Cristina Nita-Rotaru

CS526: Information security

Access Control Models

1: Discretionary Access Control

Readings for this lecture

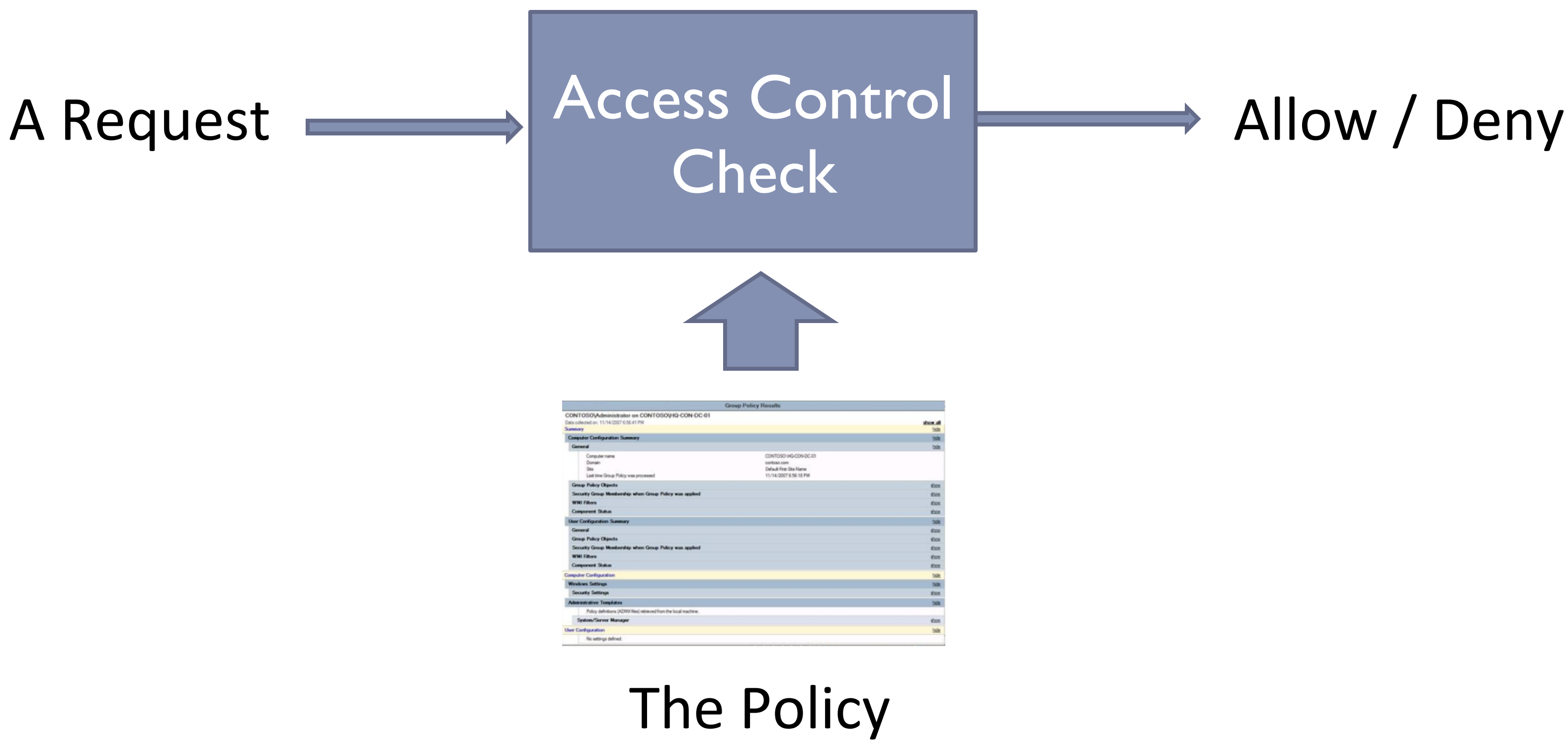
- Wikipedia
  - Discretionary Access Control
  - Confused Deputy Problem
  - Capability-based Security
  - Ambient Authority
  - Mandatory Access Control

Why computers are vulnerable?

- Programs are buggy
- Humans make mistakes
- Access control is not good enough
  - Discretionary Access Control (DAC) used in Unix and Windows assume that programs are not buggy

Access control check

- Given an access request, return an access control decision based on the policy
  - allow / deny



