

Subject Name: Information System

Unit No:03

Unit Name: Security Policies

Faculty: Mrs. Bhavana Alte

Mr. Prathmesh Gunjgur

Unit No: 3

Unit Name: Security Policies Models

Introduction to Security Policies



What is a Security Policy?

- A **security policy** is a **set of rules and procedures** that define how an organization protects its systems, data, and users from cyber threats.
- **Why Are Security Policies Important?**
- **Prevent hacking and unauthorized access**
- **Protect sensitive data from leaks and modification**
- **Ensure compliance with security standards (ISO 27001, NIST, GDPR, HIPAA, PCI-DSS, etc.)**
- ✦ **Example:** A university security policy states that **only registered students** can access online class materials.



How Can a System Become Insecure?

- A system becomes insecure when:
- It allows **unauthorized access**.
- It **leaks sensitive information**.
- It **modifies data without permission**.
- ✦ **Example:** If an ATM allows a customer to **withdraw more money than available**, it is a **breach of security** due to **lack of integrity controls**.






Security Breaches and Violations

- **What is a Security Breach?**
- A security breach happens when a system **enters an unauthorized state**.
- ✦ **Example:**
- A student hacks into the university system and **modifies their grades**.
- A company employee **copies confidential files** and sells them.
- **How Do We Prevent Security Breaches?**
- ✓ **Authentication (Passwords, Biometrics, MFA)** – Ensures **only authorized users** can access the system.
- ✓ **Access Control (RBAC, MAC, DAC)** – Limits who can modify or view data.
- ✓ **Security Monitoring (Logging, Intrusion Detection Systems)** – Detects and stops breaches before they happen.



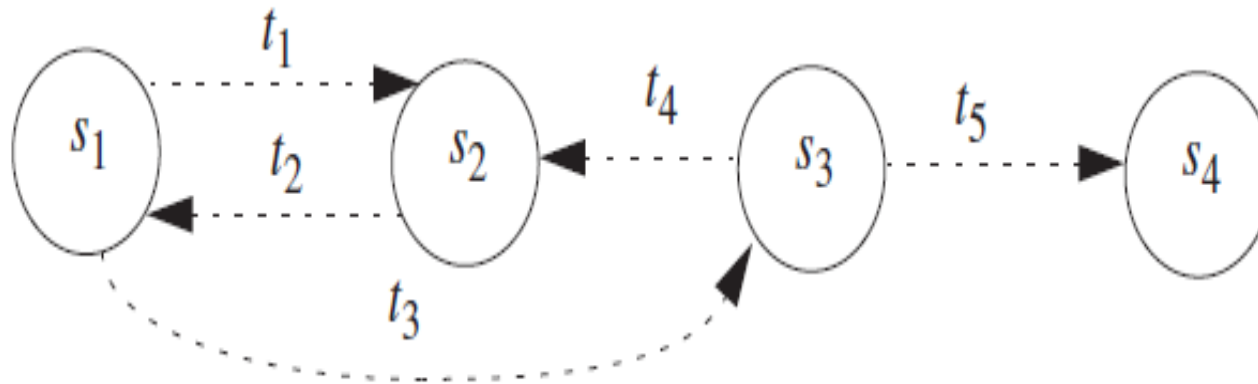
The CIA Triad – Three Key Security Properties

- Security policies aim to protect **three fundamental properties**:
- 1  **Confidentiality** – Prevents unauthorized access to information.
✦ **Example**: A hospital **encrypts** patient records so that only **authorized doctors** can access them.
- 2  **Integrity** – Ensures that data is not modified incorrectly.
✦ **Example**: A bank prevents a hacker from **changing account balances** to steal money.
- 3  **Availability** – Ensures that data and systems are always accessible.
✦ **Example**: An **emergency response system** must be **available 24/7** to save lives.
- **Discussion**: What happens if one of these fails?
- If **confidentiality fails**, sensitive data is leaked.
- If **integrity fails**, data becomes unreliable.
- If **availability fails**, critical services stop working



Security Policies and Finite-State Machines

- A **computer system** can be thought of as a **finite-state machine** where it transitions between **secure (authorized) states** and **insecure (unauthorized) states**.
- ✦ **Example:**
- ● **s1 (secure state)** → A user logs in with a password.
- ● **s2 (secure state)** → The user accesses their email.
- ● **s3 (insecure state)** → A hacker **guesses the password** and logs in.
- ● **s4 (insecure state)** → The hacker steals data.



