Lecture 14

- Hybrid Models
  - DRM
  - Traducement
- Role-Based Access Control
- Composition of policies

DRM

- Goal is to protect information on a disk
- "Owner" is actually "licensee"
  - You don't own the content
  - Owner (copyright holder) can constrain what you can do with it

How Not to Do It

- User must install special program to play content
- Program also modified kernel to:
  - Prevent your CD copying software from working (by using a blacklist)
  - Monitors running applications always (even when no CD in drive)
  - Places hidden files on system
  - Allows you to make 3 copies using their software (and none with yours)
  - Weakens kernel so bad folks can exploit this (unintentional)

Accountability Model

- Traducement
  - Developed to model recording of real estate

Goals

- A signed document cannot be altered (but new signatures can be added)
- A document may require multiple signatures
- A document submitted to the recorder’s office may be revoked by any signatory until the document is recorded, but the document is no longer eligible for additional signatures
- The recorder may only append information to the document (ie, sign it)
- If the document is recorded, it becomes a public record immutable to all parties

Desirable Qualities

- Signed document is incompletely filled out
  - But it is signed, so it can't be completed!
  - Add: if anyone alters it, all signatures are revoked
- List of document authors
  - For accountability
- Dates of creation, recordation

Model Statement

- Legal documentation emphasizes:
  - Publication, which includes relinquishing the right to change the document further
  - Association of the authors of a document with the document

Types of Signatures

- Authors: people who contribute to the document to be filed
  - May even be only by signing it
- Recorder: attests to the completion and legal validity of the document
  - Converts it into an official record

Goals in Detail

1. The set of authors remains associated with a document throughout the document's lifetime. Any alteration of the document voids all existing signatures.
2. Subjects must be able to sign documents, and the act of signing must not invalidate existing signatures.
3. The signature of the recorder's office (in the capacity of recorder) publishes the document.

Types of Entities

- Objects (documents): o
  - Use $o(x)$ to represent attribute x of object o
- Users: u
  - Administrative authority: distinguished set of users with special rights
- Rules
  - These govern manipulation of data

Author Set

- Object attribute that specifies set of all users who have written to that object
  - No author can ever be removed from this set
  - All users in this set have "creative rights" over the object

Signer Set

- Set of users who have approved of the object and its contents
  - Any user who can read an object can add herself
  - Once in the set, only administrative user can remove user from the set

Creation Rule

- When u creates o, o is indelibly stamped with time of creation
  - Author set has creator: $o'(author\_set) = \{ u \}$
  - Signer set is empty: $o'(signer\_set) = \emptyset$
- Note: creation does not imply approval
  - That's why the creator isn't in the signer set

Alteration Rule

- When u alters o, u is added to the author set, and the signer set is cleared:
  - $o'(author\_set) = \{ u \} \cup o(author\_set)$
  - $o'(signer\_set) = \varnothing$
- User not added to signer set as alteration may be automatic, so u may not know how it is altered until review
  - That's why the existing members are deleted too

Signature Rule

- When u signs o, u is added to the signer set of o; the author set of o is unchanged
  - $o'(author\_set) = o(author\_set)$
  - $o'(signer\_set) = \{ u \} \cup o(signer\_set)$

Copy Rule

- When u creates a copy O of o, the author and signer sets of o are copied to be those of O
  - $O'(author\_set) = o(author\_set)$
  - $O'(signer\_set) = o(signer\_set)$

Example

- Peter drafts document *o*
  - $o(author\_set)$ = { Peter }
  - $o(signer\_set) = \varnothing$
- Paul (his lawyer) reviews and approves so he signs it
  - $o(author\_set)$ = { Peter }
  - $o(signer\_set)$ = { Paul }

Example (*con't*)

- Mary makes changes
  - $o(author\_set)$ = { Peter, Mary }
  - $o(signer\_set) = \emptyset$
- Kate copies *o*
  - $o(author\_set)$ = { Peter, Mary }
  - $o(signer\_set) = \emptyset$

Proposition

A user is in the *signer_set* of an object

iff

the document has not been modified since the user was added to the *signer_set*

Define Preconditions

1. Each document has a *signer_set* list identifying all users who created or modified that document
2. Each document has a *signer_set* list identifying all users who approve that document

Theorem

- If a system satisfied the two preconditions, then it satisfies the preconditions after any sequence of applications of the creation, alteration, signature, and copy rules

Theorem

Let R be a rule, s be a state of a system, and s' be the state obtained by applying R to s. Let the system in state s satisfy preconditions 1 and 2, and let O and O' be the set of objects in states s and s', respectively. Then the following hold:

Theorem (con't)

If there is an object o such that $o \notin O$, $o' \in O'$, $O = O \cup \{ o' \}$, $o(author\_set) = \{ u \}$ for some subject u, and $o(signer\_set) = \varnothing$, then s satisfies Preconditions 1 and 2.