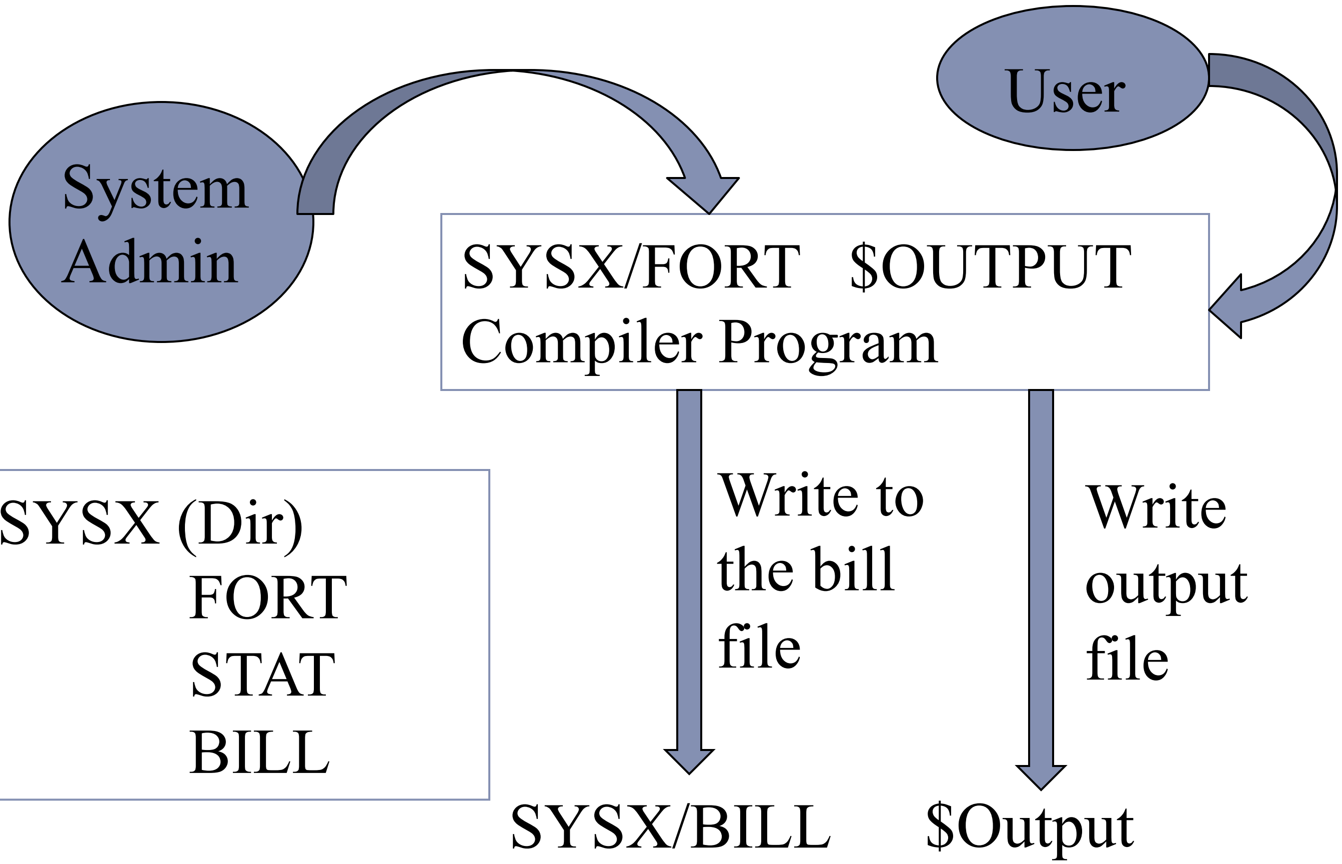
Discretionary access control

- 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.
- According to TCSEC (Trusted Computer System Evaluation Criteria)
  - "A means of restricting access to objects based on the identity and need-to-know of users and/or groups to which they belong. Controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (directly or indirectly) to any other subject."

DAC Limitations

- DAC causes the Confused Deputy problem
  - Solution: use capability-based systems
- DAC does not preserve confidentiality when facing Trojan horses
  - Solution: use Mandatory Access Control (BLP)
- DAC implementation fails to keep track of for which principals, a subject (process) is acting on behalf of
  - Solution: fixing the DAC implementation to better keep track of principals

The confused deputy problem



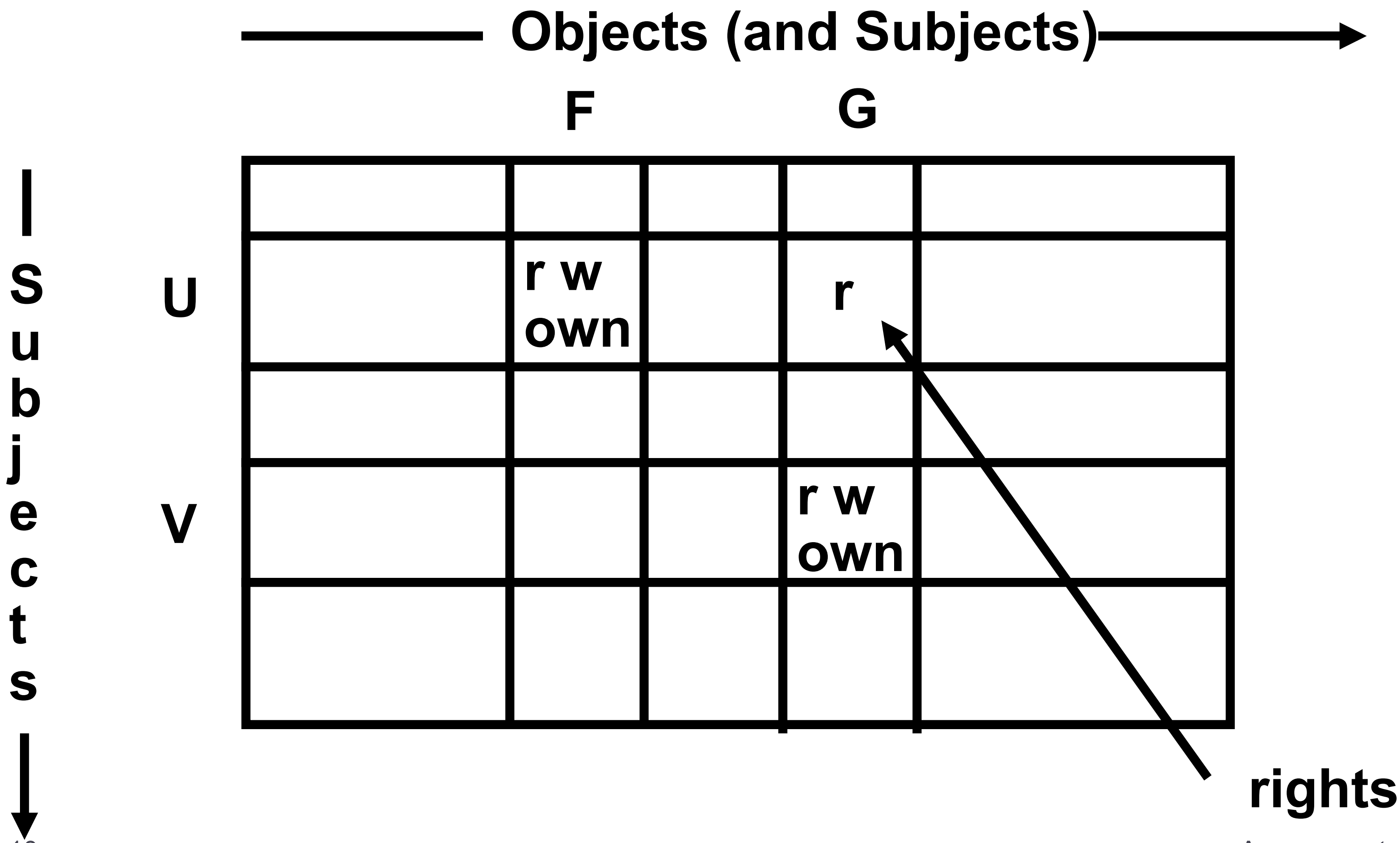
SYSX (Dir)
FORT
STAT
BILL

The Confused Deputy by *Norm Hardy*

The confused deputy problem (cont.)

- The compiler runs with authority from two sources
  - the invoker (i.e., the programmer)
  - the system admin (who installed the compiler and controls billing and other info)
- It is the deputy of two masters
- There is no way to tell which master the deputy is serving when performing a write
- Solution: Use capability

Access matrix model



Implementation of access matrix

- Access Control Lists
  - Encode columns
- Capabilities
  - Encode rows
- Access control triples
  - Encode cells

Access control lists (ACLs)

- each column of the access matrix is stored with the object corresponding to that column

F

U:r
U:w
U:own

G

U:r
V:r
V:w
V:own

Capabilities lists

U	**F/r, F/w, F/own, G/r**
V	**G/r, G/w, G/own**

each row of the access matrix is stored with the subject corresponding to that row

