Last time

- User Authentication
  - Beyond passwords
  - Biometrics
- Security Policies and Models
  - Trusted Operating Systems and Software
  - Military and Commercial Security Policies

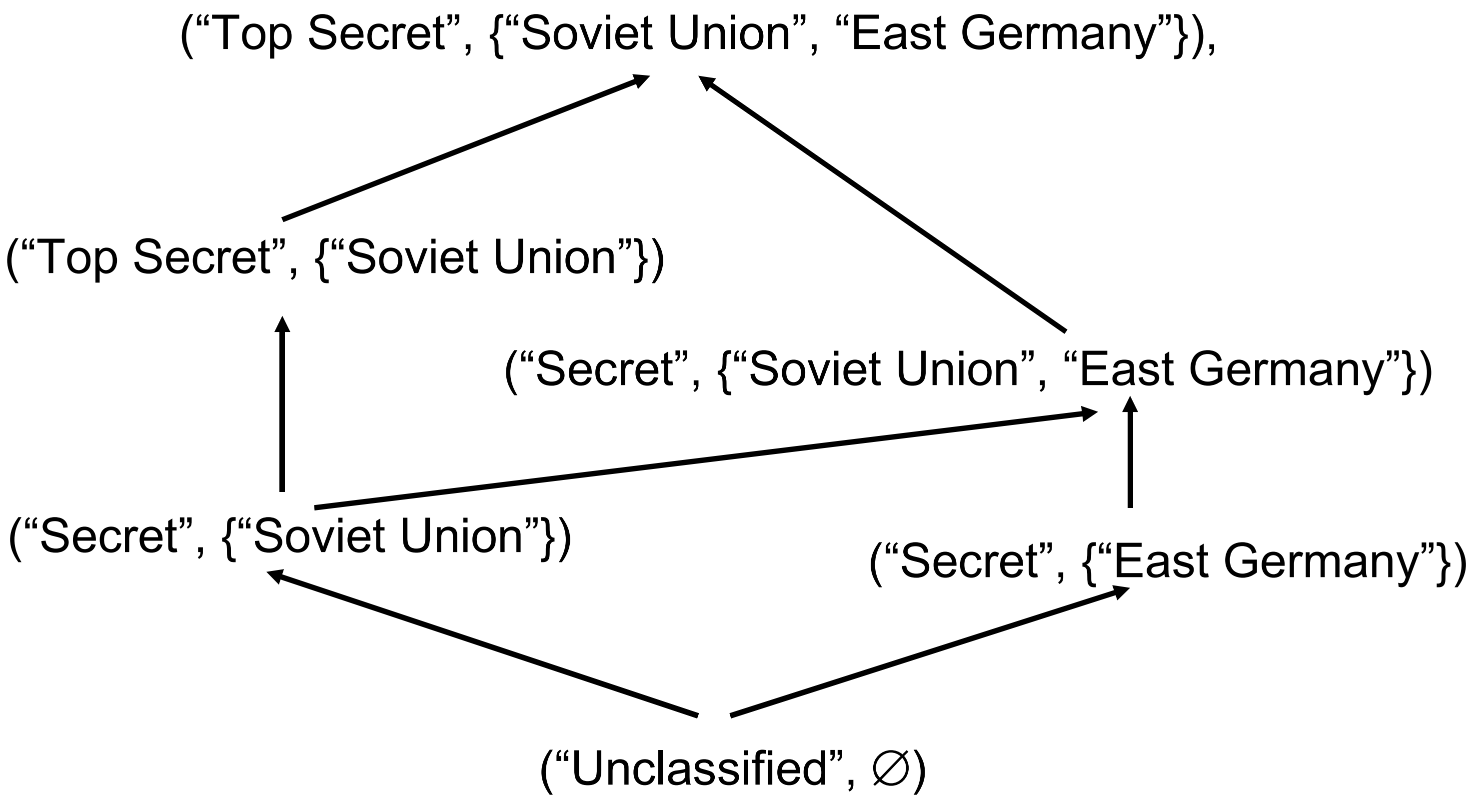
This time

- Security Policies and Models
  - Bell La-Padula and Biba Security Models
  - Information Flow Control
- Trusted Operating System Design
  - Design Elements
  - Security Features

Lattices

- Dominance relationship ≥ defined in military security model is transitive and antisymmetric
- Therefore, it defines a lattice
- For two levels a and b, neither a ≥ b nor b ≥ a might hold
- However, for every a and b, there is a lowest upper bound u for which u ≥ a and u ≥ b and a greatest lower bound l for which a ≥ l and b ≥ l
- There are also two elements U and L that dominate/are dominated by all levels
  - In example,
    U = ("Top Secret", {"Soviet Union", "East Germany"})
    L = ("Unclassified", ∅)

Example Lattice



Bell-La Padula Confidentially Model

- Regulates information flow in MLS policies, e.g., lattice-based ones
- Users should get information only according to their clearance
- Should subject s with clearance C(s) have access to object o with classification C(o)?
- ss-property ("no read up"): s should have read access to o only if $C(s) \geq C(o)$
- *-property ("no write down"): s should have write access to o only if $C(o) \geq C(s)$
  - *Comment: Version in textbook is a little bit less strict, but this is how *-property is typically defined*

Example

- No read up is straightforward
- No write down avoids the following information leak:
  - James Bond reads top-secret document and summarizes it in a confidential document
  - Miss Moneypenny with clearance “confidential” now gets access to top-secret information
- In practice, subjects are really programs (acting on behalf of users)
  - Else James Bond couldn’t even talk to Miss Moneypenny
  - If program accesses top-secret information, OS ensures that it won’t be allowed to write to confidential file later
  - Even if program does not leak information to confidential file
  - Might need explicit declassification operation for usability purposes

