Data Security and Privacy

Topic 5: The Bell LaPadula Model

Announcements

- Quiz on Thursday Jan 25

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)

- There are security classifications or security levels
  - Users/principals/subjects have security clearances
  - Objects have security classifications
- Example of security levels
  - Top Secret
  - Secret
  - Confidential
  - Unclassified
- In this case Top Secret > Secret > Confidential > Unclassified
- Security goal (confidentiality): ensures that information do not flow to those not cleared for that level

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, and prevent users from obtaining access to information for which they lack authorization.
  - Discretionary access control fails to achieve MLS
- Typically use Mandatory Access Control
- Primary Security Goal: Confidentiality

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 Model: A MAC Model for Achieving Multi-level Security

- Introduce 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

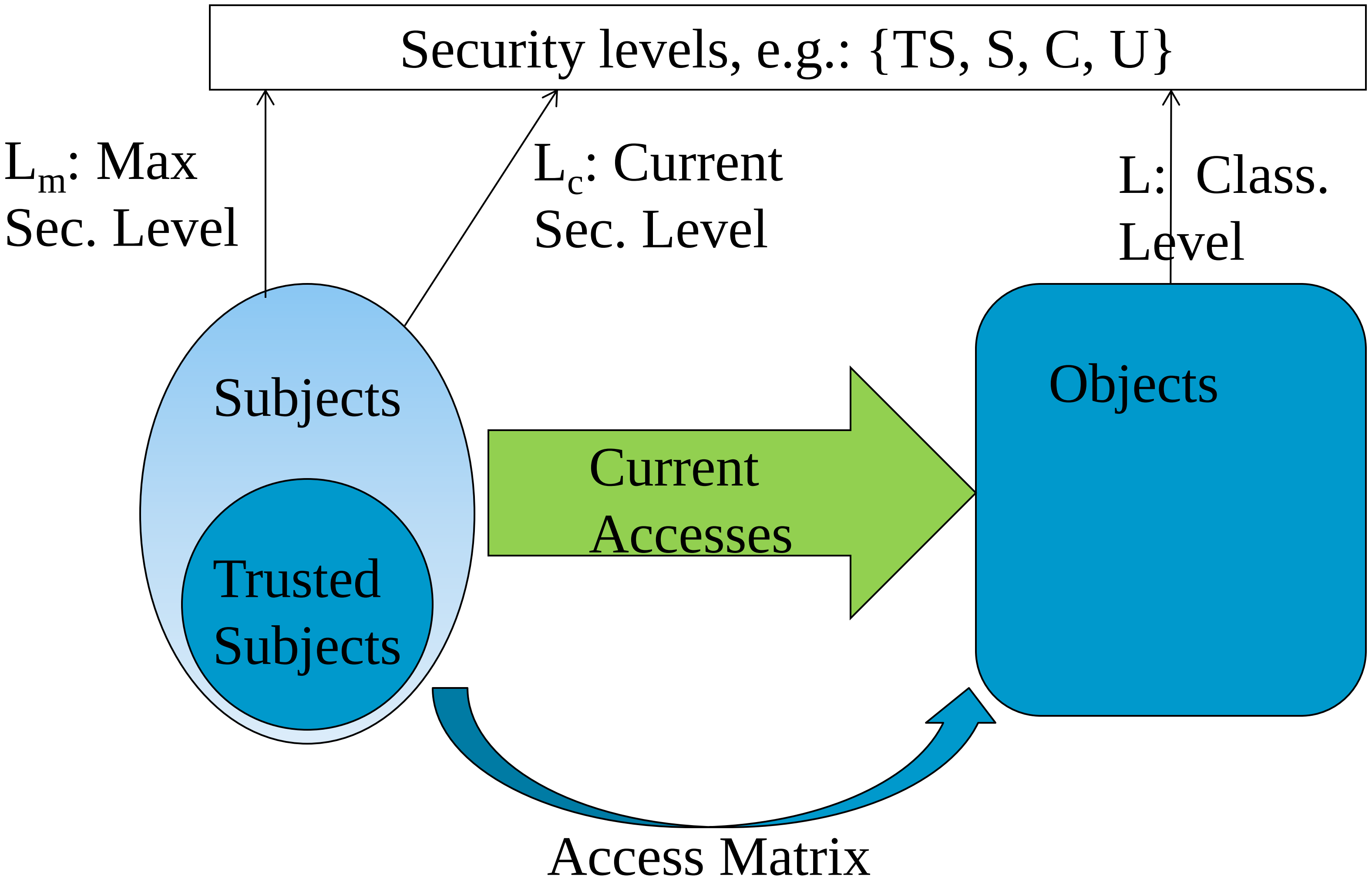
- Define a system as secure iff. every reachable state satisfies 3 properties
  - simple-security property, *-property, discretionary-security property

