Metapolicy and Policy

Core Aspects of Security

1. **The "Big Three" (CIA Triad):**
    - **Confidentiality:** Preventing unauthorized disclosure of information (e.g., "Who can read this?").
    - **Integrity:** Preventing unauthorized modification of information (e.g., "Who can write or change this?"). Often more critical than confidentiality in commercial settings.
    - **Availability:** Ensuring information and system resources are accessible when needed (e.g., preventing Denial-of-Service attacks).
2. **Additional Key Aspects:**
    - **Authentication:** Verifying the identity of a user or system.
    - **Non-repudiation:** Preventing an entity from denying having performed an action.
3. **Context is King:** The relative importance of Confidentiality, Integrity, and Availability depends entirely on the system.
    - **Military System:** Confidentiality may be paramount.
    - **Banking System:** Integrity is most critical. (commercial)
    - **Online Retailer:** Availability is a matter of survival.

Security as Risk Management

Since perfect security is impossible, the goal is to manage risk.

- **Risk Definition:** The possibility that a **threat** will exploit a **vulnerability** to adversely impact an asset.
- **Risk Management Process:**
    i. Assess Assets, Threats, and Vulnerabilities.
    ii. Assess and Prioritize Risks.

iii. Make Risk Management Decisions.

- **Coping with Risk:**
  - **Acceptance:** Tolerate the risk.
  - **Avoidance:** Stop the activity that causes the risk.
  - **Mitigation:** Implement countermeasures to reduce the risk (most technical controls).
  - **Transfer:** Shift the risk to another party (e.g., insurance).
- **Annualized Loss Expectancy (ALE):** A common risk assessment tool (Loss Amount × Incidence Rate). It calculates the expected value of a risk but has limitations as it doesn't capture the full context of high-impact, low-probability events.
  - possible losses
  - their likelihood
  - potential cost for an average year.

    ALE = **Single Loss Expectancy (SLE)** × **Annualized Rate of Occurrence (ARO)**
- **SLE** = Cost to the business if the risk occurs once.
- **ARO** = Estimated number of times the loss is expected to occur in a year.
- **ALE** = Expected average loss per year, used to decide whether to implement safeguards.

Security Policy & Metapolicy

How do we define security?
• In the most general terms, security seems to mean something like
• protection of assets against threats.
• But the question is very specific to the context.

- **Security Policy:** A system-specific refinement of the metapolicy adequate to provide guidance to developers and users of the system.
- **Metapolicy:** The overall security goals of the system.
- **Relationship:** The policy implements the metapolicy. Without understanding the metapolicy, the policy rules can seem arbitrary and cannot be properly evaluated.
- If the metapolicy is what we care about, why bother with the policy at all?

- The metapolicy is often too general to provide adequate guidance.
- The metapolicy may be subject to multiple interpretations.
- There may be multiple acceptable policies that accomplish the security goals.
- The policy provides specific and enforceable guidelines to the system user/developer.

Policy Example: Multi-Level Security (MLS) & The Bell-LaPadula (BLP) Model

MLS is a classic policy for protecting confidentiality in environments like the military, where information and users have different sensitivity levels.

1. Labeling

- **Objects** (files, documents) are given a **sensitivity label**.
- **Subjects** (users, programs) are given a **clearance label**.
- **Label Structure:** `(Hierarchical-Level, Set-of-Categories)`
  - **Hierarchical Level:** A linear order (e.g., `Unclassified < Confidential < Secret < Top Secret`).
  - **Categories:** An unordered set for "need-to-know" (e.g., `{Nuclear, Crypto}`), enforcing the **Principle of Least Privilege**.