If there is an object o such that $o'(author\_set) = \{ u \} \cup o(author\_set)$, and $o(signer\_set) = \varnothing$, then s' satisfies Preconditions 1 and 2.

Theorem (con't)

If there is an object o such that

$o'(author\_set) = o(author\_set)$

and

$o'(signer\_set) = \{ u \} \cup o(signer\_set)$

then s' satisfies Preconditions 1 and 2.

Theorem (con't)

If there is an object $x' \notin O$ but $x' \in O'$, and there is an object $o \in O$ such that

$x'(author\_set) = o(author\_set)$

and

$x'(signer\_set) = o(signer\_set)$

then s' satisfies Preconditions 1 and 2.

Problem: Naming

- Individuals from different counties may collaborate
  - Different recording offices may have different security policies
  - Collaborators must enforce most conservative elements of policies
- So authors' names may be ambiguous . . .

Domain Rule

- Authors contained in the author set shall be given unique names
  – Use some sort of scoping scheme to have names that reflect the administrative domain of the user

Authorship Integrity

An object is *recorded* when:

1. its author set is a subset of the signer set and
2. the recorder's office affixes the signature of the recorder to the object

Example

- Peter, Paul, Mary now sign the document
  - $o(author\_set)$ = { Peter, Mary }
  - $o(signer\_set)$ = { Peter, Paul, Mary }
- Recorder checks for completeness, then executes recordation transformation
  - Possible as $o(author\_set) \subseteq o(signer\_set)$

RBAC

- Access depends on function, not identity
  - Example:
    - Allison, bookkeeper for Math Dept, has access to financial records.
    - She leaves.
    - Betty hired as the new bookkeeper, so she now has access to those records
  - The role of "bookkeeper" dictates access, not the identity of the individual.

Definitions

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

Axioms

- Let S be the set of subjects and 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
- *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)

Axiom

- *Rule of transaction authorization*:

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

  – If a subject s can execute a transaction, then the transaction is an authorized one for the role s has assumed

Containment of Roles

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

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

Separation of Duty

- Let r be a role, and let s be a subject such that $r \in auth(s)$. Then the predicate $meauth(r)$ (for mutually exclusive authorizations) is the set of roles that s cannot assume because of the separation of duty requirement.
- Separation of duty:

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

Multiple Policies

- Problem
  - Policy composition
- Noninterference
  - HIGH inputs affect LOW outputs
- Nondeducibility
  - HIGH inputs can be determined from LOW outputs
- Restrictiveness
  - When can policies be composed successfully

Composition of Policies

- Two organizations have two security policies
- They merge
  - How do they combine security policies to create one security policy?
  - Can they create a coherent, consistent security policy?

The Problem

- Single system with 2 users
  - Each has own virtual machine
  - Holly at system high, Lara at system low so they cannot communicate directly
- CPU shared between VMs based on load
  - Forms a *covert channel* through which Holly, Lara can communicate

Example Protocol

- Holly, Lara agree:
  - Begin at noon
  - Lara will sample CPU utilization every minute
  - To send 1 bit, Holly runs program
    - Raises CPU utilization to over 60%
  - To send 0 bit, Holly does not run program
    - CPU utilization will be under 40%
- Not “writing” in traditional sense
  - But information flows from Holly to Lara

Policy vs. Mechanism

- Can be hard to separate these
- In the abstract: CPU forms channel along which information can be transmitted
  - Violates *-property
  - Not "writing" in traditional sense
- Conclusions:
  - Model does not give sufficient conditions to prevent communication, *or*
  - System is improperly abstracted; need a better definition of "writing"