Types of Security Policies

- **1. Confidentiality Policies**

- Control **who can access information**.

- ✦ **Example:**

- A government uses **classified levels (Top Secret, Secret, Confidential)** to protect national security data.

- **◆ 2. Integrity Policies**

- Ensure **data is not modified incorrectly**.

- ✦ **Example:**

- A **bank transaction system** ensures that **only authorized employees** can modify financial records.

- **◆ 3. Availability Policies**

- Ensure that data is **always accessible** when needed.

- ✦ **Example:**

- A **hospital system guarantees** that **patient records** are available **even during a cyberattack**.



Security Mechanisms

- A **security mechanism** is a tool or procedure that **enforces a security policy**.
- ✦ **Example:**
- **File permissions** prevent unauthorized users from modifying files.
- **Firewalls** block unauthorized network access.
- **Encryption** protects sensitive information from hackers.



Real-World Security Policy Examples

- **Homework Copying and Security Policy**
- A university has a **strict policy** that prohibits students from **copying another student's homework**. This policy is in place to prevent **cheating and academic dishonesty**.
- **The Situation**
- **Student A** works on their homework and **saves the file** on the university's computer system.
- **Student A forgets to set file permissions to restrict access** (meaning others can see and copy the file).
- **Student B** finds Student A's homework file and **copies it** without asking.
- **Student B submits the copied work as their own assignment.**
- **Conclusion: Student B violated the security policy**, even though **Student A left the file unprotected**.





The Role of Trust in Security Policies

- **Why is Trust Important?**
- Security policies, mechanisms, and procedures **all rely on trust**. If **trust is broken**, security **fails**.
- ✦ **Example:** A security patch is released for a company's operating system. If the IT administrator installs it, she is **trusting that**:
 - The patch **came from the official vendor** and was not modified by a hacker.
 - The patch **was tested correctly** and does not introduce new vulnerabilities.
 - The patch **will work on the company's system** without breaking existing software.
- 💡 **Key Concept:**
Trust is **assumed** in every security mechanism. If trust is misplaced, **security is compromised**.



Types of Access Control in Security Policies

- Security policies define **who can access what** using **access control models**.
- **1. Discretionary Access Control (DAC)**
-  **The owner of the data decides who gets access.**
 - ✦ **Example:**
 - A **Google Drive file owner** decides **who can view or edit the document**.
 - A **Windows user sets file permissions** manually.
 -  **Weakness:** If the owner **accidentally grants access to the wrong person**, security is compromised



2. Mandatory Access Control (MAC)

- **Access is controlled by the system, not the user.**
 - ✦ **Example:**
- **A military document marked "Top Secret" can only be accessed by people with the correct security clearance.**
- **Even if a user owns the document, they cannot change access permissions.**
- **◆ Strength: More secure** because users **cannot override policies.**
 - ◆ **Weakness: Less flexible** for everyday business use.



3. Originator Controlled Access (ORCON)

- What is ORCON?
- It is an **access control model** where the **creator of the information** (not the system owner) decides **who can access it** and **how it can be shared**.
- **Scenario: A Confidential Business Agreement**
- **Company A (TechCorp) hires Company B (DesignWorks) to create a new logo and branding strategy.**
- ✓ **TechCorp provides confidential design documents** to DesignWorks.
- ✓ The contract states that **DesignWorks cannot share these files** with anyone **without TechCorp's approval**.
- ✓ Even though **DesignWorks owns the file**, they **must get permission from TechCorp** before showing it to subcontractors.
- ✦ **What happens if DesignWorks wants to share the file with a freelancer?**
- ⦿ **They must ask TechCorp first.** If TechCorp **denies the request**, they **cannot share it**.



