Systems and Internet Infrastructure Security

Network and Security Research Center
Department of Computer Science and Engineering
Pennsylvania State University, University Park PA

Advanced Systems Security: Mandatory Access Control Models

Trent Jaeger
Systems and Internet Infrastructure Security (SIIS) Lab
Computer Science and Engineering Department
Pennsylvania State University

February 9, 2010

Reference Monitor Components

- Interface
  - Where to make access control decisions (mediation)
  - Which access control decisions to make (authorization)
  - Linux Security Modules interface
- Decision function
  - Compute decision based on request and policy
  - E.g., SELinux, LIDS, DTE, etc. modules
- Policy – our focus today
  - How to represent access control policy
  - Main mechanism issue – find mechanism to enable verification that policy achieves function and meets security guarantees

Access Control

- Determine whether a **principal** can perform a requested **operation** on a target **object**
- **Principal:** user, process, etc.
- **Operation:** read, write, etc.
- **Object:** file, tuple, etc.
- Lampson defined the familiar **access matrix** and its two interpretations ACLs and capabilities [Lampson70]

Why are we still talking about access control?

- An **access control policy** is a specification for an **access decision function**
- The policy aims to achieve
  - Permit the principal's intended function (availability)
  - Ensure security properties are met (integrity, confidentiality)
    - Limit to "Least Privilege," Protect system integrity, Prevent unauthorized leakage, etc.
    - Also known as 'constraints'
  - Enable administration of a changeable system (simplicity)

“Simple” example

- Prof Alice manages access to course objects
  - Assign access to individual (principal: Bob)
  - Assign access to aggregate (course-students)
  - Associate access to relation (students(course))
  - Assign students to project groups (student(course, project, group))
- Prof Alice wants certain guarantees
  - Students cannot modify objects written by Prof Alice
  - Students cannot read/modify objects of other groups
- Prof Alice must be able to maintain access policy
  - Ensure that individual rights do not violate guarantees
  - However, exceptions are possible – students may distribute their results from previous assignments for an exam

Access Control is Hard Because

- Access control requirements are domain-specific
  - Generic approaches over-generalize
- Access control requirements can change
  - Anyone could be an administrator
- The Safety Problem [HRU76]
  - Can only know what is leaked right now
- Access is fail-safe, but Constraints are not
  - And constraints must restrict all future states

Safety Problem [HRU76]

- Determine if an unauthorized permission is leaked given
  - An initial set of permissions and
  - An access control system, mainly administrative operations
- For a traditional approach, the safety problem is *undecidable*
  - Access matrix model with multi-operational commands
  - Main culprit is create – create object/subject with **own** rights
  - Prove reduction of a Turing machine to the multi-operational access matrix system
- Result led to
  - Safe, but limited models: take-grant, schematic protection model, typed access matrix model
  - Further support for models in which the constraints are implicit in the model – e.g., lattice models
  - Check safety on each policy change – constraint approach of RBAC

Compare to Other CS Problems

- Processor design
  - Hard, but can get some smart people together to construct one, fixed, testable design
- Network protocol design
  - TCP: A small number of control parameters necessary to manage all reasonable options, within a layered architecture
  - Constraints, such as DDoS, are ad hoc
- Software design
  - Specific goals in mind to achieve function, constraints are ad hoc

