Hybrid Policies

- Chinese Wall Model
  - Focuses on conflict of interest
  - Combines integrity and confidentiality
- ~~ORCON~~
  - Neither mandatory nor discretionary access control
- RBAC
  - Base controls on job function
- ~~CISS Policy~~

1

Chinese Wall Model

- Introduced by Brewer-Nash in 1989

Problem:

  - Consultant advises Bank1 and Bank2 about investments
  - Conflict of interest: his advice for either bank would affect his advice to the other bank
- Solution
  - Consultant can only access objects on his side of the wall
- Organization
  - Organize entities into "conflict of interest" classes
  - Control read accesses based on COI and access history
  - Control writing to all classes to ensure information is not passed along in violation of rules
  - No control over sanitized data (no conflict)

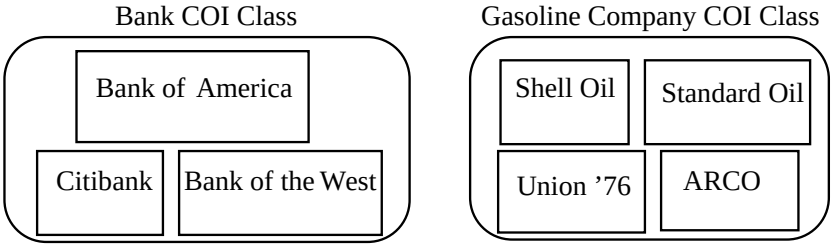
2

Definitions

- *Objects* : items of information related to a company
- *Company dataset* (CD): collection of objects related to a single company
  - Written *CD*(o)
- *Conflict of interest class* (COI): collection of datasets of companies in competition
  - Written *COI*(o)
- Assumption: each object belongs to exactly one *CD* and each CD to one *COI* class

3

Example



4

Temporal Element

- Rights depend on access history
- Initially, a subject can read any object in any CD of any COI
- If a subject reads an object in a CD in a COI, he can *never* read an object in another CD in the same COI
  - Possible that information learned earlier may allow him to make decisions later
- *PR*(*s*) denotes the set of objects that a subject *s* has already read

5

Sanitization

- Public information may belong to a CD
  - As is publicly available, no conflicts of interest arise
  - So, should not affect ability of subject to read
  - Typically, all sensitive data removed from such information before it is released publicly (called *sanitization*)

6

CW-Simple Security Condition

s can read *o* iff any of these conditions holds:

1. There is an o' such that $o' \in PR(s)$ and $CD(o') = CD(o)$
   - Meaning *s* has read something else in *o*'s dataset
2. For all $o' \in O$, $o' \in PR(s) \Rightarrow COI(o') \neq COI(o)$
   - Meaning *s* has not read any objects in $COI(o)$
3. *o* is a sanitized object

Initially, $PR(s) = \varnothing$, so any initial read request is granted

7

What about writing

- Alice and Bob work in same trading house
- Alice can read objects in Citibank's CD and in Shell's CD
- Bob can read objects in Bank of America's CD and in Shell's CD
- If Alice could write (information from Citibank's objects) to objects in Shell's CD, then Bob can read it
  - Hence, indirectly, he can read information from Citibank's CD, a clear conflict of interest

8

CW-*-Property

s can write to *o* if and only if :

1. The CW-simple security condition permits *s* to read *o*
   - No blind writes as in BLP

   and
2. For all *unsanitized* objects *o′*, if *s* can read *o′*, then *CD*(*o′*) = *CD*(*o*)
   - Says that s can write to an object if all the objects it can read are in the same dataset or sanitized

9

Compare to Bell-LaPadula

- Fundamentally different
  - ChW has no security labels, BLP does
  - ChW has notion of past accesses, BLP does not
- BLP can capture state at any time, but cannot track changes over time
  - Each (COI, CD) pair gets security category
  - Two clearances, *S* (sanitized) and *U* (unsanitized)
    - *U dom S*
  - Subjects assigned clearance for compartments that do not have categories corresponding to CDs in the same COI class

10

RBAC (http://csrc.nist.gov/rbac/)

- A policy-neutral model, that can express both DAC (role as identity) and MAC (role as clearance)
- Access/right often depends on role (job function), not on identity
  - Example:
    - Allison, bookkeeper, has access to financial records.
    - Bob hired to replace Allison as the new bookkeeper
    - Bob now has access automatically to those records
  - The role of "bookkeeper" determines access, not the identity of the individual, and 'connects' the subject to the permission(s).

11

Definitions

- Role *r* : collection of job functions
  - *trans*(*r*): set of authorized transactions for *r*
- Active role of subject *s* : the role *s* is currently in
  - *actr*(*s*)
- Authorized roles of *s* : set of roles *s* can assume
  - *authr*(*s*)
- *canexec*(*s*, *t*) is true if and only if subject *s* can execute transaction *t* at current time

12

Axioms (mandatory style)

S the set of subjects; T the set of transactions.

- *Rule of role assignment*:

$(\forall s \in S)(\forall t \in T)\ [canexec(s, t) \rightarrow actr(s) \neq \varnothing]$.

  - If s can execute a transaction, it has a role
  - This ties transactions to roles, not users

- *Rule of role authorization*:

$(\forall s \in S)\ [actr(s) \subseteq authr(s)]$.

  - Subject must be authorized to assume an active role (otherwise, any subject could assume any role)

13

Axiom

- *Rule of transaction authorization*:

$(\forall s \in S)(\forall t \in T)$
$[canexec(s, t) \rightarrow t \in trans(actr(s))]$.

  - A subject s can execute a transaction only if the transaction is authorized one for the role s has assumed (active)

14

Containment of Roles

- Trainer can do all the transactions that trainee can do (and then some). This means role r contains role r' ($r > r'$). So:

  $(\forall s \in S)[\ r' \in authr(s) \wedge r' > r \rightarrow r \in authr(s)\ ]$

  $(\forall t \in T)[\ t \in trans(r) \wedge r' > r \rightarrow t \in trans(r')\ ]$

- The set of roles is organized in a hierarchy (partial order)

15

Separation of Duty (static)

- For r a role, the predicate $meauth(r)$ (for *m*utually *e*xclusive *auth*orizations) is the set of roles that a subject s , for which $r \in auth(s)$, cannot assume because of some separation of duty requirement.
- Separation of duty constraint:

  $(\forall r_1, r_2 \in R)\ [\ r_2 \in meauth(r_1) \rightarrow$

  $[\ (\forall s \in S)\ [\ r_1 \in authr(s) \rightarrow r_2 \notin authr(s)\ ]\ ]\ ]$

16