Security Policies

- 1. Confidentiality Policy – Keeping Information Private
- **Scenario: A Hospital's Patient Record System**
- A hospital has a **strict confidentiality policy** stating that **only doctors and authorized medical staff** can access patient records.
- Nurses can **see** a patient's prescription but **cannot modify it**.
- Receptionists can **only see appointment details** but **not medical history**.
- ✓ **Allowed:** A doctor accesses a patient's report to prescribe the right medicine.
- ⚡ **Not Allowed:** A hospital receptionist **tries to check** a patient's disease history.



-
- **2. Integrity Policy – Protecting Data from Unauthorized Changes**
 - **✦ Scenario: A Bank's Online Transaction System**
 - The bank ensures that **only customers and authorized staff** can modify account details.
 - Customers **can transfer money** but **cannot edit their past transactions**.
 - Bank managers can **approve or reject a loan request**, but a **normal employee cannot**.
 - **✓ Allowed:** A customer **transfers \$500** to another account.
⊘ Not Allowed: A hacker tries to **change a bank balance from \$500 to \$50,000**.



-
- **3. Availability Policy – Ensuring Services are Always Accessible**
 - ✦ **Scenario: Emergency Call Services (911)**
 - The **911 emergency system must always be available 24/7.**
 - The system has **backup power generators** in case of electricity failures.
 - There are **multiple servers in different locations** to handle **high call volumes.**
 - ✓ **Allowed:** A citizen calls **911 for an ambulance**, and the call **connects immediately.**
 - ⦿ **Not Allowed:** Due to a **server failure**, 911 **goes offline for 2 hours**, delaying emergency help.



4. Acceptable Use Policy – Rules for Using Company Devices & Internet

- **Scenario: A Company's Office Internet Policy**
- Employees **can use company internet** for **work-related activities**.
- Employees **CANNOT** download movies, play online games, or access **social media** during work hours.
- Websites like **Facebook and YouTube** are **blocked** on office Wi-Fi.
- ✓ **Allowed:** An employee **accesses email and cloud storage** for work.
- ⓧ **Not Allowed:** An employee **streams Netflix on office Wi-Fi** during work hours.



5. Password Policy – Ensuring Strong Authentication

✦ Scenario: A University's Online Exam System

- Students must **set passwords with at least 12 characters, including uppercase, lowercase, numbers, and symbols.**
- Passwords **must be changed every 3 months.**
- The system **locks an account after 5 failed login attempts** to prevent hacking.
- ✓ **Allowed:** A student **logs in using a strong password** like P@ssw0rd2024!.
- ✗ **Not Allowed:** A student **uses a weak password** like 12345678, which hackers can easily guess.



6. Data Retention Policy – Managing Old Data Securely

- ✦ **Scenario: A Company's Email Archiving Policy**
- The company **deletes emails older than 2 years** unless they are required for legal reasons.
- Employees **must back up important files** before the system auto-deletes old emails.
- Customer transaction records are **stored for 5 years** for auditing purposes.
- ✓ **Allowed:** The finance department keeps **tax records for 5 years** before deleting them.
 - ⊗ **Not Allowed:** An employee **keeps old emails forever**, increasing storage costs.



7. Bring Your Own Device (BYOD) Policy – Using Personal Devices for Work

- ✦ **Scenario: A Marketing Firm's Mobile Policy**
- Employees **can use their personal laptops and phones** for work.
- Personal devices **must have security software installed**.
- Employees **must report stolen devices immediately** to IT for remote data wipe.
- ✓ **Allowed:** An employee **logs into work email on their personal phone** with a secure VPN.
 - ⚡ **Not Allowed:** An employee **uses a personal laptop** without antivirus software, which gets infected with malware.



8. Incident Response Policy – Handling Security Breaches

- **Scenario: An E-Commerce Website Faces a Cyberattack**
- The **website is hacked**, and customer credit card data is at risk.
- The **IT security team immediately blocks unauthorized access**.
- The **company informs affected customers** and **enhances security measures**.
- ✓ **Allowed:** The IT team **follows the company's response plan** to fix security loopholes.
 - ⚡ **Not Allowed:** The company **ignores the attack** and **does not inform customers**, leading to lawsuits.



Unit No: 3

Unit Name: Security Policies Models

Confidentiality Policies



What is Confidentiality?