Access control triples

Subject	Access	Object
U	r	F
U	w	F
U	own	F
U	r	G
V	r	G
V	w	G
V	own	G

commonly used in relational DBMS

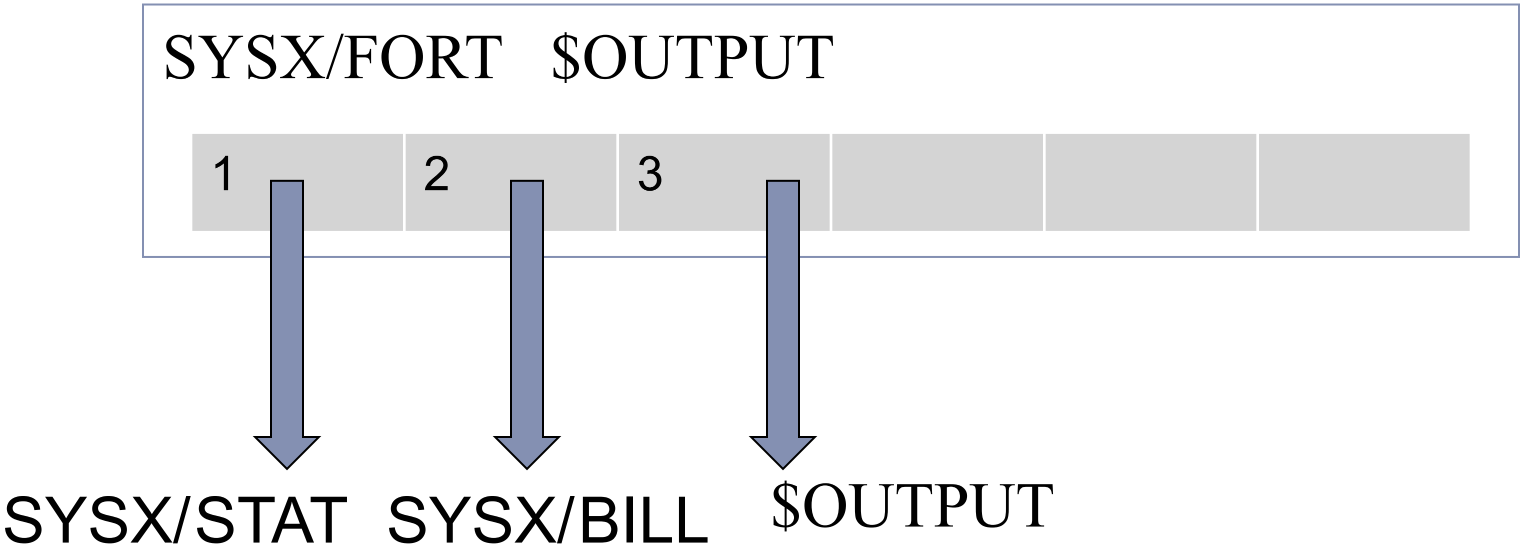
Different notions of capabilities

- Capabilities as a row representation of Access Matrices
- Capabilities used in Linux as a way to divide the root power into multiple pieces that can be given out separately
- Capabilities as a way of implementing the whole access control systems
  - Subjects have capabilities, which can be passed around
  - When accessing resources, subjects select capabilities to access
    - An example is open file descriptors

More on capability based access control

- Subjects have capabilities, which
  - Give them accesses to resources
    - E.g., like keys
  - Are transferable and unforgeable tokens of authority
    - Can be passed from one process to another
      - Similar to opened file descriptors
- Why capabilities may solve the confused deputy problems?
  - When accessing a resource, must select a capability, which also selects a master

Back to the confused deputy problem



- Invoker must pass in a capability for $OUTPUT, which is stored in slot 3.
- Writing to output uses the capability in slot 3.
- Invoker cannot pass a capability it doesn't have.

Capability vs. ACL

- Consider two security mechanisms for bank accounts
- One is identity-based. Each account has multiple authorized owners. You go into the bank and show your ID, then you can access all accounts you are authorized
  - Once you show ID, you can access all accounts
  - You have to tell the bank which account to take money from
- The other is token-based. When opening an account, you get a passport to that account and a PIN, whoever has the passport and the PIN can access

Capabilities vs. ACL: Ambient authority

- Ambient authority means that a user’s authority is automatically exercised, without the need of being selected
  - causes the confused deputy problem
- Example: You are carrying a lot of keys. When you walk to a door, the door automatically opens if you have the right key. You don’t need to select a key.

- No ambient authority in capability systems

Capability vs. ACL: Naming

- ACL systems need a namespace for objects
- In capability systems, a capability can serve both to designate a resource and to provide authority
- ACLs also need a namespace for subjects or principals
  - as they need to refer to subjects or principals
- Implications
  - the set of subjects cannot be too many or too dynamic
  - most ACL systems grant rights to user accounts principals, and do not support fine-grained subject rights management

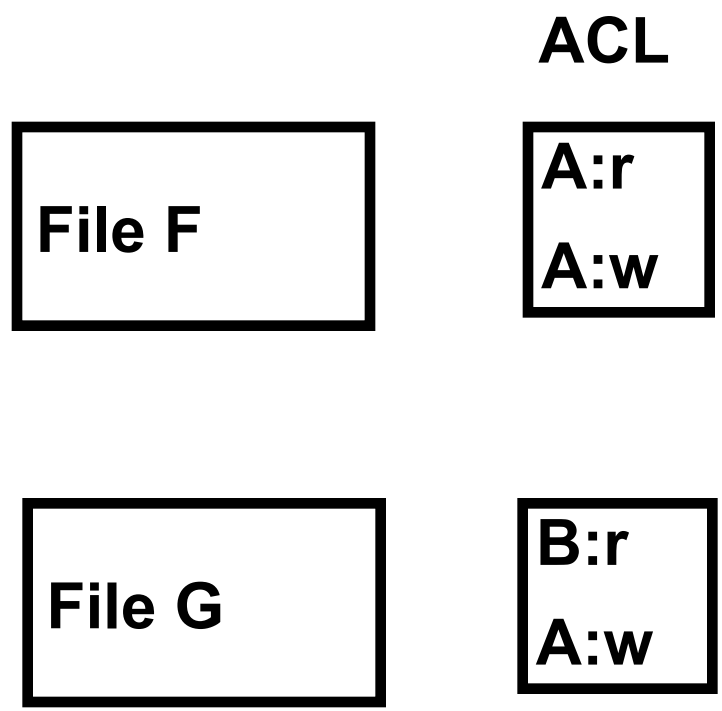
Conjectures on why most real-world OS use ACL, rather than capabilities

- Capability is more suitable for process level sharing, but not user-level sharing
  - user-level sharing is what is really needed
- Processes are more tightly coupled in capability-based systems because they need to pass capabilities around
  - programming may be more difficult

Inherent weakness of DAC

- Unrestricted DAC allows information flows from an object which can be read to any other object which can be written by a subject
  - Suppose A is allowed to read some information and B is not, A can read and tell B
- Suppose users are trusted not to do this deliberately. It is still possible for Trojan Horses to copy information from one object to another

