BLP, Access Control, and Covert Channels

Access Control Policy

MAC vs. DAC

- **Mandatory Access Controls (MAC):**
  - Rules enforced on **every** attempted access.
  - Not at the discretion of system users.
  - **Example:** BLP (Simple Security and *-Property must be satisfied).
- **Discretionary Access Controls (DAC):**
  - Rule enforcement may be waived/modified by users.
  - **Example:** Unix file permissions (owner can modify protections).

Access Control Matrix (ACM)

- Represents any access control policy.
- Matrix shows allowed accesses for each subject-object pair.

BLP Example:

- Subjects:
  - Subj1: (H, {A,B,C})
  - Subj2: (L, {})
  - Subj3: (L, {A,B,C})
- Objects:
  - Obj1: (L, {A,B,C})
  - Obj2: (L, {})
  - Obj3: (L, {B,C})

	Obj1	Obj2	Obj3
Subj1	R	R	R
Subj2	W	RW	W

Subj3	RW	R	R

Note: For large systems, ACM is implicit in rules (computed on-the-fly).

Lessons Learned

- BLP is an example of an **access control policy** (mandatory: rules are enforced in every attempted access).
- Any access control policy can be modeled as an explicit matrix.

Further Remarks on BLP

Lattice Structure

- BLP labels under the **dominates** relation form a **partial order**.
- This partial order can be represented as a **lattice**.

Example: Hierarchical levels {H, L} with H > L, categories {A, B}.

Information Flow in Lattice

- **Path from L_1 to L_2** → allowed information flow from L_1 to L_2 : a subject at level $L2$ can read a level $L1$ object, **OR** a subject at level $L1$ can write a level $L2$ object
- **No path from L_1 to L_2** → Simple Security prevents read **AND** *-Property prevents write.
- **BLP Goal:** Information should only flow **upward** in the lattice.
  - Equivalently: flow from L_1 to L_2 only if $L_2 \geq L_1$.

Metapolicy of BLP

- **Confidentiality:** Constrain information flow among security levels.
- **Question:** Can we satisfy BLP rules but violate metapolicy?
  - If yes, BLP rules are insufficient!

Lessons Learned

- BLP = Simple Security + *-Property + Tranquility.
- BLP labels form a lattice → **lattice-based security**.

- BLP Metapolicy: constrain information flow among levels.
- Metapolicy provides evaluation criteria for BLP rules.

Covert Channels

Definition

A **covert channel** is a path for **illegal flow of information** between subjects, utilizing system resources **not designed** for inter-subject communication.

Simple BLP System Example

Operations:

- `READ(S, O)` : If O exists and $L_s \geq L_o$, return value; else return 0.
- `WRITE(S, O, V)` : If O exists and $L_s \leq L_o$, change to V; else do nothing.
- `CREATE(S, O)` : If O doesn't exist, create at level L_s; else do nothing.
- `DESTROY(S, O)` : If O exists and $L_s \leq L_o$, destroy; else do nothing.

Covert Channel:

- High subject S^H can signal to low subject S^L by creating/destroying objects.
- S^L attempts to read object O:
  - If S^H created O → S^L reads value (1).
  - If S^H destroyed O → S^L reads 0 (0).
- **One bit transmitted** per attempt.

Importance

- Even one bit violates metapolicy.
- With coordination, arbitrary amounts of information can be transmitted over time.
- Information is **not in object contents**, but in **existence/state** of objects.

Types of Covert Channels

1. **Storage Channels:** Information recorded in system state.
   - Example: File existence, resource status.

2. **Timing Channels:** Information in ordering/duration of events.
    - Example: CPU scheduling, disk access ordering.
3. **Implicit Channels:** Information in program control flow.
    - Example: Conditional branches based on secret data.
4. **Other Types:** Termination, probability, resource exhaustion, power consumption.

Channel Characteristics

- **Existence:** Is channel present?
- **Bandwidth:** Information transmission rate (can be thousands of bits/sec).
- **Noiseless/Noisy:** Can information be transmitted without loss?
  It is usually infeasible for realistic systems to eliminate every potential covert channel.

Dealing with Covert Channels

1. **Eliminate:** Modify system implementation.
2. **Reduce Bandwidth:** Introduce noise.
3. **Monitor:** Intrusion detection for exploitation patterns.

Conditions for Covert Channels

Storage Channel:

- Sender/receiver access shared object attribute.
- Sender can modify attribute.
- Receiver can reference attribute.
- Mechanism for initiating/sequencing accesses.

Timing Channel:

- Sender/receiver access shared object attribute.
- Both have time reference (clock, timer).
- Sender controls timing of receiver's detection.
- Mechanism for initiating/sequencing accesses.