- **Confidentiality** means ensuring that sensitive data is only accessible to authorized users and is protected from unauthorized access.
- It prevents data leaks, unauthorized disclosure, and misuse of information.
- **Example:** A bank ensures that only account holders and authorized employees can view customer transactions.



Confidentiality Models

- Several models help enforce confidentiality in security systems.
- **A. Bell-LaPadula (BLP) Model**
- A widely used model in government and military security.
- **Core Principles:**
 - **No Read Up ("Simple Security Rule")** → A lower-level user **cannot** read a higher-level document.
 - **No Write Down ("Star (*) Property")** → A higher-level user **cannot** write to a lower-level document.
- **Example:**
- A government employee with a **Confidential** clearance **CANNOT** read **Top Secret** data.
- A **Top Secret** analyst **CANNOT** copy sensitive military data and paste it into a public document.



-
- **B. Mandatory Access Control (MAC)**
 - Access is assigned by **security labels** based on policies.
 - **Example:** A company's payroll data can only be accessed by the HR department, while other employees **cannot view or modify** their salaries.
 - **C. Discretionary Access Control (DAC)**
 - Users can control who accesses their data.
 - **Example:** In **Google Drive**, a user can share a document with **view-only or edit permissions**.



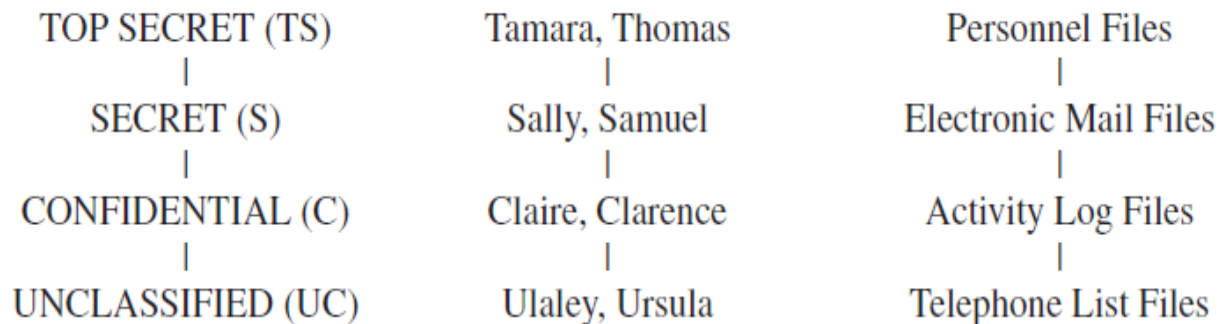
Bell-LaPadula Model (BLP) – A Security Model

- The **Bell-LaPadula (BLP) model** is a security model designed to enforce **confidentiality** in computer systems.
- It was developed in the 1970s for military and government applications where preventing unauthorized access to classified information is crucial.
- The model is built around **security levels** and **access control rules** that restrict how subjects (users or processes) interact with objects (files, data, or resources).
- Each **subject** (e.g., a person or process) and **object** (e.g., a document or database) is assigned a **security level** based on a classification hierarchy.
- A subject with a **higher clearance** (e.g., Top Secret) has access to more sensitive information than one with a lower clearance (e.g., Confidential).



Understanding Security Levels

- The levels are **hierarchical**, meaning:
- UC<C<S<TS subject with a **higher clearance** (e.g., Top Secret) has access to more sensitive information than one with a lower clearance (e.g., Confidential).



- **Figure** At the left is the basic confidentiality classification system.
- The four security levels are arranged with the most sensitive at the top and the least sensitive at the bottom.
- In the middle are individuals grouped by their security clearances, and at the right is a set of documents grouped by their security levels.