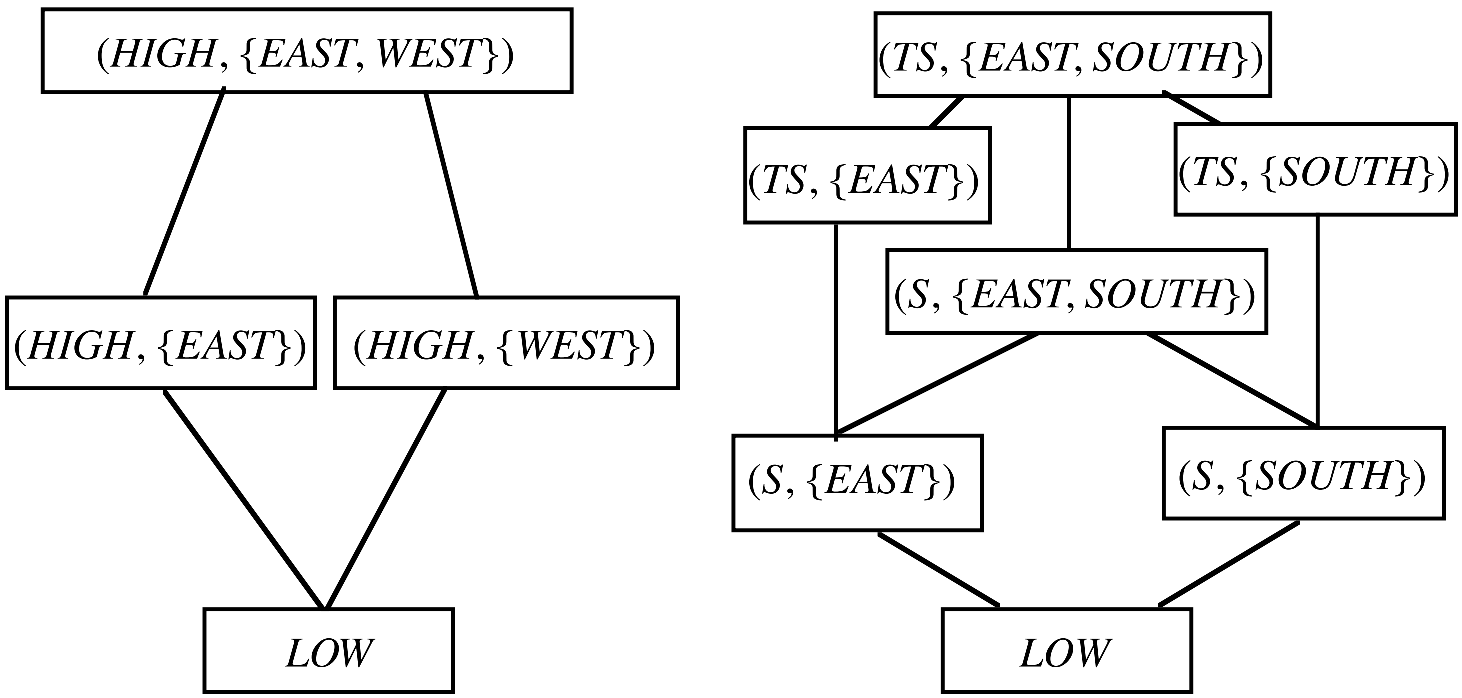
Composition of Bell-LaPadula

- Why?
  - Some standards require secure components to be connected to form secure (distributed, networked) system
- Question
  - Under what conditions is this secure?
- Assumptions
  - Implementation of systems precise with respect to each system's security policy

Issues

- Compose the lattices
- What is relationship among labels?
  - If the same, trivial
  - If different, new lattice must reflect the relationships among the levels

Example


(HIGH, {EAST, WEST})
(HIGH, {EAST})
(HIGH, {WEST})
LOW
(TS, {EAST, SOUTH})
(TS, {EAST})
(TS, {SOUTH})
(S, {EAST, SOUTH})
(S, {EAST})
(S, {SOUTH})
LOW

Analysis

- Assume S < HIGH < TS
- Assume SOUTH, EAST, WEST different
- Resulting lattice has:
  - 4 clearances (LOW < S < HIGH < TS)
  - 3 categories (SOUTH, EAST, WEST)