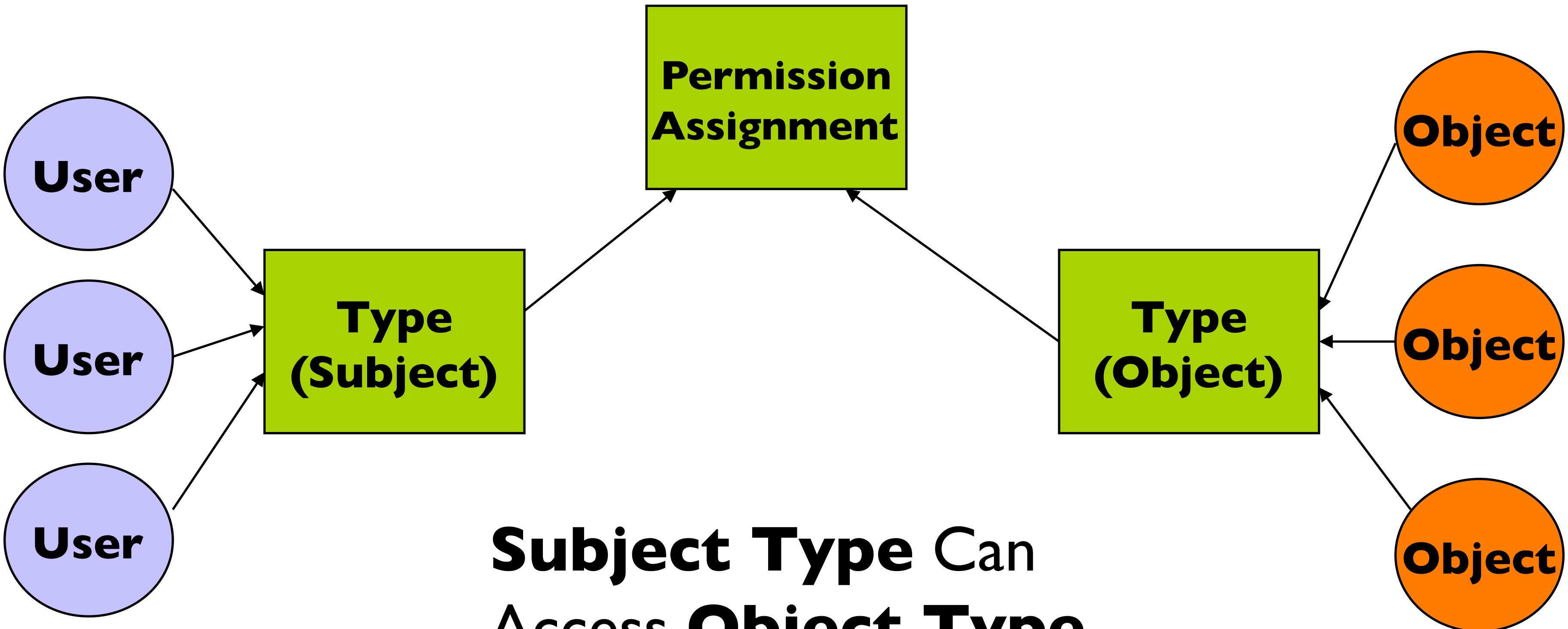
Access Control Models

- Discretionary Access Matrix
  - UNIX, ACL, various capability systems
- Mandatory (Usually) Access Matrix
  - TE, RBAC, groups and attributes, parameterized
- Plus Transitions
  - DTE, SELinux, Java
- Lattice Access Control Models
  - Bell-LaPadula, Biba, Denning
- Predicate Models
  - ASL, OASIS, domain-specific models, many others
- Safety Models
  - Take-grant, Schematic Protection Model, Typed Access Matrix

Administration

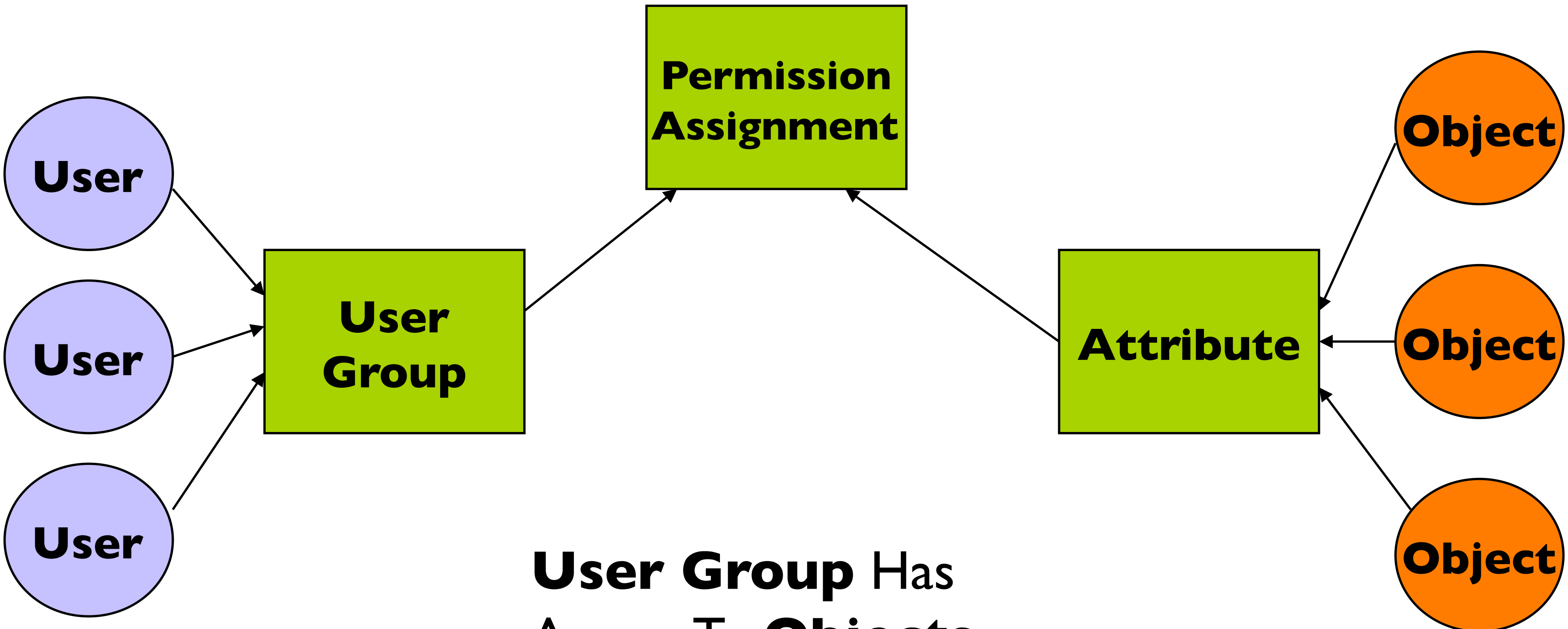
- Discretionary Access Control
  - Users (typically object owner) can decide permission assignments
- Mandatory Access Control
  - System administrator decides on permission assignments
- Flexible Administrative Management
  - Access control models can be used to express administrative privileges

