WILLIAM & MARY

CHARTERED 1693

CSCI 667: Concepts of Computer Security

Prof. Adwait Nadkarni

Access Control Administration

There are two central ways to specify a policy

- *Discretionary* - object "owners" define policy
  - Users have discretion over who has access to what objects and when (trusted users)
  - Canonical example: the UNIX filesystem
    - —RWX assigned by file owners
- *Mandatory* - Environment enforces static policy
  - Access control policy defined by environment, user has no control control over access control (untrusted users)
  - Canonical example: process labeling
    - System assigns labels for processes, objects, and a dominance calculus is used to evaluate rights

DAC vs. MAC

- Discretionary Access Control
  - User defines the access policy
  - Can pass rights onto other subjects (called *delegation*)
  - Their programs can pass their rights
    - Consider a Trojan horse
- Mandatory Access Control
  - System defines access policy
  - Subjects cannot pass rights
  - Subjects' programs cannot pass rights
    - Consider a Trojan horse here

DAC vs. MAC in Access Matrix

- Subjects:
  - DAC: users
  - MAC: labels
- Objects:
  - DAC: files, sockets, etc.
  - MAC: labels
- Operations:
  - Same
- Administration:
  - DAC: owner, copy flag, ...
  - MAC: external, reboot
- MAC: largely static matrix;
- DAC: all can change

	O_1	O_2	O_3
S_1	Y	Y	N
S_2	N	Y	N
S_3	N	Y	Y

Safety Problem

- For a protection system
  - (ref mon, protection state, and administrative operations)
- Prove that any future state will not result in the leakage of an access right to an unauthorized user
  - Q: Why is this important?
- For most discretionary access control models,
  - Safety is *undecideable*
- Means that we need another way to prove safety
  - Restrict the model (no one uses)
  - Test incrementally (constraints)
- How does the safety problem affect MAC models?

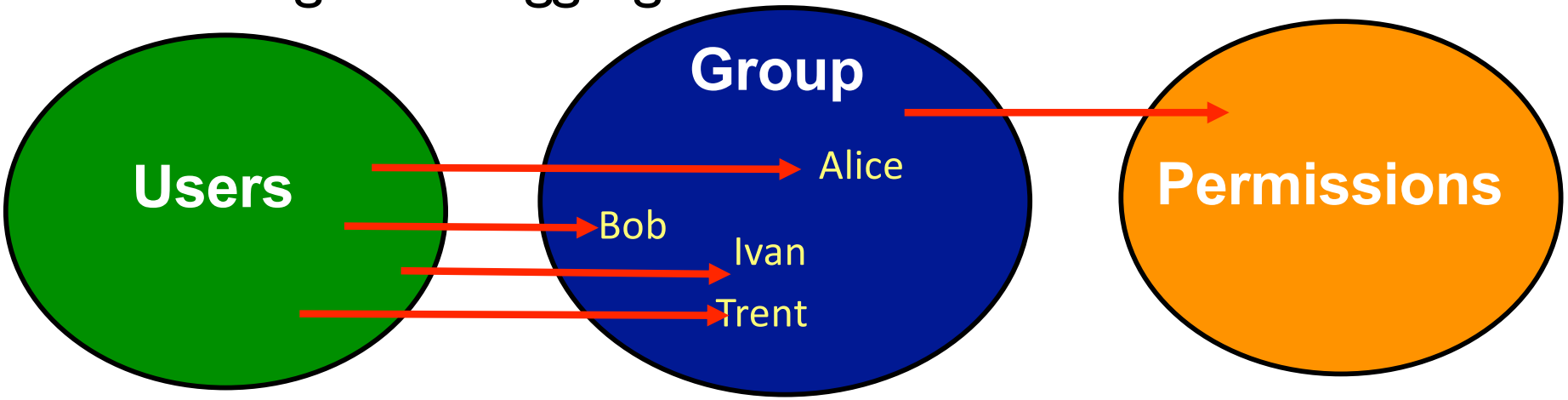
Access Control Models

- What language should I use to express policy?
  - Access Control Model
- Oodles of these
  - Some specialize in secrecy
    - Bell-LaPadula
  - Some specialize in integrity
    - Clark-Wilson
  - Some focus on jobs
    - RBAC
  - Some specialize in least privilege
    - SELinux Type Enforcement
- Q: Why are there so many different models?