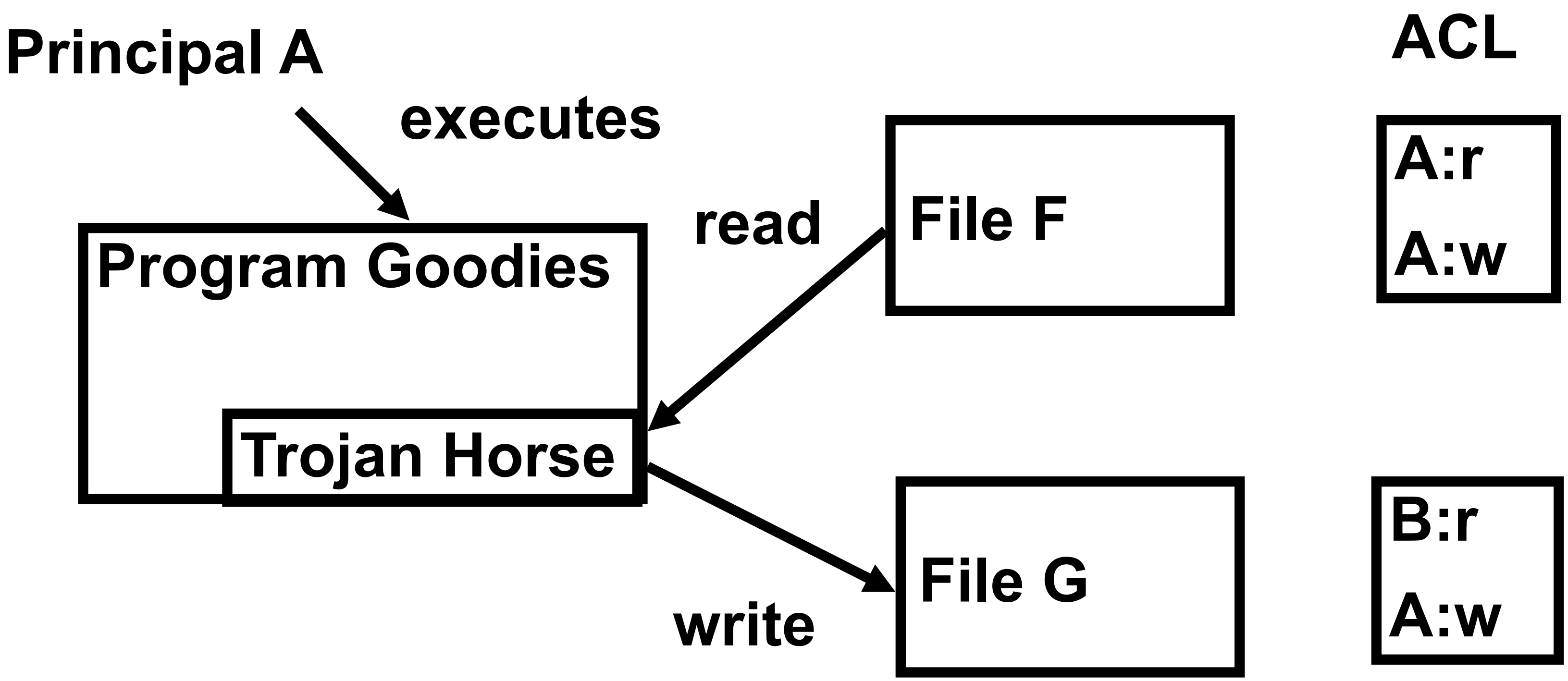
Trojan Horse example



Principal B cannot read file F

Trojan Horse example

- Principal B can read contents of file F copied to file G



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 is developed to address this problem

DAC's weaknesses caused by the gap

- A request: a subject wants to perform an action
  - E.g., processes in OS
- The policy: each principal has a set of privileges
  - E.g., user accounts in OS

- Challenging to fill the gap between the subjects and the principals
  - relate the subject to the principals

Unix DAC revisited (1)

Action	Process	Effective UID	Real Principals
User A Logs In	shell	User A	User A
Load Binary “Goodie” Controlled by user B	Goodie	User A	? ?

- When the Goodie process issues a request, what principal(s) is/are responsible for the request?
- Under what assumption, it is correct to say that User A is responsible for the request?

Assumption: Programs are benign, i.e., they only do what they are told to do.

UNIX DAC revisited (2)

Action	Process	Effective UID	Real Principals
	shell	User A	User A
Load AcroBat Reader Binary	AcroBat	User A	User A
Read File Downloaded from Network	AcroBat	User A	? ?

- When the AcroBat process (after reading the file) issues a request, which principal(s) is/are responsible for the request?
- Under what assumption, it is correct to say that User A is responsible for the request?

Assumption: Programs are correct, i.e., they handle inputs correctly.

Why DAC is vulnerable?

- Implicit assumptions
  - Software are benign, i.e., behave as intended
  - Software are correct, i.e., bug-free
- The reality
  - Malware are popular
  - Software are vulnerable
- The problem is not caused by the discretionary nature of policy specification!
  - i.e., owners can set policies for files

Why DAC is vulnerable? (cont')

- A deeper reason in the enforcement mechanism
  - A single invoker is not enough to capture the origins of a process
- When the program is a Trojan
  - The program-provider should be responsible for the requests
- When the program is vulnerable
  - It may be exploited by input-providers
  - The requests may be issued by injected code from input-providers
- Solution: include input-providers as the principals

2: Bell LaPadula Model

Readings for this lecture

- Wikipedia
  - Bell-LaPadula model

- David E. Bell: Looking Back at the Bell-La Padula Model

Access control at different abstractions

- Using principals
  - Determines which principals (user accounts) can access what documents
- Using subjects
  - Determines which subjects (processes) can access what resources
  - This is where BLP focuses on

Multi-level security (MLS)

- The capability of a computer system to carry information with different sensitivities (i.e. classified information at different security levels)
  - permit simultaneous access by users with different security clearances and needs-to-know
  - prevent users from obtaining access to information for which they lack authorization.
  - **Discretionary access control fails to achieve MLS**
- Example of security levels
  - Top Secret > Secret > Confidential > Unclassified
- Security goal is confidentiality: ensures that information does not flow to those not cleared for that level

Mandatory access control

- Mandatory access controls (MAC) restrict the access of subjects to objects based on a system-wide policy
  - denying users full control over the access to resources that they create. The system security policy (as set by the administrator) entirely determines the access rights granted

Bell-LaPadula: A MAC model for achieving multi-level security

- Introduced in 1973
- Air Force was concerned with security in time-sharing systems
  - Many OS bugs
  - Accidental misuse
- Main Objective:
  - Enable one to formally show that a computer system can securely process classified information

What is a Security Model?