Type Enforcement [BoebertKain84]



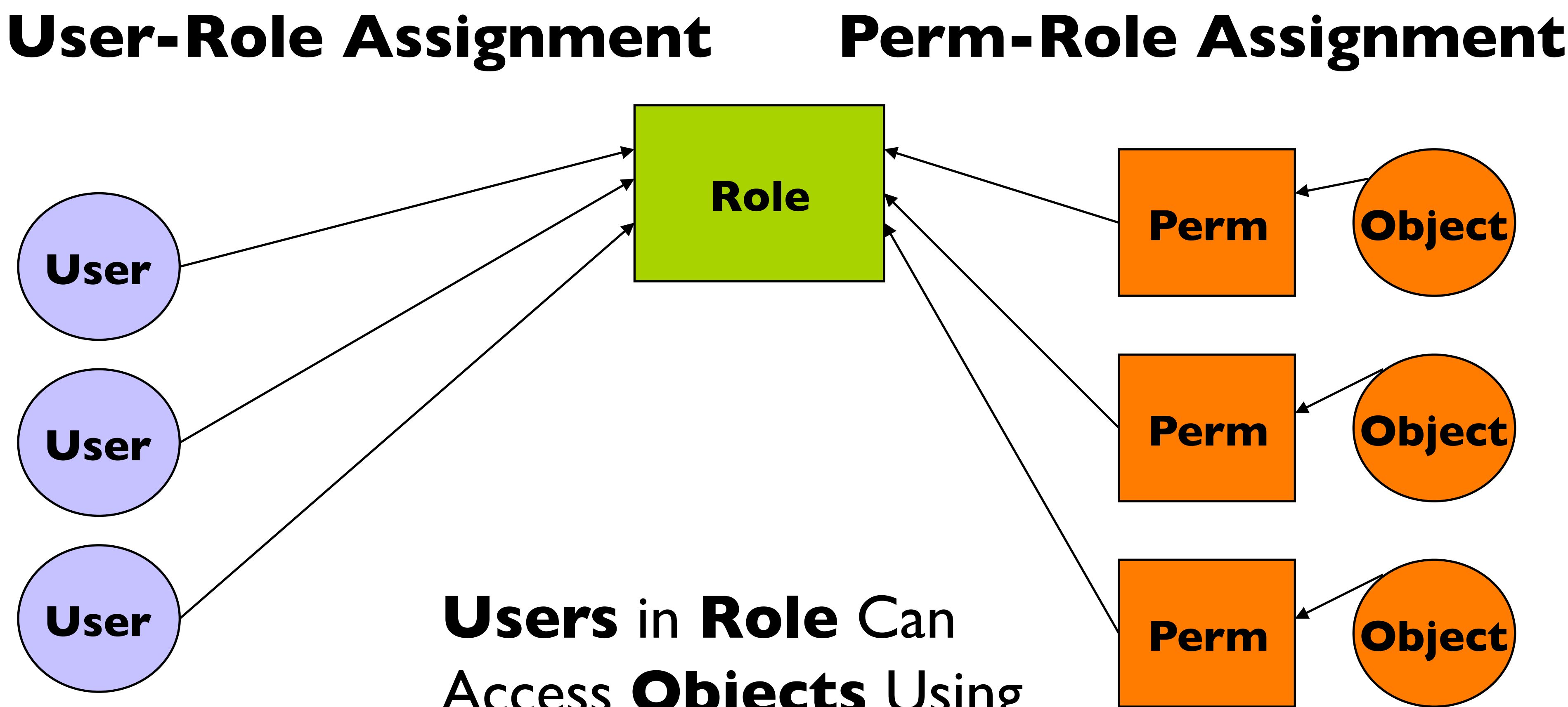
Subject Type Can Access **Object Type** To Perform Operations On **Objects**