- Prove a Basic Security Theorem (BST)
  - so that give the description of a system, one can prove that the system is secure

The BLP Security 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 sec level $L_m(s)$, and a current sec level $L_c(s)$
  - Each object has a classification level

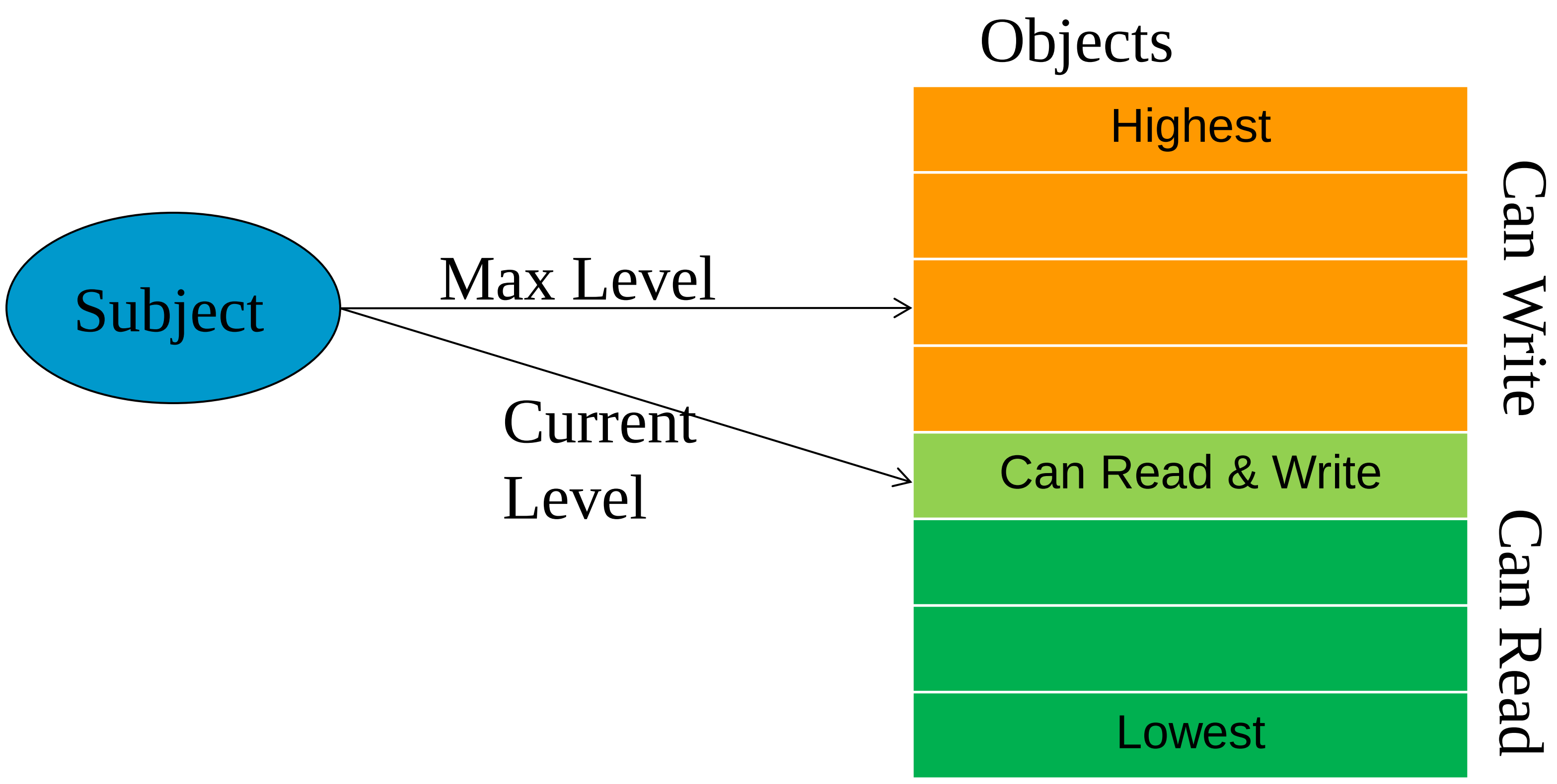
Elements of the BLP Model



The BLP Security Policy

- A state is secure if it satisfies
  - Simple Security Condition (no read up):
    - S can read O iff $L_m(S) \geq L(O)$
  - The Star Property (no write down): for any S that is not trusted
    - S can read O iff $L_c(S) \geq L(O)$ (no read up)
    - S can write O iff $L_c(S) \leq L(O)$ (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.