- A model describes the system
  - e.g., a high level specification or an abstract machine description of what the system does
- A security policy
  - defines the security requirements for a given system
- Verification techniques that can be used to show that a policy is satisfied by a system
- System Model + Security Policy = Security Model

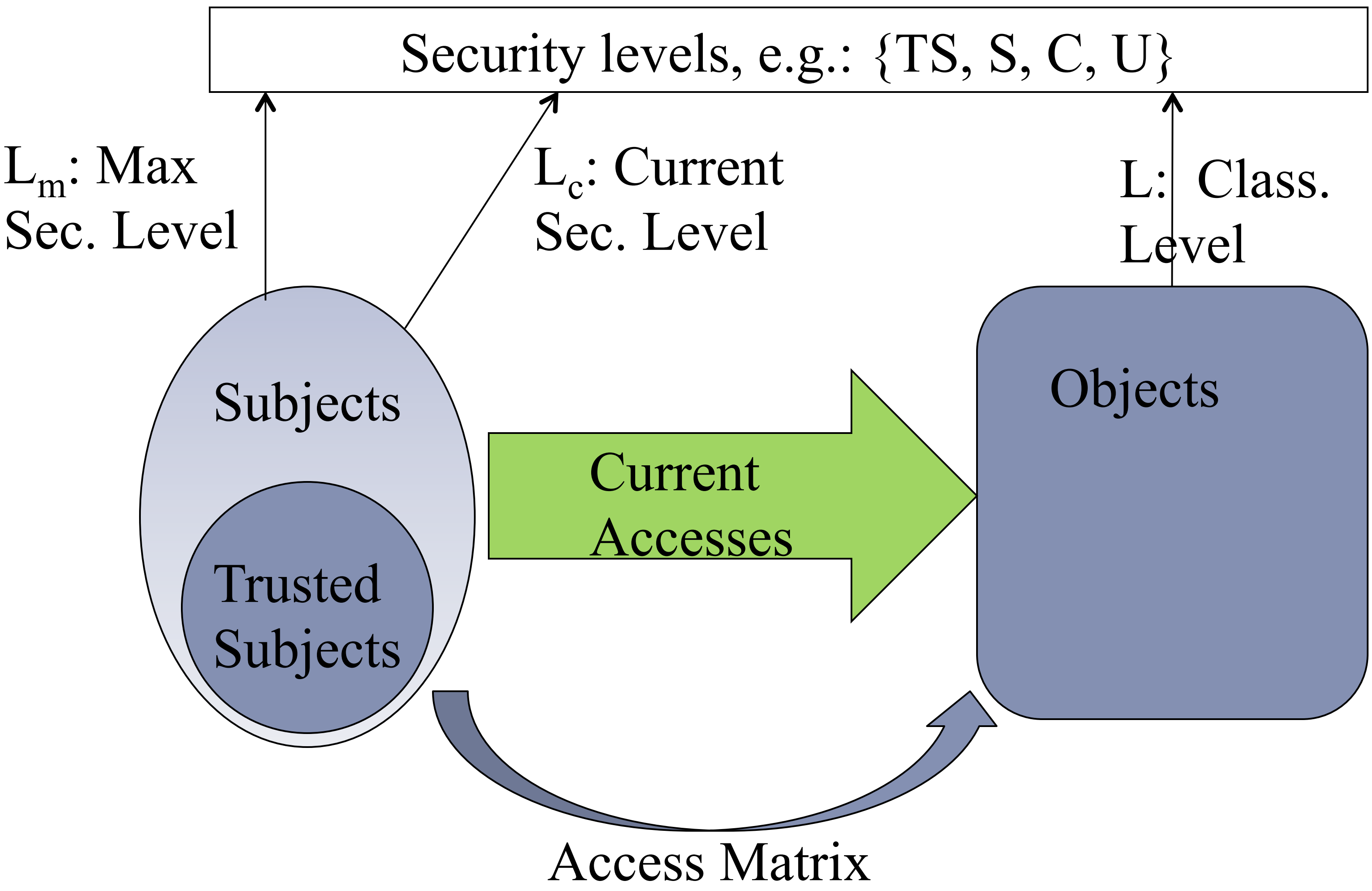
Approach of BLP

- Use state-transition systems to describe computer systems
- Define a system as secure iff. every reachable state satisfies 3 properties
  - simple-security property
  - *-property
  - discretionary-security property
- Prove a Basic Security Theorem (BST)
  - so that given the description of a system, one can prove that the system is secure

BLP: System Model

- A computer system is modeled as a state-transition system
- There is a set of subjects; some are designated as trusted.
- Each state has objects, an access matrix, and the current access information
- There are state transition rules describing 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