Group and Attributes



User Group Has
Access To **Objects**
With the **Attribute**

Role-based Access Control



Users in **Role** Can Access **Objects** Using **Permissions**

Role vs. Types Data Structures

- RBAC
  - U: set of users
  - P: set of permissions
  - R: set of roles
- Type Enforcement
  - E: set of subjects or objects
  - Permission Assignment
    - ST: set of subject types
    - OT: set of object types
    - O: set of operations

Role-based Access Control Model

- Users: U
- Permissions: P
- Roles: R
- Assignments: User-role, perm-role, **role-role**
- **Sessions: S**
- Function: user(S), roles(S)
- **Constraints: C**

RBAC Family of Models

- $RBAC_0$ contains all but hierarchies and constraints
- $RBAC_1$ contains $RBAC_0$ and hierarchies
- $RBAC_2$ contains $RBAC_0$ and constraints
- $RBAC_3$ contains all
- The RBAC family idea has always been more a NIST initiative
- The RBAC families are present in the NIST RBAC standard [NIST2001] with slight modifications:
  - $RBAC_0$, $RBAC_1$ (options), $RBAC_3$ (SSD) , $RBAC_3$ (DSD)

RBAC Products

- SUN Solaris
- Sybase SQL Server
- BMC INCONTROL for Security Management
- Systor Security Administration Manager
- Tivoli TME Security Management
- Computer Associates Protect IT
- Siemens rbacDirX