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 and does not address the covert channel problem

Is BLP Notion of Security Good?

- The objective of BLP security is to ensure
  - a subject cleared at a low level should never read information classified high
- The ss-property and the *-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 s_1, s_2, o_1, o_2
  - $f_S(s_1)=f_C(s_1)=f_O(o_1)=high$
  - $f_S(s_2)=f_C(s_2)=f_O(o_2) =low$
- And the following execution
  - s_1 gets access to o_1, read something, release access, then change current level to low, get write access to o_2, write to o_2
- 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 *-property
  - ss-property is not enough to stop illegal information flow

Other Limitations with BLP

- Deal only with confidentiality, does not deal with integrity at all
  - Confidentiality is often not as important as integrity in most situations
  - Addressed by integrity models (such as Biba, Clark-Wilson, which we will cover later)

- Does not deal with information flow through covert channels

Overt (Explicit) Channels vs. Covert Channels

- Security objective of MLS in general, BLP in particular is
  - high-classified information cannot flow to low-cleared users
- Illegal information flow via overt channels (e.g., read/write an object) is blocked by BLP
- Illegal information flow by covert channels can still occur
  - 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

More on Covert Channels

- Covert channels cannot be blocked by *-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
- Covert channels are achieved by collaboration or high and low subjects.

More on MLS: Security Levels

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

Security Categories

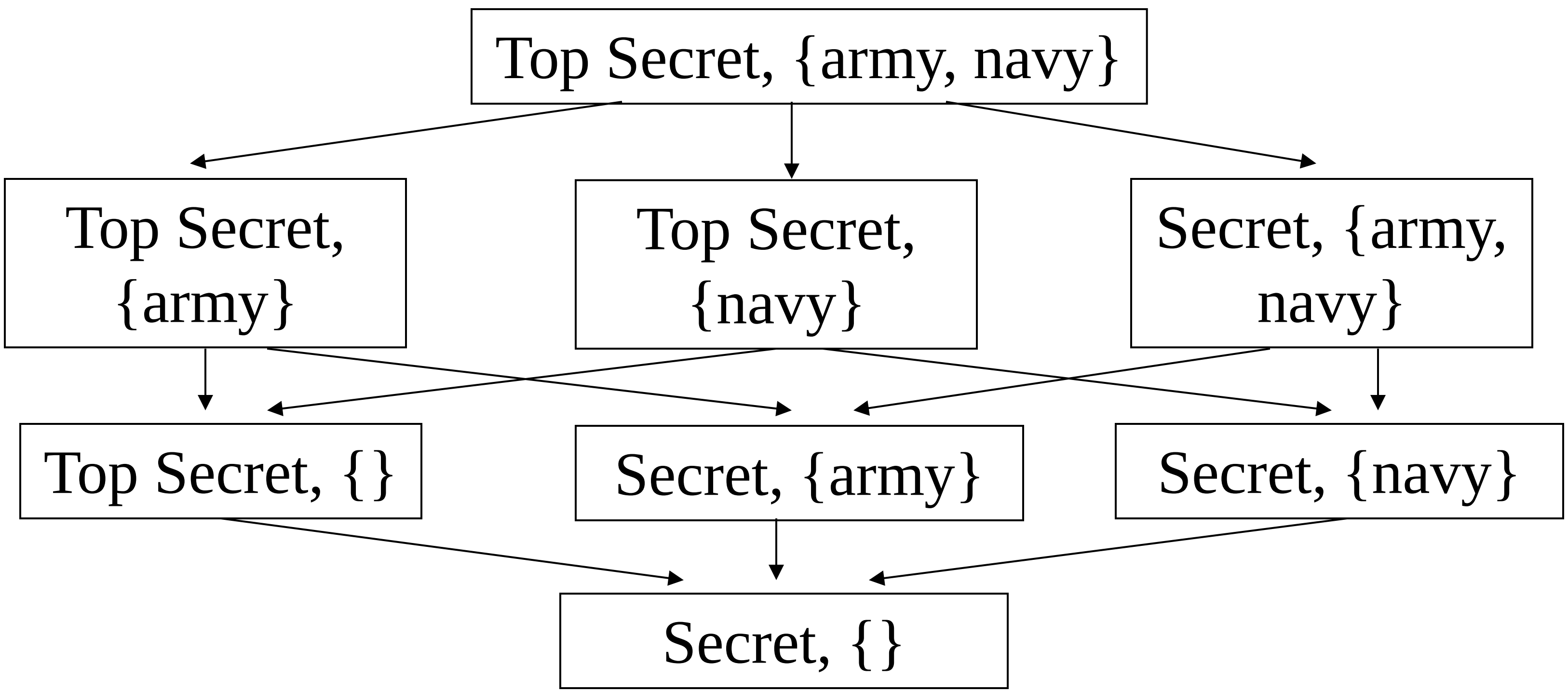
- Also known as compartments
- Typical military security categories
  - army, navy, air force
  - nato, nasa, noforn
- Typical commercial security categories
  - Sales, R&D, HR
  - Dept A, Dept B, Dept C