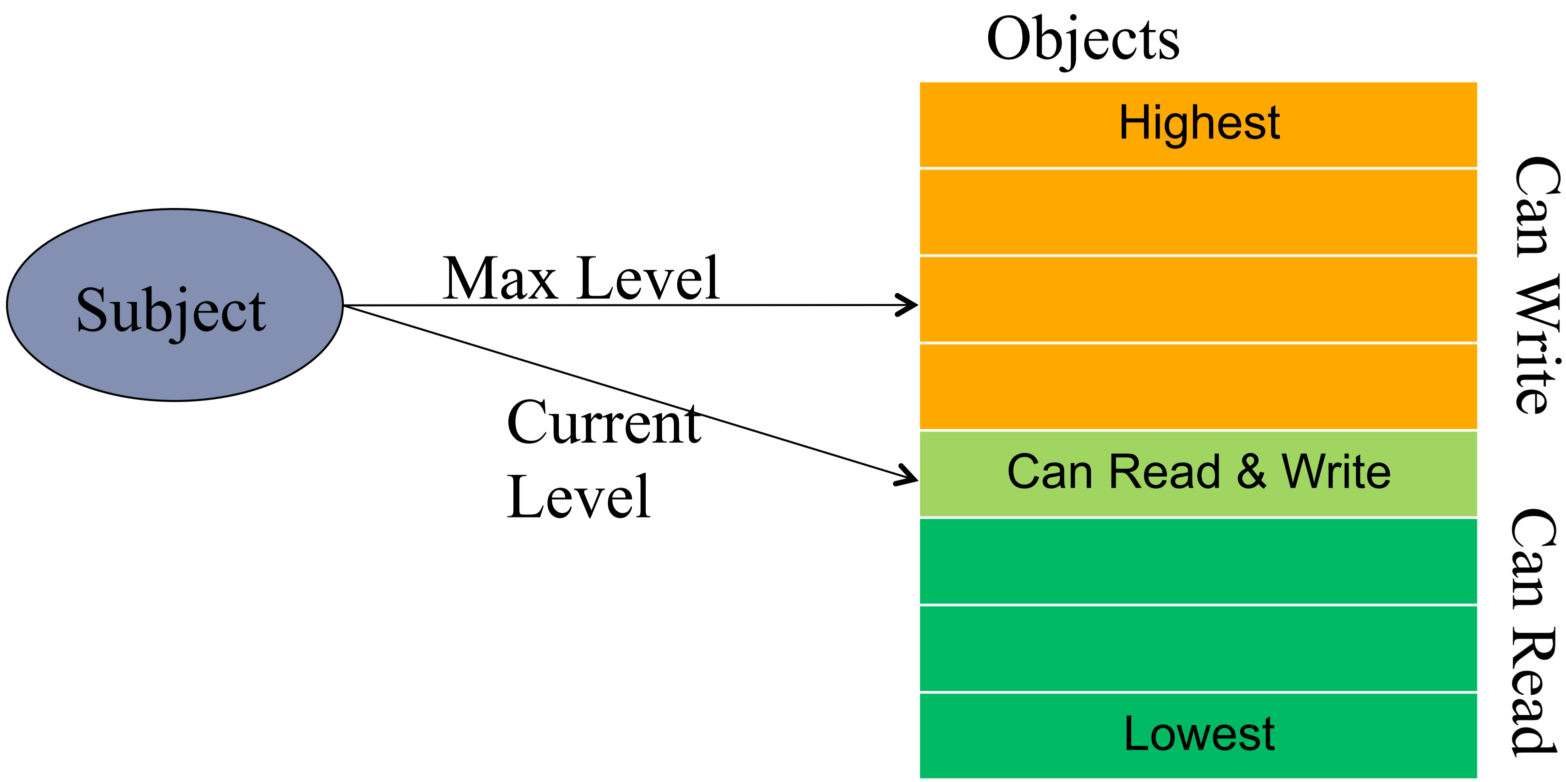
Elements of the BLP model



BLP: Security policy

- A state is secure if it satisfies
  - Simple Security Condition (**<u>no read up</u>**):
    - S can read O iff Lm(S) ≥ L(O)
  - The Star Property (**<u>no write down</u>**): for any S that is not trusted
    - S can read O iff Lc(S) ≥ L(O) (no read up)
    - S can write O iff Lc(S) ≤ L(O) (no write down)
  - Discretionary-security property
    - every access is allowed by the access matrix
- A system is secure if and only if every reachable state is secure.
- Note: Trusted subjects are not restricted to the Star Property

Implication of the BLP policy



Star property

- Applies to subjects not to principals and users
- Users are trusted (must be trusted) not to disclose secret information outside of the computer system
- Subjects are not trusted because they may have Trojan Horses embedded in the code they execute
- Star-property prevents **overt leakage of information** but does not address the covert channel problem

Overt (explicit) channels vs. covert channels

- Security objective of MLS in general, BLP in particular
  - high-classified information cannot flow to low-cleared users
- Overt channels of information flow
  - read/write an object
- Covert channels of information flow
  - communication channel based on the use of system resources not normally intended for communication between the subjects (processes) in the system

Examples of covert channels

- Using file lock as a shared boolean variable
- By varying its ratio of computing to input/output or its paging rate, the service can transmit information to a concurrently running process
- Timing of packets being sent

- Covert channels are often noisy
- However, information theory and coding theory can be used to encode and decode information through noisy channels

BLP and covert channels

- Covert channels cannot be blocked by star-property
- It is generally very difficult, if not impossible, to block all covert channels
- One can try to limit the bandwidth of covert channels
- Military requires cryptographic components be implemented in hardware
  - to avoid Trojan horse leaking keys through covert channels

Limitations of BLP notion of security

- The objective of BLP security is to ensure
  - a subject cleared at a low level should never read information classified high
- The simple-security-property and the star-property are sufficient to stop such information flow at any given state
- What about information flow across states?

BLP security is not sufficient!

- 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**

More on the BLP Notion of Security

- When a subject A copies information from high to a low object f, this violates the star-property, but no information leakage occurred yet
  - Only when B, who is not cleared at high, reads f, does leakage occurs
  - If the access matrix limits access to f only to A, then such leakage may never occur
- BLP notion of security is neither sufficient nor necessary to stop illegal information flow (through direct/overt channels)
- The state based approach is too low level and limited in expressive power

How to Fix The BLP Notion of Security?

- May need to differentiate externally visible objects from other objects
  - e.g., a printer is different from a memory object

- State-sequence based property
  - e.g., exists no sequence of states so that there is an information path from a high object to a low externally visible object or to a low subject

The Basic Security Theorem

- This provides the verification techniques piece in
  - Model – Policy – Verification framework

- Restatement of The Basic Security Theorem: A system is a secure system if and only if the starting state is a secure state and each action (concrete state transition that could occur in an execution sequence) of the system leads the system into a secure state.

Observations of the BST

- The BST is purely a result of defining security as a state-based property.
  - It holds for any other state-based property
- The BST cannot be used to justify that the BLP notion of security is "good"
  - This is McLean's main point in his papers
    - "A Comment on the Basic Security Theorem of Bell and LaPadula" [1985]
    - "Reasoning About Security Models" [1987]
    - "The Specification and Modeling of Computer Security" [1990]

Main contributions of BLP

- The overall methodology to show that a system is secure
  - adopted in many later works
- The state-transition model
  - which includes an access matrix, subject security levels, object levels, etc.
- The introduction of star-property
  - Simple-security-property is not enough to stop illegal information flow

Other limitations of BLP

- Addresses only confidentiality, not integrity
- Confidentiality is often not as important as integrity in most situations
- Integrity addressed by other models (such as Biba, Clark-Wilson)

- Does not deal with information flow through covert channels

More on MLS: Security levels

- Used as attributes of both subjects & objects
  - clearance & classification
- Typical military security levels:
  - top secret ≥ secret ≥ confidential ≥ unclassified
- Typical commercial security levels
  - restricted ≥ proprietary ≥ sensitive ≥ public

Security categories

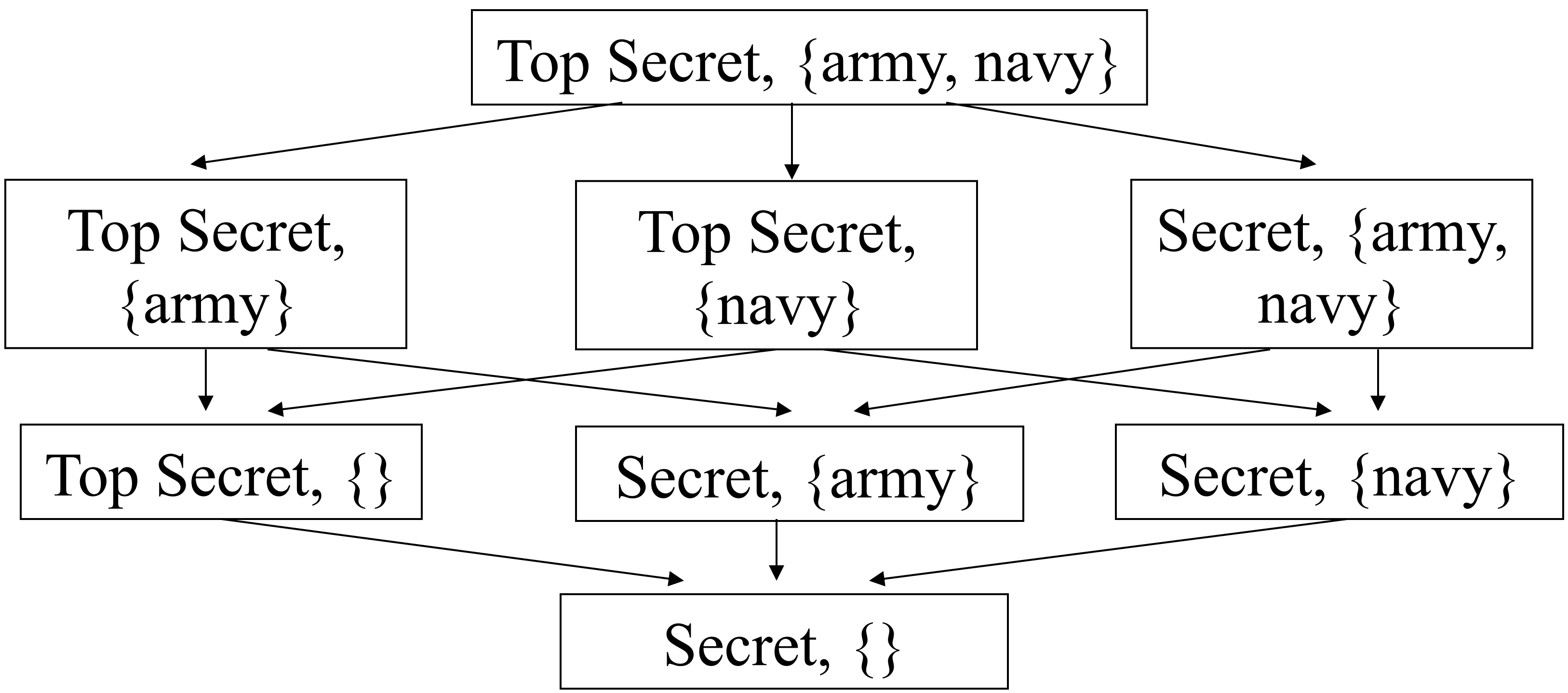
- Also known as compartments
- Typical military security categories
  - army, navy, air force
  - nato, nasa, noforn
- Typical commercial security categories
  - Sales, R&D, HR
  - Dept A, Dept B, Dept C