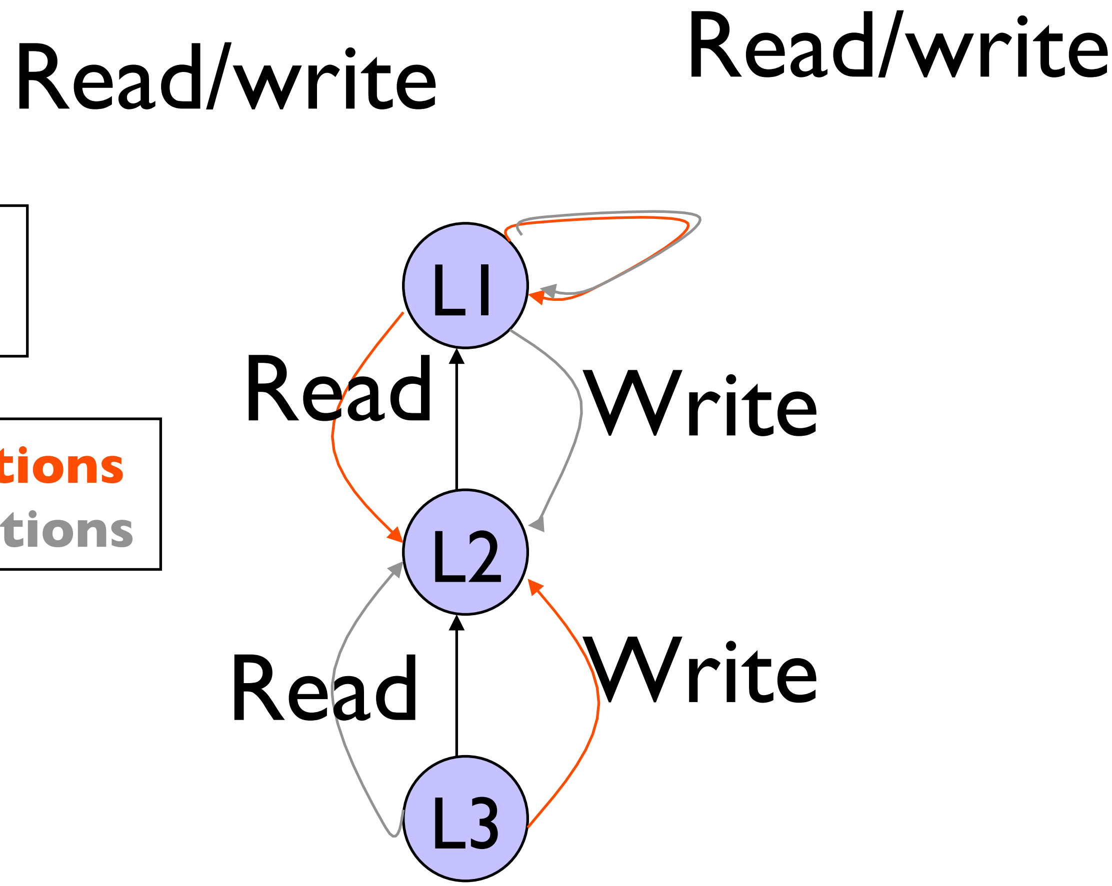
Lattice Access Control Models

- Subjects and Objects have security levels and optional categories
- Confidentiality Policy (e.g., Bell-LaPadula)
  - Simple property: may read only if the subject's security level dominates the object's security level (read-down)
  - *-property: may write only if the subject's security level is dominated by the object's security level (write-up)
  - Tranquility property: may not change the security level of an object concurrent to its use
- Integrity Policy
  - Biba is the dual of BLP for integrity

Security Levels and Policies

Dominance
1 > 2 > 3

BLP Operations
Biba Operations



Purpose of BLP and Biba

- BLP
  - Prevent Trojan horses from leaking information to lower security levels
  - Mandatory access control and implicit constraints
- Biba
  - Prevent low integrity information flows to higher integrity processes
  - E.g., code, configuration, user requests, buffer overflows
- Categories/Compartments for separation within levels
- Safety is implicit in the model
  - No additional constraints are needed to express security guarantees

Denning's Lattice Model

- Formalizes information flow models
  - FM = {N, P, SC, /, t}
- Shows that the information flow model instances form a lattice
  - {SC, t} is a partial ordered set,
  - SC is finite,
  - SC has a lower bound,
  - and / is a lub operator
- Implicit and explicit information flows
- Semantics for verifying that a configuration is secure
- Static and dynamic binding considered
- Biba and BLP are among the simplest models of this type

Implicit and explicit flows

- Explicit
  - Direct transfer to b from a (e.g., b = a)
- Implicit
  - Where value of b may depend on value of a indirectly (e.g., if a = 0, then b = c)
- Model covers all programs
  - Statement S
  - Sequence S1, S2
  - Conditional c: S1, …, Sm
- Implicit flows only occur in conditionals

Semantics

- Program is secure if:
  - Explicit flow from S is secure
  - Explicit flow of all statements in a sequence are secure (e.g., S1; S2)
  - Conditional c:S1, …, Sm is secure if:
    - The explicit flows of all statements S1, …, Sm are secure
    - The implicit flows between c and the objects in Si are secure

Static and Dynamic Binding

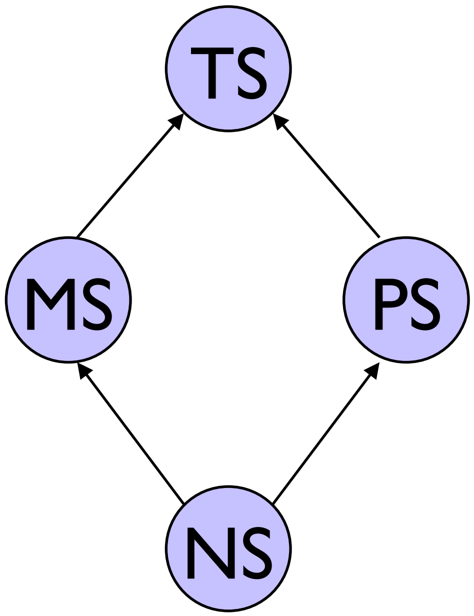
- Static binding
  - Security class of an object is fixed
  - This is the case for BLP and Biba
  - This is not the case for all system models
- Dynamic binding
  - Security class of an object can change
  - For b = a, then the security class of b is b / a
  - Rare approach

