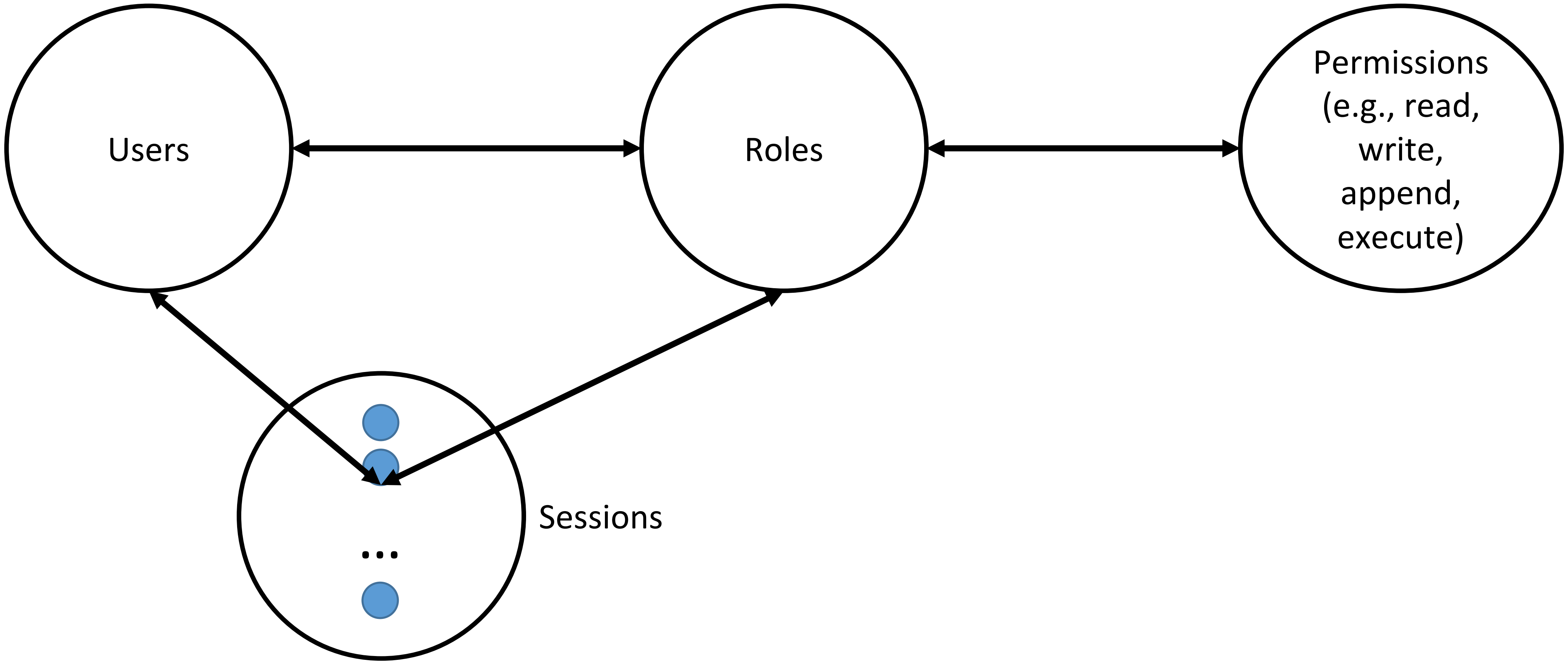
- Can be configured to do MAC
  - roles simulate clearances (ESORICS 96)