2. The "Dominates" Relation

A subject's clearance `(L_s, C_s)` **dominates** an object's sensitivity `(L_o, C_o)` if:

- `L_s >= L_o` (Subject's hierarchical level is higher or equal), **AND**
- `C_s ⊇ C_o` (Subject's categories are a superset of the object's).
- Lessons learned
  - For our MLS example, we partition information into containers and provide labels that reflect the sensitivity of the information.
  - The labels are structured, a hierarchical component and a set of need-to-know categories.
  - A folder with mixed information must be labeled to protect the information at the highest hierarchical level and protect all categories of information.

3. The Core BLP Security Properties

- **Simple Security Property (No-Read-Up):**
  A subject can **read** an object **only if** the subject's clearance **dominates** the object's sensitivity label.
- **The *-Property (No-Write-Down):**
  A subject can **write** to an object **only if** the object's sensitivity label **dominates** the subject's clearance. This prevents sensitive information from flowing to less-trusted levels.

The Simple Security Property ("no read up") ensures subjects cannot **access** information above their clearance. The *-Property ("no write down") ensures they cannot **leak** information below their level. Together, these two rules create a **one-way information flow** that only moves **upward** in the classification hierarchy—preventing any unauthorized disclosure downward. This combination achieves the meta-policy of **strict confidentiality** by making it impossible for information to flow from higher to lower levels, regardless of user intent.

In Bell–LaPadula (confidentiality model):

- **Simple security property** (“no read up”):
  A subject can read an object only if `level(subject) ≥ level(object)`.
- ***-property** (“no write down”):
  A subject can write to an object only if `level(subject) ≤ level(object)`.

These are the *access* rules. A **metapolicy** for confidentiality is a higher-level rule about *how labels themselves may change* so that you can’t bypass simple / * properties by relabeling.

Now consider changing an object’s classification label:

---

1. Upgrading an object (e.g., Secret → Top Secret)

- This **removes** some read permissions (fewer people can read it).
- It does **not** create any new flows from high to low.
- So, with respect to confidentiality, **upgrading is safe** and does *not* violate a reasonable metapolicy.
  Many systems explicitly allow only this direction of change for objects.

---

2. Downgrading an object (e.g., Top Secret → Secret or Unclassified)

This is where trouble starts.

Before downgrading:

- High subjects can write to the object (allowed by *-property, since they write to same or higher level).
- Low subjects **cannot read** it.

After downgrading:

- Low subjects **can now read** the object.
- But that object may already contain high-level information written when it was high.

So effectively you have:

> High → (write to high object) → [downgrade label] → low can read

That is an **indirect "write down"**, which violates the *intent* of the *-property and thus the **confidentiality metapolicy**.

For this reason:

- The Bell–LaPadula model usually assumes **tranquility**:
  - **Strong tranquility**: labels never change.
  - **Weak tranquility**: labels may change, but never in a way that violates the security policy (in practice: no unsafe downgrades).
- Real systems treat downgrading as **declassification**, done only by a **trusted subject** outside the normal policy (e.g., after manual review/redaction).

Summary

- **Upgrading** an object's level: generally safe for confidentiality, does not violate the metapolicy.
- **Downgrading** an object's level: *can* violate the confidentiality metapolicy (it's an implicit "write down"), and is either disallowed or only allowed via trusted declassification mechanisms.

4. Tranquility Properties

- **Strong Tranquility:** Labels never change. (Often too restrictive).
- **Weak Tranquility:** Labels cannot change in a way that violates the security policy's spirit (e.g., prevents unauthorized declassification).

Access Control Policies

- **BLP is an Access Control Policy:** It controls what actions (accesses) subjects can perform on objects.
- **Mandatory vs. Discretionary Access Control (MAC vs. DAC):**
  - **MAC (e.g., BLP):** Rules are enforced on every access; users cannot override them.
  - **DAC (e.g., Unix file permissions):** The owner of an object can discretionarily change its access rules.
- **Access Control Matrix (ACM):**
  - A theoretical model where a matrix explicitly lists the permissions for every subject-object pair. In practice, this matrix is too large, so rules like BLP's properties are used to compute permissions dynamically.
  - **ALL subjects & objects**