Security labels

- Labels = Levels × P (Categories)
- Define an ordering relationship among Labels
  - $(e1, C1) \leq (e2, C2)$ iff. $e1 \leq e2$ and $C1 \subseteq C2$
- This ordering relation is a partial order
  - reflexive, transitive, anti-symmetric
  - e.g., $\subseteq$
- All security labels form a lattice

An Example Security Lattice

- levels={top secret, secret}
- categories={army,navy}



The need-to-know principle

- Even if someone has all the necessary official approvals (such as a security clearance) to access certain information they should not be given access to such information unless they have a need to know: that is, unless access to the specific information necessary for the conduct of one's official duties.
- Can be implemented using categories and or DAC

3: Integrity Protection Models: Biba, Clark-Wilson, Chinese Wall

Readings for this lecture

- Related Papers (Optional):
  - Kenneth J. Biba: "Integrity Considerations for Secure Computer Systems", MTR-3153, The Mitre Corporation, April 1977.
  - David D. Clark and David R. Wilson. “A Comparison of Commercial and Military Computer Security Policies.” In IEEE SSP 1987.
  - David FC. Brewer and Michael J. Nash. “The Chinese Wall Security Policy.” in IEEE SSP 1989.

Motivations

- BLP focuses on confidentiality
- In most systems, integrity is equally, if not more, important
- Data integrity vs. system integrity
  - Data integrity means that data cannot be changed without being detected

What is integrity in systems?

- Attempt 1: Critical data do not change.
- Attempt 2: Critical data changed only in "correct ways"
  - E.g., in DB, integrity constraints are used for consistency
- Attempt 3: Critical data changed only through certain "trusted programs"
- Attempt 4: Critical data changed only as intended by authorized users.

Biba: Integrity levels

- Each subject (process) has an integrity level
- Each object has an integrity level
- Integrity levels are totally ordered

- Integrity levels different from security levels in confidentiality protection
  - Highly sensitive data may have low integrity
  - What is an example of a piece of data that needs high integrity, but no confidentiality?

Five mandatory policies in Biba

- Strict integrity policy
- Subject low-water mark policy
- Object low-water mark policy
- Low-water mark integrity audit policy
- Ring policy

- In practice, one may be using one or more of these policies, possibly applying different policies to different subjects
  - E.g., subjects for which ring policy is applied are trusted to be able to correctly handle inputs;

Strict integrity policy (BLP reversed)

- Rules:
  - s can read o iff $i(s) \leq i(o)$
    - **no read down**
    - stops indirect sabotage by contaminated data
  - s can write to o iff $i(s) \geq i(o)$
    - **no write up**
    - stops directly malicious modification

- Fixed integrity levels
- No information path from low object/subject to high object/subject

Subject low-water policy

- Rules
  - s can always read o; after reading $i(s) \leftarrow \min[i(s), i(o)]$
  - s can write to o iff $i(s) \geq i(o)$

- Subject's integrity level decreases as reading lower integrity data

- No information path from low-object to high-object

Object low-water mark policy

- Rules
  - s can read o; iff $i(s) \leq i(o)$
  - s can always write to o; after writing $i(o) \leftarrow \min[i(s), i(o)]$
- Object's integrity level decreases as it is contaminated by subjects
- In the end, objects that have high labels have not been contaminated

Low-water mark integrity audit policy

- Rules
  - s can always read o; after reading
    $i(s) \leftarrow \min[i(s), i(o)]$
  - s can always write to o; after writing
    $i(o) \leftarrow \min[i(s), i(o)]$

- Tracing, but not preventing contamination
- Similar to the notion of tainting in software security

Ring policy

- Rules
  - Any subject can read any object
  - s can write to o iff $i(s) \geq i(o)$
- Integrity levels of subjects and objects are fixed.
- Intuitions:
  - subjects are trusted to process low-level inputs correctly

Object integrity levels

- The integrity level of an object may be based on
  - Quality of information (levels may change)
    - Degree of trustworthiness
    - Contamination level:
  - Importance of the object (levels do not change)
    - Degree of being trusted
    - Protection level: writing to the objects should be protected
- What should be the relationship between the two meanings, which one should be higher?

Trusted vs. trustworthy

- A component of a system is trusted means that
  - the security of the system depends on it
  - failure of component can break the security policy
  - determined by its role in the system

- A component is trustworthy means that
  - the component deserves to be trusted
  - e.g., it is implemented correctly
  - determined by intrinsic properties of the component

Integrity vs. Confidentiality

Confidentiality	Integrity
Control reading preserved if confidential info is not read	Control writing preserved if important obj is not changed
For subjects who need to read, control writing after reading is sufficient, no need to trust them	For subjects who need to write, has to trust them, control reading before writing is not sufficient

Integrity requires trust in subjects!