RBAC

Groups

- Groups are collections of identities who are assigned rights as a collective
- Important in that it allows permissions to be assigned in aggregates of users …



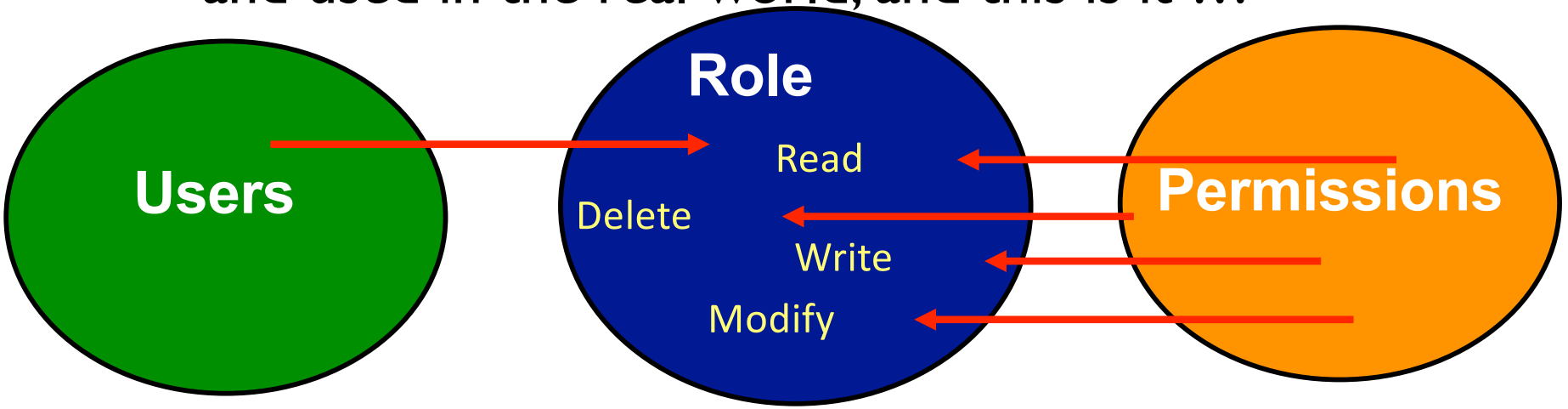
- This is really about "membership"
  - Standard DAC
  - Permissions are transient

Job Functions

- In an enterprise, we don't really do anything as ourselves, we do things as some job function
  - E.g., student, professor, doctor
- *One could manage this as groups, right?*
  - We are assigned to groups all the time, and given similar rights as them, i.e., mailing lists

Roles

- A role is a collection of privileges/permissions associated with some function or affiliation
- NIST studied the way permissions are assigned and used in the real world, and this is it …



- Important: the permissions are static, the user-role membership is transient
- This is not standard DAC

Role Based Access Control

- Role based access control is a class of access control not direct MAC and DAC, but may one or either of these.
- A lot of literature deals with RBAC models
- Most formulations are of the type
  - U: users -- these are the subjects in the system
  - R: roles -- these are the different roles users may assume
  - P: permissions --- these are the rights which can be assumed
- There is a many-to-many relation between:
  - Users and roles
  - Roles and permissions
- Relations define the role-based access control policy

RBAC Sessions

- During a session, a user assumes a subset available roles
  - Known as activating a set of roles
  - The user rights are the union of the rights of the activated roles
    - Note: the session terminates at the user's discretion

- Q: Why not just activate all the roles?