Detecting Covert Channels

Shared Resource Matrix Methodology (SRMM)

- **Richard Kemmerer's approach.**
- Table describing system commands and their effects on shared attributes.
- **R:** Operation References (provides info about) attribute.
- **M:** Operation Modifies attribute.
- **R and M in same row** → potential covert channel.

Example:

- `CREATE(S, O)` → R for "file existence" attribute.
- `DESTROY(S, O)` → M for "file existence" attribute.
- Row with both R and M indicates potential channel.

Using SRMM

1. Use BLP for standard information flows.
2. Use SRMM to identify covert channels.
3. Deal with channels: close, restrict, or monitor.

Note: SRMM requires deep knowledge of system semantics/implementation.

Non-Interference

Concept

- **Information Flow Policy:** Specifies which subjects can interfere with others.
- **Interfere:** "Do something that has an effect visible to."
- **Non-Interference (NI):** S^H actions should have **no visible effects** to S^L.

Specification

- Policy = reflexive binary relation (a → b) over subjects.
- **Example:** L → H (L can interfere with H, but not vice versa).

MLS to NI

- Any MLS policy → NI policy.
- $S_i \rightarrow S_j$ if level of S_j dominates level of S_i.
- **Not all NI policies → MLS policies** (NI policies need not be transitive).

Non-Transitive Policies Example

- Internet → Firewall → LAN
- No direct channel: Internet → LAN (explicitly unwanted).

Verifying NI

- For NI policy L → H:
  - Show: Any interleaving of L and H actions → L's view identical to L-only actions.
  - **Concept:** If L's view identical regardless of H's actions, policy holds.

Strengthening NI

- Enlarge L's view to include more system attributes:
  - Files → standard BLP.
  - System flags → prevent storage channels.
  - Clock → prevent timing channels.
  - Everything observable → complete isolation.

Challenges

- Real systems have many interferences (often benign).
- Proving NI for realistic systems is extremely difficult.

Integrity

Definition

- **Integrity:** Protecting information from unauthorized modification.
- **Key Questions:**
  - Who is authorized to modify data?

  - How to separate/protect assets?
  - Can erroneous changes be detected/corrected?
  - Can authorizations change over time?

Integrity vs. Confidentiality

- More context-dependent than confidentiality.
- Violations can occur without external interaction (e.g., incorrect computation).
- Often more important in commercial settings.

Integrity Labels

- **Analogous to BLP confidentiality labels:**
  - Hierarchical component: **trustworthiness level** (e.g., Novice, Student, Expert).
  - Categories: **domains of competence** (e.g., Physics, Finance).
- **Example:** Physics professor → (Expert: {Physics}).
- **Important:** Integrity labels **orthogonal** to confidentiality labels.

Integrity Metapolicy

- **Goal:** Prevent bad (low integrity) information from tainting good (high integrity) information.
- **Analogous to BLP but reversed:**
  - Don't allow information to **flow up** in integrity.
  - Subject shouldn't **write up** (low → high) or **read down** (high → low) in integrity.

Commercial Integrity Concerns (Steve Lipner)

- Users use existing production software (no custom programs).
- Programmers develop/test on nonproduction systems.
- Controlled/audited process for moving applications to production.
- Managers/auditors need access to system state/logs.

Integrity Principles

- **Separation of Duty:** Multiple subjects for critical functions.
- **Separation of Function:** Single subject cannot complete complementary roles.

- **Auditing:** Maintain audit trails for recoverability/accountability.

Lessons learned

- Integrity relates to how much we trust an entity to produce, protect, or modify data.
- Unlike confidentiality, violations of integrity don’t require external action.
- In some applications, particularly in the commercial world, integrity is more important than confidentiality.

Suppose we associate integritylabelswith subjects
and with objects in our system

- The label should reflect the trustworthiness of the subject, or
- the reliability of the information in the object.
  Important proviso: integrity labels are not also clearance labels.
- In a system that enforces both integrity and confidentiality, subjects/objects must have labels for each .
- e.g., a piece of information may be of dubious validity but very sensitive, or highly reliable and of little sensitivity.

The integrity metapolicy

- As with MLS, we want to define an access control policy that implements the security (integrity) goals of the system.
- But what are the rules?
- Recall with MLS, the BLP rules were really designed to constrain the flow of information within the system.
- We called that the metapolicy .
- So what is the metapolicy for integrity?
- Possible answer: Don't allow bad information to "taint" good information.
- Alternatively: don't allow information to "flow up" in integrity.

Summary of Key Points

1. **BLP** is a mandatory access control policy for confidentiality.
2. **Covert channels** exploit system resources for illegal information flow.
3. **Non-interference** policies specify allowed interference between subjects.
4. **Integrity** requires separate modeling from confidentiality.
5. **Risk management** is fundamental to realistic security.
6. **Policies** implement **metapolicies** (overarching security goals).