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

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

Role-Based Access Control



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

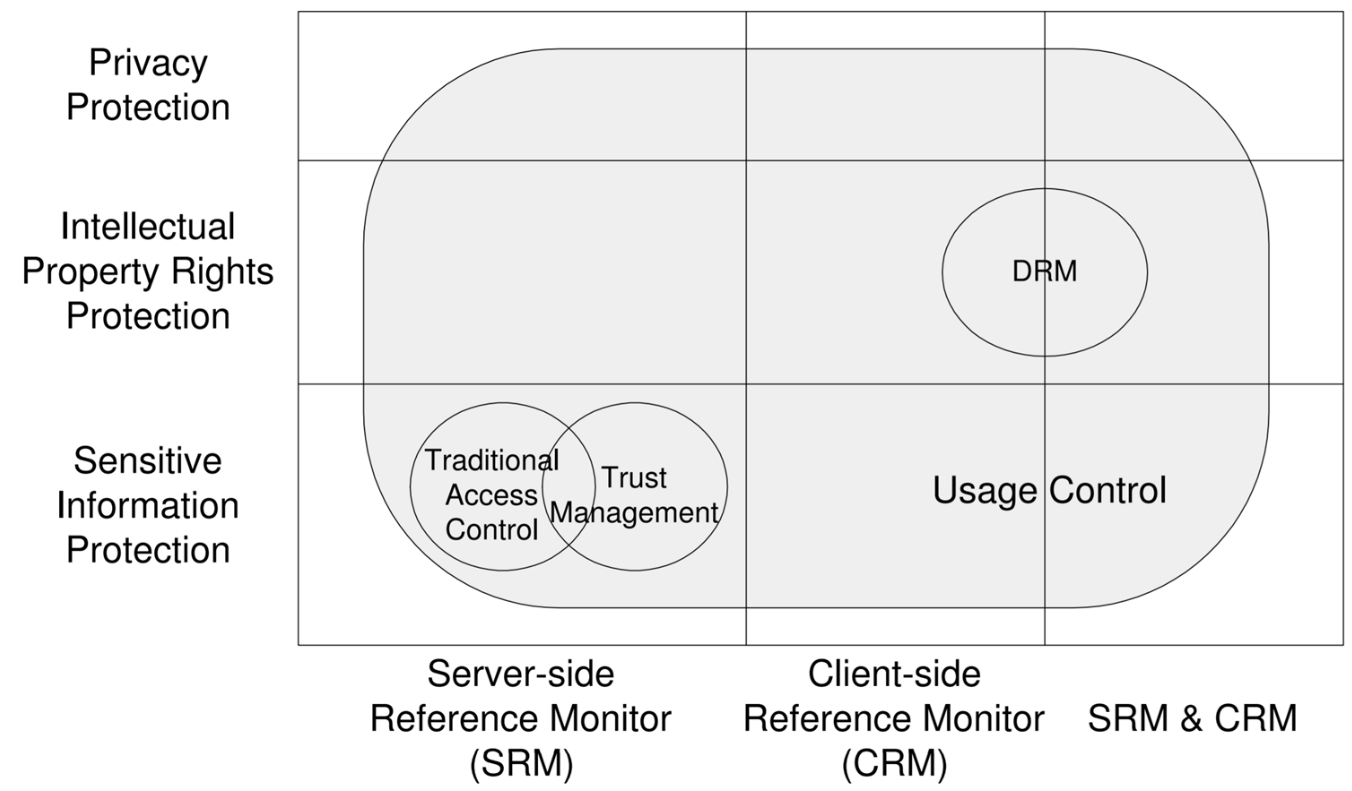
Future Access Control

Attribute-Based Access Control (ABAC)

- Attributes are name:value pairs
  - possibly chained
  - values can be complex data structures
- Associated with
  - users
  - subjects
  - objects
  - contexts
- Converted by policies into rights just in time
  - policies specified by security architects
  - attributes maintained by security administrators
  - ordinary users morph into architects and administrators

Usage Control (UCON)

- Unified framework for access control, trust management and digital rights management


Privacy Protection
Intellectual Property Rights Protection
Sensitive Information Protection
DRM
Traditional Access Control
Trust Management
Usage Control
Server-side Reference Monitor (SRM)
Client-side Reference Monitor (CRM)
SRM & CRM

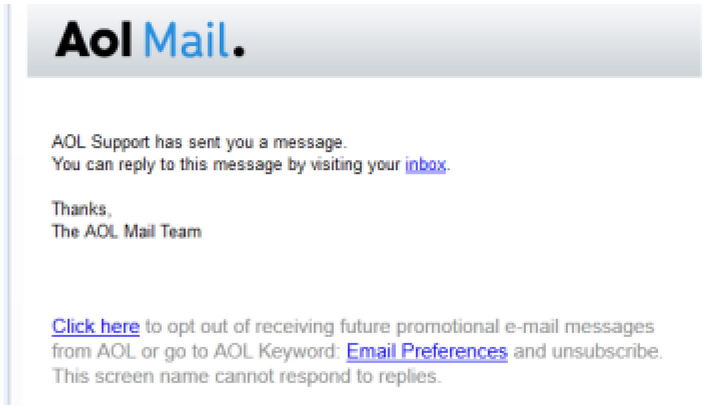
Social engineering

Example - Usable Security

- Description “Usability is one of the most important and yet hardest design problems in many secure systems.” *by Ross Anderson*

- Technology writer David Pogue calculated we spend 17 man-years every day on CAPTCHAs (Scientific American, March 2012)

- Phishing
  - 1996


Aol Mail.

AOL Support has sent you a message.
You can reply to this message by visiting your inbox.

Thanks,
The AOL Mail Team

Click here to opt out of receiving future promotional e-mail messages from AOL or go to AOL Keyword: Email Preferences and unsubscribe.
This screen name cannot respond to replies.

- “Given a choice between dancing pigs and security, users will pick dancing pigs every time.” *by Edward Felten and Gary McGraw*

Dancing pigs!

Security and humans

- Security policies must be in place

  …and must be followed.

- Regardless of how strong (and expensive) your secure deployment is:
  - Humans can still write their passwords on post-it notes
  - Humans can still give their passwords to anyone they trust
  - Humans can still open tempting attachments…

Social engineering

- Non-technical intrusion
- Involves tricking people to break security policies
    - Manipulation
- Relies on false confidence
    - Everyone trusts someone
    - Authority is usually trusted by default
    - Non-technical people don’t want to admit their lack of expertise
        - They ask fewer questions.
    - Most people are eager to help.
        - When the attacker poses as a fellow employee in need.

Social engineering

- People are not aware of the value of the information they possess.
- Vanity, authority, eavesdropping – they all work.
- When successful, social engineering bypasses ANY kind of security.

SOCIAL ENGINEERING
The clever manipulation of the natural human tendency to trust.

Types of phishing

- By used technology
  - Smishing (SMS)
  - Vishing (Voice)
  - Email phishing
  - Angler phishing (via social networks)
- By target
  - Watering Hole Phishing (people visiting a certain website)
  - Spear phishing (a specific organization)
  - Whaling (C-level from a specific organization)

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