Constraints

- You want to constrain evolution of protection states
  - Constraints are explicit ways of doing just this
  - Constraints available (in RBAC)
    - role assumption
    - perm-role assignment
    - user-role assignment
- Examples in RBAC:
  - *Required inclusion*: You must be acting as an *employee* of Willliam & Mary to be a *professor*
    - You must assume a (parent) role to assume another (child) role
  - *Mutual exclusion*: can not be both CFO and auditor for the same company (unless you work for Enron)
  - *Cardinality constraint*: only *one* (or n) of a particular role

Trusted Processes

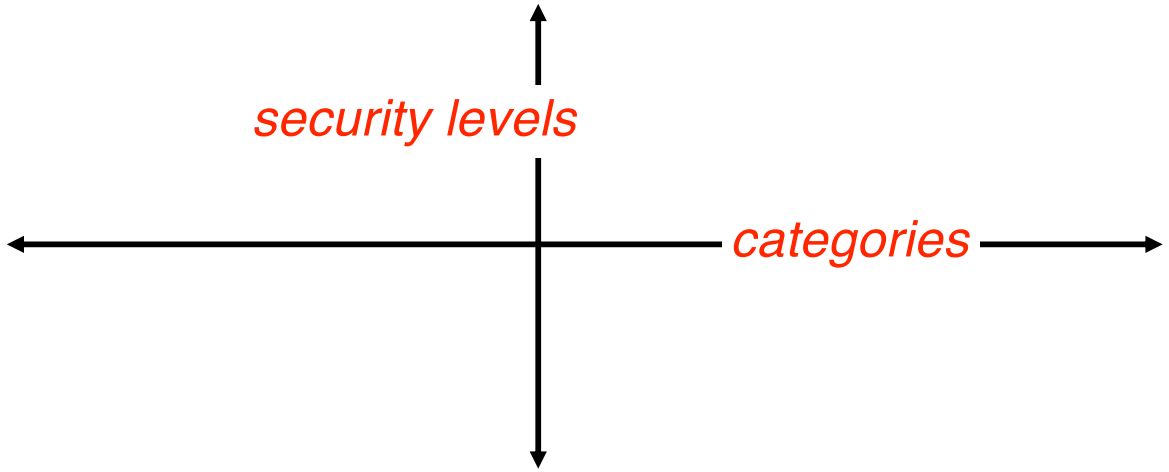
- Does it matter if we do not trust some of J's processes?
  - *Trojan Horse*: Attacker controlled code run by J can violate secrecy.
  - *Confused Deputy*: Attacker may trick trusted code to violate integrity

	O_1	O_2	O_3
J	R	RW	RW
S_2	-	R	RW
S_3	-	R	RW

Information Flow Control

Multilevel Security

- A multi-level security system tags all object and subject with security tags classifying them in terms of sensitivity/access level.
  - We formulate an access control policy based on these levels
  - We can also add other dimensions, called categories which horizontally partition the rights space (in a way similar to that as was done by roles)



US DoD Policy

- Used by the US military (and many others), the Lattice model uses MLS to define policy
- Levels:

UNCLASSIFIED < CONFIDENTIAL < SECRET < TOP SECRET

- Categories (actually unbounded set)

NUC(lear), INTEL(igence), CRYPTO(graphy)

- Note that these levels are used for physical documents in the governments as well.