Key difference between confidentiality and integrity

- For confidentiality, controlling reading & writing is sufficient
  - theoretically, no subject needs to be trusted for confidentiality; however, one does need trusted subjects in BLP to make system realistic

- For integrity, controlling reading and writing is insufficient
  - one has to trust all subjects who can write to critical data

Impacts of The Need to Trust Subjects

- Trusting only a small security kernel is no longer possible
- No need to worry about covert channels for integrity protection
- How to establish trust in subjects becomes a challenge

Application of Integrity Protection

- Mandatory Integrity Control in Windows (since Vista)
  - Uses four integrity levels: Low, Medium, High, and System
  - Each process is assigned a level, which limit resources it can access
  - Processes started by normal users have Medium
  - Elevated processes have High
    - Through the User Account Control feature
  - Some processes run as Low, such as IE in protected mode
  - Reading and writing do not change the integrity level
    - Ring policy.

The Clark-Wilson Model

- David D. Clark and David R. Wilson. “A Comparison of Commercial and Military Computer Security Policies.” In IEEE SSP 1987.
- Military policies focus on preventing disclosure
- In commercial environment, integrity is paramount
  - no user of the system, even if authorized, may be permitted to modify data items in such a way that assets or accounting records of the company are lost or corrupted

Two High-level Mechanisms for Enforcing Data Integrity

- Well-formed transaction
  - a user should not manipulate data arbitrarily, but only in constrained ways that preserve or ensure data integrity
    - e.g., use an append-only log to record all transactions
    - e.g., double-entry bookkeeping
    - e.g., passwd

Can manipulate data only through trusted code!

Two High-level Mechanisms for Enforcing Data Integrity

- Separation of duty
  - ensure external consistency: data objects correspond to the real world objects
  - separating all operations into several subparts and requiring that each subpart be executed by a different person
  - e.g., the two-man rule

Implementing the Two High-level Mechanisms

- Mechanisms are needed to ensure
  - control access to data: a data item can be manipulated only by a specific set of programs
  - program certification: programs must be inspected for proper construction, controls must be provided on the ability to install and modify these programs
  - control access to programs: each user must be permitted to use only certain sets of programs
  - control administration: assignment of people to programs must be controlled and inspected

The Clarke-Wilson Model for Integrity

- Unconstrained Data Items (UDIs)
  - data with low integrity
- Constrained Data Items (CDIs)
  - data items within the system to which the integrity model must apply
- Integrity Verification Procedures (IVPs)
  - confirm that all of the CDIs in the system conform to the integrity specification
- Transformation Procedures (TPs)
  - well-formed transactions

Differences from MAC/BLP

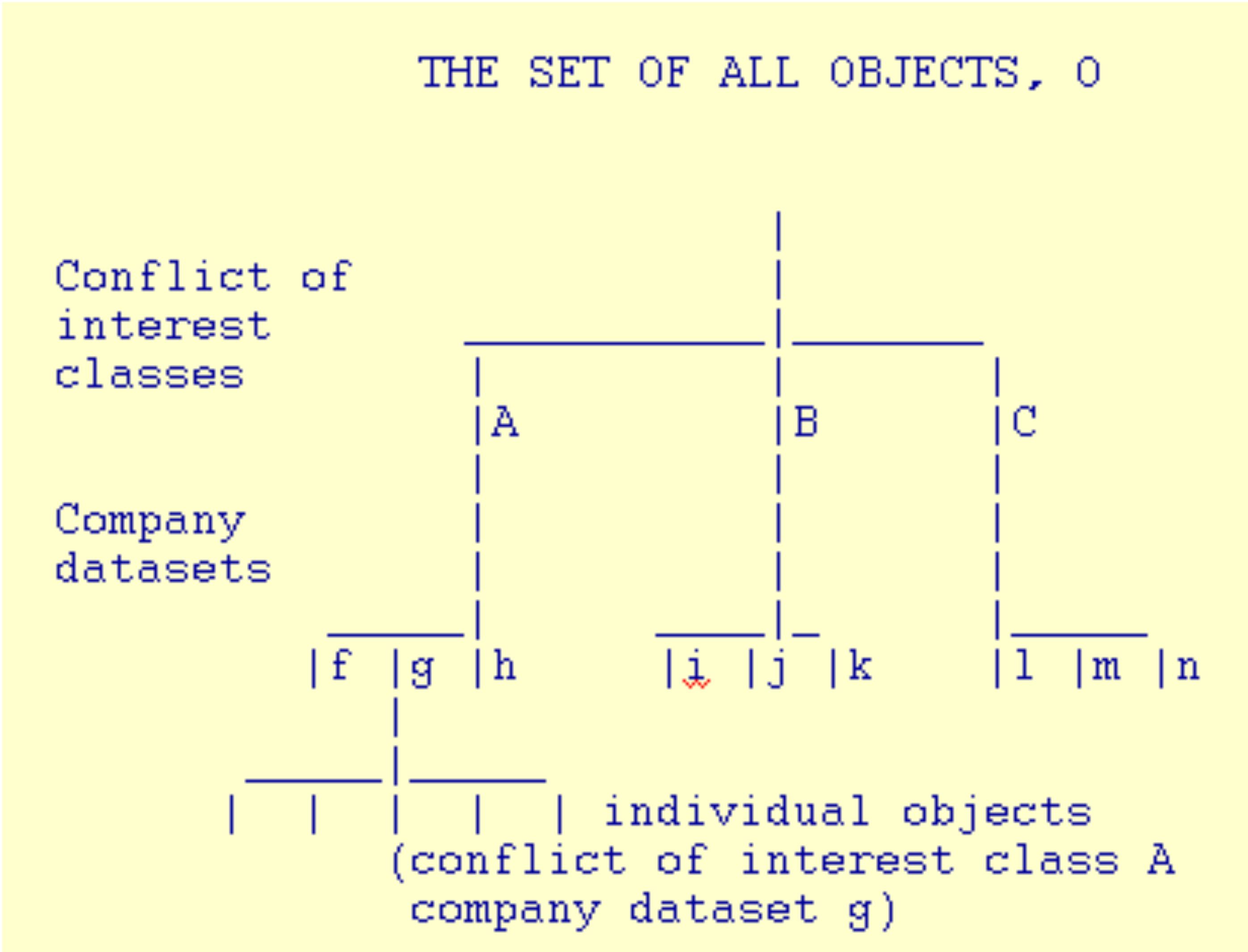
- A data item is not associated with a particular security level, but rather with a set of TPs
- A user is not given read/write access to data items, but rather permissions to execute certain programs

Comparison with Biba

- Biba lacks the procedures and requirements on identifying subjects as trusted
- Clark-Wilson focuses on how to ensure that programs can be trusted

The Chinese Wall Security Policy

- Goal: Avoid Conflict of Interest
- Data are stored in a hierarchical arranged system
  - the lowest level consists of individual data items
  - the intermediate level group data items into company data sets
  - the highest level group company datasets whose corporation are in competition


THE SET OF ALL OBJECTS, O
Conflict of interest classes
A
B
C
Company datasets
f
g
h
i
j
k
l
m
n
individual objects (conflict of interest class A company dataset g)

Simple Security Rule in Chinese Wall Policy

- Access is only granted if the object requested:
  - is in the same company dataset as an object already accessed by that subject, i.e., within the Wall, or belongs to an entirely different conflict of interest class.