Security Labels

- Labels = Levels $\times$ 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

Terminology: Trusted Computing Base (TCB)

- The set of all hardware, software and procedural components that enforcing the security policy depends upon.
  - In order to break security, an attacker must subvert some part of the TCB.
  - The smaller the TCB, the more secure a system is.
- What consists of the conceptual Trusted Computing Based in a Unix/Linux system?
  - Depends on the security objective
  - hardware, kernel, system binaries, system configuration files, setuid root programs, etc., at the minimum

One approach to improve security is to reduce the size of TCB, i.e., reduce what one relies on for security.

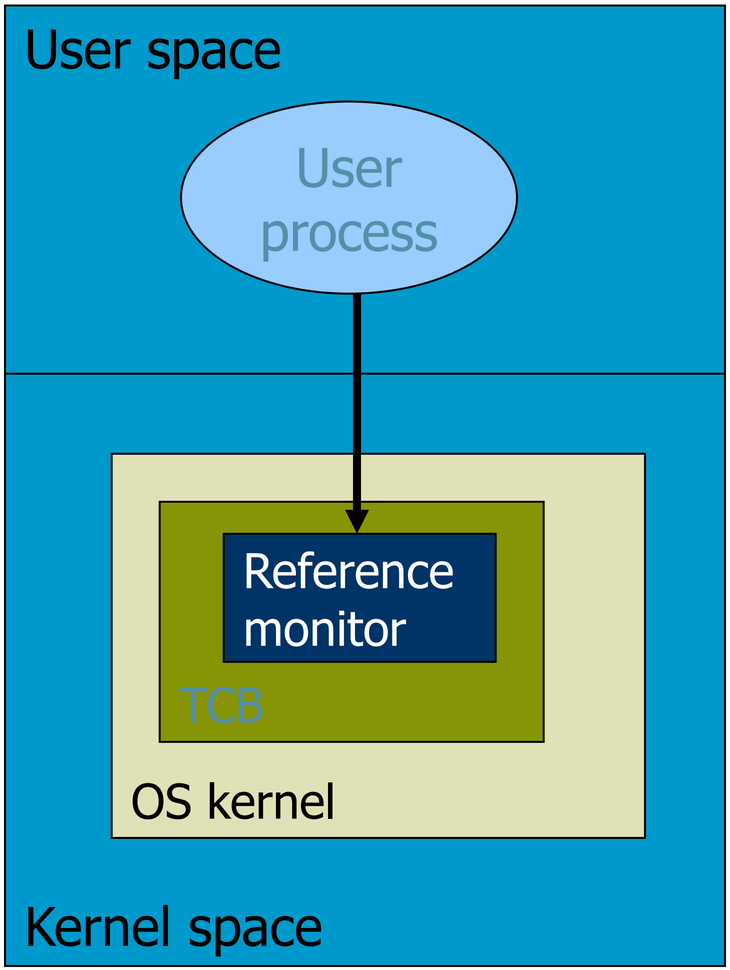
Assurance

- Assurance: “estimate of the likelihood that a system will not fail in some particular way”
- Based on factors such as
  - Software architecture
    - E.g., kernelized design,
  - Development process
  - Who developed it
  - Technical assessment

Kernelized Design for High-Assurance Systems

- Uses the reference monitor concept

- Reference monitor
  - Part of TCB
  - All system calls go through reference monitor for security checking
  - Security does not depends on the whole kernel
  - Most OS not designed this way



Reference Monitor

- Three required properties for reference monitors in high-assurance systems
  - tamper-proof
  - non-bypassable (complete mediation)
  - small enough to be analyzable

Assurance Criteria

- Criteria are specified to enable evaluation
- Originally motivated by military applications, but now is much wider
- Examples
  - Orange Book (Trusted Computer System Evaluation Criteria)
  - Common Criteria

TCSEC: 1983–1999

- Trusted Computer System Evaluation Criteria
  - Also known as the Orange Book
  - Series that expanded on Orange Book in specific areas was called *Rainbow Series*
  - Developed by National Computer Security Center, US Dept. of Defense
- Heavily influenced by Bell-LaPadula model and reference monitor concept
- Emphasizes confidentiality

Coming Attractions …

- Integrity Protection