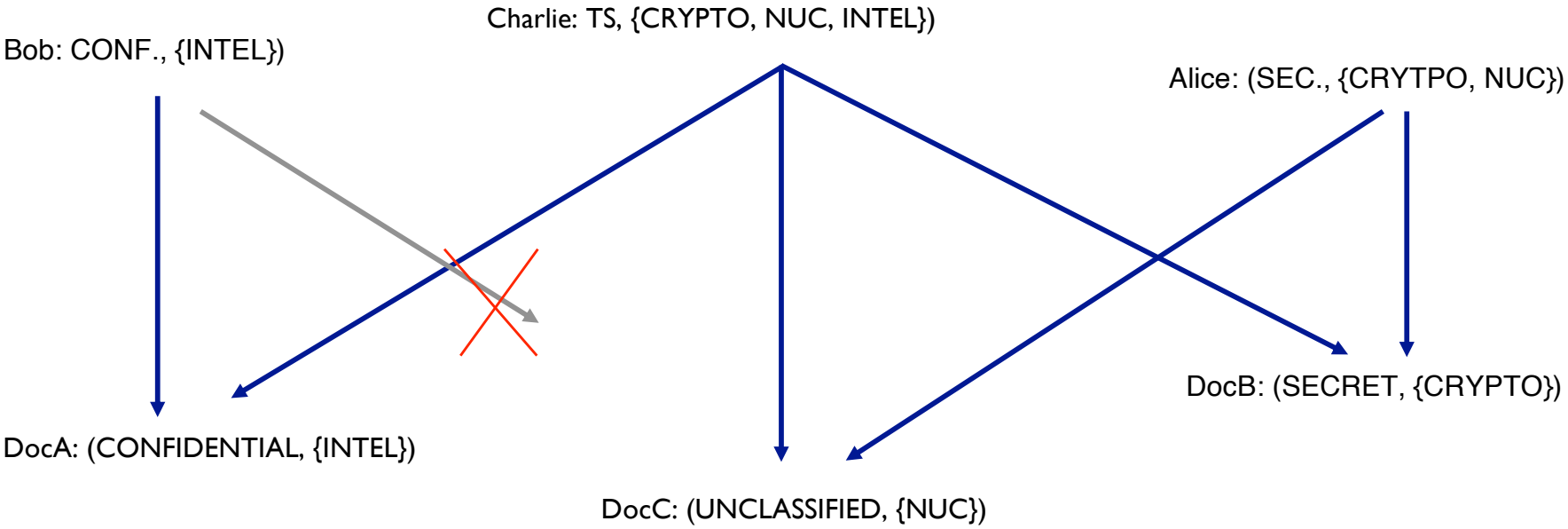
Assigning Security Levels

- All subjects are assigned clearance levels and compartments
  - Alice: (SECRET, {CRYTPO, NUC})
  - Bob: (CONFIDENTIAL, {INTEL})
  - Charlie: (TOP SECRET, {CRYPTO, NUC, INTEL})

- All objects are assigned an access class
  - DocA: (CONFIDENTIAL, {INTEL})
  - DocB: (SECRET, {CRYPTO})
  - DocC: (UNCLASSIFIED, {NUC})

Evaluating Policy

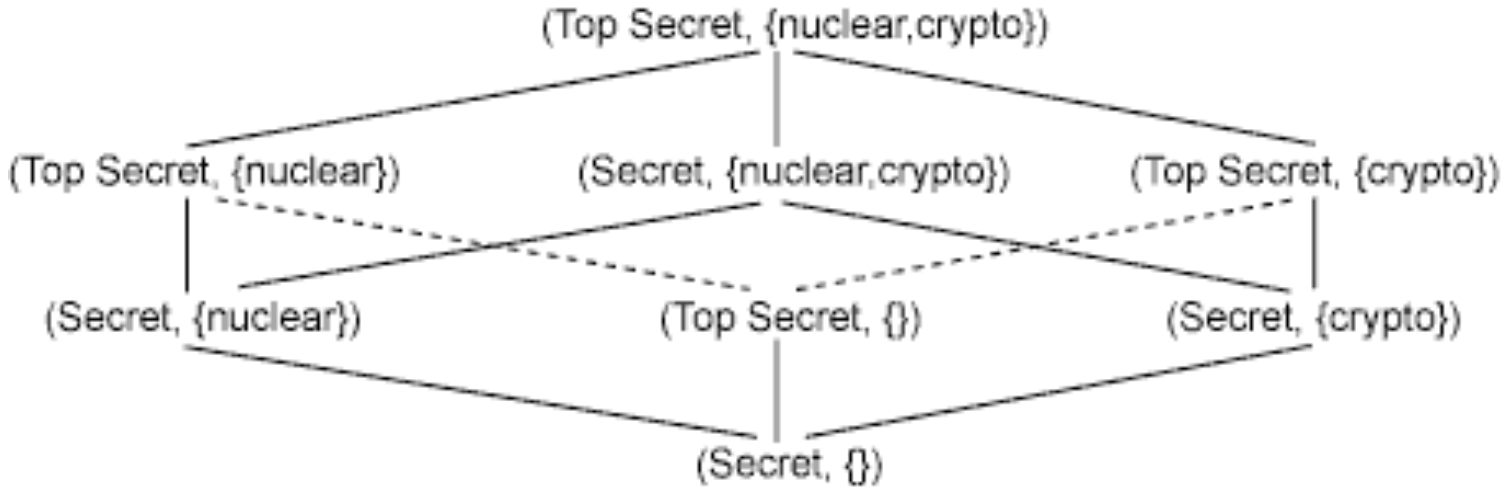
- Access is allowed if
- subject clearance level >= object sensitivity level *and* subject categories ⊇ object categories (*read down*)