Biba Integrity Model

- Prevent inappropriate modification of data
- Dual of Bell-La Padula model
- Subjects and objects are ordered by an integrity classification scheme, I(s) and I(o)
- Should subject s have access to object o?
- Write access: s can modify o only if $I(s) \geq I(o)$
  - Unreliable person cannot modify file containing high integrity information
- Read access: s can read o only if $I(o) \geq I(s)$
  - Unreliable information cannot “contaminate” subject

Low Watermark Property

- Biba's access rules are very restrictive, a subject cannot ever view lower integrity object
- Can use dynamic integrity levels instead
  - Subject Low Watermark Property:
    If subject s reads object o, then I(s) = glb(I(s), I(o)), where glb() = greatest lower bound
  - Object Low Watermark Property:
    If subject s modifies object o, then I(o) = glb(I(s), I(o))
- Integrity of subject/object can only go down, information flows down
- What kind of property does Bell-La Padula imply?

Review of Bell-La Padula & Biba

- Very simple, which makes it possible to prove properties about them
  - E.g., can prove that if a system starts in a secure state, the system will remain in a secure state
- Probably too simple for great practical benefit
  - Need declassification
  - Need both confidentially and integrity, not just one
  - What about object creation?
- Information leaks might still be possible through covert channels

Information Flow Control

- An information flow policy describes authorized paths along which information can flow
- For example, Bell-La Padula describes a lattice-based information flow policy
- In compiler-based information flow control, a compiler checks whether the information flow in a program could violate an information flow policy
- How does information flow from a variable x to a variable y?
- Explicit flow: E.g., `y:= x;` or `y:= x / z;`
- Implicit flow: `If x = 1 then y := 0; else y := 1`

Information Flow Control (cont.)

- See text for other sample statements
- Input and output variables of program each have a (lattice-based) security classification S() associated with them
- For each program statement, compiler verifies whether information flow is secure
- For example, `x := y + z` is secure only if S(`x`) ≥ lub(S(`y`), S(`z`)), where lub() is lowest upper bound
- Program is secure if each of its statements is secure

Trusted System Design Elements

- Design must address which objects are accessed how and which subjects have access to what
  - As defined in security policy and model
- Security must be part of design early on
  - Hard to retrofit security, see Windows 95/98
- Design principles for security
  - Least privilege
    - Operate using fewest privileges possible
  - Economy of mechanism
    - Protection mechanism should be simple and straightforward
  - Open design
    - Avoid security by obscurity
    - Rely on secret keys or passwords, but not on secret algorithms

Security Design Principles (cont.)

- Complete mediation
  - Every access attempt must be checked
- Permission based
  - Default should be denial of access
- Separation of privileges
  - Two or more conditions must be met to get access
- Least common mechanism
  - Every shared mechanism could potentially be used as a covert channel
- Ease of use
  - If protection mechanism is difficult to use, nobody will use it or it will be used in the wrong way

Security Features of Trusted OS

- Identification and authentication
  - See earlier
- Access control
- Object reuse protection
- Complete mediation
- Trusted path
- Accountability and audit
- Intrusion detection

Access Control

- Mandatory access control (MAC)
  - Central authority establishes who can access what
  - Good for military environments
  - For implementing Chinese Wall, Bell-La Padula, Biba
- Discretionary access control (DAC)
  - Owners of an object have (some) control over who can access it
  - You can grant others access to your home directory
  - In UNIX, Windows,…
- RBAC is neither MAC nor DAC
- Possible to use combination of these mechanisms

Object Reuse Protection

- Alice allocates memory from OS and stores her password in this memory
- After using password, she returns memory to OS
  - By calling free() or simply by exiting procedure if memory is allocated on stack
- Later, Bob happens to be allocated the same piece of memory and he finds Alice's password in it
- OS should erase returned memory before handing it out to other users
- Defensive programming: Erase sensitive data yourself before returning it to OS
- Similar problem exists for files, registers and storage media

Hidden Data

- Hidden data is related to object reuse protection
- You think that you deleted some data, but it is still hidden somewhere
  - Deleting a file will not physically erase file on disk
  - Deleting an email in GMail will not remove email from Google's backups
  - Deleting text in MS Word might not remove text from document
  - Putting a black box over text in a PDF leaves text in PDF
  - Shadow Copy feature of Windows Vista keeps file snapshots to enable restores

Complete Mediation / Trusted Path

- Complete mediation
  - All accesses must be checked
  - Preventing access to OS memory is of little use if it is possible to access the swap space on disk
- Trusted path
  - Give assurance to user that her keystrokes and mouse clicks are sent to legitimate receiver application
  - Remember the fake login screen?
  - Turns out to be quite difficult for existing desktop environments, both Linux and Windows
    - Don’t run sudo if you have an untrusted application running on your desktop

Recap

- Security Policies and Models
  - Bell La-Padula and Biba Security Models
  - Information Flow Control
- Trusted Operating System Design
  - Design Elements
  - Security Features

Next time

- Trusted Operating System Design
  - Security Features
  - Trusted Computing Base
  - Least Privilege in Popular OSs
  - Assurance

- Security in Networks
  - Network Concepts
  - Threats in Networks
  - Network Security Controls
  - Firewalls
  - Intrusion Detection Systems