Model Examination

- Certification Mechanism
  - Static check eliminates covert channels
  - Limits
    - Language defect could miss a check (buffer overflow)
    - Hardware malfunction
- Approach
  - Verify information flow w/i a statement
    - d = a + b; $\underline{a}$ / $\underline{b}$ t $\underline{d}$; $\underline{d}$ must dominate
  - Set statement security level S = $\underline{d}$
  - Statement sequence S = S1|S2 – must be able to flow to greatest lower bound
  - Verify c t $\underline{d1}$, …, $\underline{dn}$ for implicit flow

Verification Example

d = PS
e = MS
e t d X



d = TS
e = MS
e t d OK
S = S1 1 S2 = MS
c t S, c dominated by MS

Information Flow Plus Models

- For integrity, Biba information flow models are insufficient
  - Integrity is captured by rules
- Consider accounting
  - A balance B = YB + D – W
    - *Where YB is yesterday's balance, D is deposits, and W is withdrawals*
  - The integrity of data in commercial environments is maintained by *well-formed transactions*
- How do we model commercial integrity?

Clark-Wilson Model

- Constrained Data Items: Data with integrity controls
- Unconstrained Data Items: Remaining data
- Integrity Verification Procedures: Check that CDIs satisfy integrity constraints
  - *The integrity of constrained data must be verified before use*
- Transformation Procedures: Take data from one valid state to another
  - *High integrity data may only be modified by transformation procedures that implement well-formed transactions*

Clark-Wilson Model

- Constrained Data Items: Data with integrity controls
- Unconstrained Data Items: Remaining data
- Integrity Verification Procedures: Check that CDIs satisfy integrity constraints
  - *The integrity of constrained data must be verified before use*
- Transformation Procedures: Take data from one valid state to another
  - *High integrity data may only be modified by transformation procedures that implement well-formed transactions*

Clark-Wilson Model

- Consists of a set of certification and enforcement rules governing system function
- Authentication: authenticate trusted personnel (ER3)
- Authorization: only they may run IVPs and TPs (ER2)
- Audit: Log operations on CDIs (CR4)
- Separation of duty: Separate certification and use (ER4)

Clark-Wilson Model

- Its key rules control how data is accessed
- CR1: IVP must ensure all CDIs are in a valid state
- CR2: TPs must be certified to transform CDIs from one valid state to another
- CR5: Any TP that takes a UDI as input must either discard it or upgrade it into a CDI
- Security depends on certification of such properties, but

Chinese Wall Model

- Consider a consulting business
- A consultant is authorized to work for any client, but some clients have secrecy and integrity requirements relative to other clients
  - Coca-Cola and Pespi
- The Chinese Wall model enables definition of such scenarios
  - Only allow subjects to read data from one of the conflicted parties
  - Must control writing too

Chinese Wall Model

- Company Dataset: The set of objects that may belong to a company – CD(O)
- Conflict of Interest Class: Datasets of companies in conflict – COI(O)
    - Each object has only one
- Read iff (CW-Simple Security Property) Let PR(S) be the set of objects that a subject S has already read
    - *If a subject S reads an O belonging to dataset CD, she can never read another O' where CD(O') is a member of COI(O) and CD(O') is not equal CD(O)*
    - Objects can be sanitized

Chinese Wall Model

- What about control of writing?
- Suppose CD1 and CD2 are have a conflict of interest
  - What if one user can read from CD3 and CD1…
  - And another can read from CD3 and CD2?
- Now suppose either user can write to CD3
  - What happens?
- Thus, a writer can only access objects in one dataset

Other Models

- Plus Type Enforcement plus Domain Transitions
  - DTE, SELinux, Java
- Predicate Models
  - ASL, OASIS, domain-specific models, many others
- Safety Models
  - Take-grant, Schematic Protection Model, Typed Access Matrix

Take Away

- Once we have a goal, we need to specify it
    - And manage it
- A mandatory protection system requires system administration
    - To avoid the safety problem
- But, we still need to know that the policy expresses our goals
    - Lots of options
- *Options mainly focus on aggregating expressions (e.g., RBAC) or being more closely mapped to goals*