Q: What would *write-up* be?

Bell-LaPadula (BLP) Model

- A Confidentiality MLS policy that enforces:
  - *Simple Security Policy*: a subject at specific classification level cannot read data with a higher classification level. This is short hand for "*no read up*".
  - * *(star) Property*: also known as the confinement property, states that subject at a specific classification cannot write data to a lower classification level. This is shorthand for "*no write down*".



How about integrity?

- MLS as presented before talks about who can "read" a document (confidentiality)
- Integrity considers who can "write" to a document
  - Thus, who can effect the integrity (content) of a document
  - Example: You may not care who can read DNS records, but you better care who writes to them!
- Biba defined a dual of secrecy for integrity
  - Lattice policy with, "no read down, no write up"
    - Users can only *create* content at or *below* their own integrity level (a monk may write a prayer book that can be read by commoners, but not one to be read by a high priest).
    - Users can only *view* content at or *above* their own integrity level (a monk may read a book written by the high priest, but may not read a pamphlet written by a lowly commoner).

Integrity, Sewage, and Wine

- Mix a gallon of sewage and one drop of wine gives you?
- Mix a gallon of wine and one drop of sewage gives you?

Integrity is really a contaminant problem: you want to make sure your data is not contaminated with data of lower integrity.

Biba (example)

- Which users can modify what documents?
  - Remember "*no read down, no write up*"

Charlie: (TS, {CRYPTO, NUC, INTEL})

Bob: (CONF., {INTEL})

Alice: (SEC., {CRYTPO, NUC})

?????

DocB: (SECRET, {CRYPTO})

DocA: (CONFIDENTIAL, {INTEL})

DocC: (UNCLASSIFIED, {NUC})

LOMAC

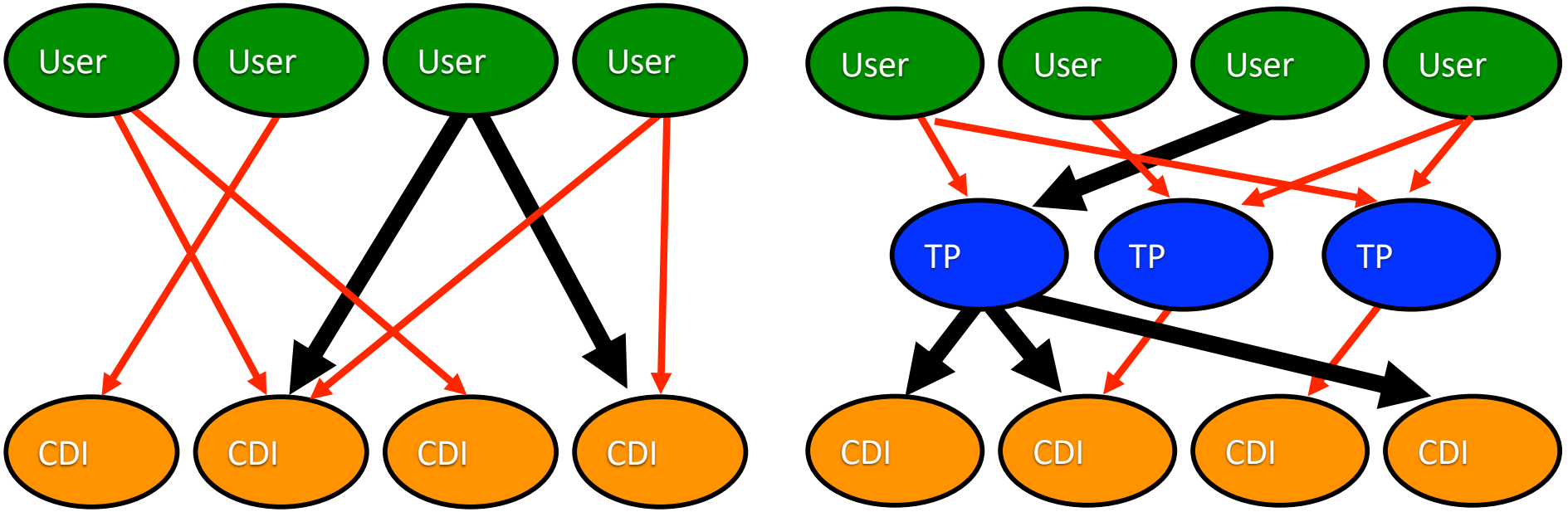
- Low-Water Mark integrity
  - Change integrity level based on actual dependencies
- Subject is initially at the highest integrity
  - But integrity level can change based on objects accessed
- Ultimately, subject has integrity of lowest object read
  - Example of "*self revocation*"

Clark-Wilson Integrity

- Map Integrity in Business (e.g., accounting) to Computing
- High Integrity Data (objects)
  - "Constrained Data Items" (CDIs)
- High Integrity Processes (programs)
  - "Transformation Procedures" (TPs)
- Check Integrity of Data Initially (verification)
  - "Integrity Verification Procedures" (IVPs)
- Premise
  - If the IVPs verify initial integrity
  - and high integrity data is only modified by TPs
  - Then, the integrity of computation is preserved

CW Permissions

- A user can access an CDI using TP iff
  1. The user has been granted CDI access
  2. The TP has been granted CDI access
  3. The user has been granted access to the TP



Clark-Wilson Issues

- Assure Function
  - Certify IVPs, TPs to be 'valid' (i.e., correct) (C1,C2)
  - Is there a general way of defining correctness?
- Handle Low Integrity Data
  - A TP must upgrade or discard any UDI (low integrity data) it receives (C5)

Reality: nice model, but too heavyweight in general for most applications. CW-lite (Jaeger) is an alternative that is tractable to implement.

CSCI680-04 QUIZ (6 points)

10/12/2017

1. Instead of an access control matrix, what are the ***two ways*** to store the protection state in a system? (1 point)

   Answer: Capability List (CL), Access Control List (ACL)

2. How is discretionary access control (DAC) different from mandatory access control (MAC)? (2 points)

   Answer:

   DAC: Owners can arbitrarily change the protection state of their objects (i.e., in a *discretionary* manner)

   MAC: The protection state and its transitions are *defined by an administrator*, i.e., cannot be changed at the discretion of the object owner.

3. Briefly state the ***two properties*** enforced by the Bell-LaPadula model for information secrecy (and describe in one short sentence each) (3 points)

   Answer:

   1. **Simple property: no read up** (i.e., the subject can't read an object at a higher level)
   2. ***- property: no write down** (i.e., a subject can't write to an object at a lower level)

   *Contrary to some answers, both these properties provide **secrecy** (i.e., enforce the flow of data from a lower secrecy level to a higher secrecy level only, and not the other way around).*

Midterm Next Tuesday (10/22)

- **Crib sheet:** *1 page, both sides, handwritten*

- **Get a calculator**
- **Room:** 002 McGlothlin-Street
- **8 am → 9:20 am (REACH CLASS BEFORE TIME)**
- **Includes** *every lecture:* Including the first one about reading papers.
- Use correct *cryptographic notation (slides)*

Good Luck!