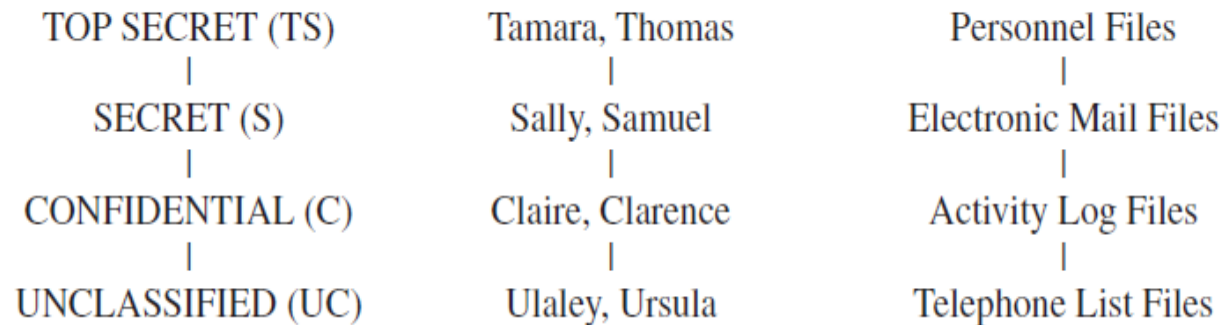
The Simple Security Condition ("No Read Up")

- The **Simple Security Condition** (often called "No Read Up" or **SS-property**) states:
- A subject **S** can read an object **O** only if:
- The **security classification** of **O** (denoted as $L(O)$) is **less than or equal** to the **security clearance** of **S** (denoted as $L(S)$).
- The subject has **discretionary read access** to the object.



Problem: Indirect Information Leakage

- A security loophole exists where a high-clearance subject (e.g., Tamara) **could copy sensitive information into a lower-security file**, allowing lower-clearance users (e.g., Claire) to access it.
- To prevent this, we introduce a second rule: **the -Property*.



The -Property ("*No Write Down*")

- A subject (user/process) **cannot write** to an object (file/document) **at a lower security level** than their own.
- **Formal Definition:**
- A subject **S** can write to an object **O** only if:
- The security level of **S** (denoted as **L(S)**) is **less than or equal to** the security level of **O** (i.e., **$L(S) \leq L(O)$**).
- The subject has discretionary **write access** to the object.

- **Why Is This Needed?**
- **Prevents Information Leaks** → If a **Top Secret (TS)** user writes classified data into an **Unclassified (UC)** document, then a **low-clearance** user could read it, violating confidentiality.
- **Stops Insider Threats** → A malicious employee with **Secret clearance** might try to copy data into a **Confidential** or **Unclassified** file so others can access it later.
- **Ensures Controlled Information Flow** → Data should only move **upward or within** the same level, **never downward**.



Example Scenario

- Imagine a military database with four security levels:
- **Top Secret (TS)**
- **Secret (S)**
- **Confidential (C)**
- **Unclassified (UC)**
- Now consider **Tamara** (TS clearance) and **Ulaley** (UC clearance):
- Tamara (TS) wants to **write** sensitive intelligence into a **UC file** so Ulaley (UC) can access it.
- The **Star Property** prevents this!
- Tamara can **only write to TS or higher-level files**, ensuring that **confidential data never flows down to a lower level**.



The Bell-LaPadula Model has two main rules:

- **Simple Security Rule ("No Read Up")** → Subjects **cannot read** objects **above** their clearance.
 - Example: Ulaley (UC) **cannot read** a **Top Secret** file.
- **Star Property ("No Write Down")** → Subjects **cannot write** to objects **below** their clearance.
 - Example: Tamara (TS) **cannot write** into a **UC** file.
- These two rules together **ensure confidentiality** by **blocking unauthorized access or leaks**.

TOP SECRET (TS)	Tamara, Thomas	Personnel Files
SECRET (S)	Sally, Samuel	Electronic Mail Files
CONFIDENTIAL (C)	Claire, Clarence	Activity Log Files
UNCLASSIFIED (UC)	Ulaley, Ursula	Telephone List Files



Example: The Data General B2 UNIX System

- The **Data General B2 UNIX (DG/UX)** system is a **secure UNIX-based operating system** that implements **Mandatory Access Controls (MACs)** based on the **Bell-LaPadula Model**.
- This system is designed to **control information flow** and **prevent unauthorized access or leaks**.
- **1. Assigning MAC Labels**
 - Every **process (subject)** and **object (file, directory, etc.)** in the system has a **MAC label**, which determines its security classification.
 - **Subjects (Users/Processes)** get MAC labels at **login time** from the **Authorization and Authentication (A&A) Database**.
 - **Objects (Files, Directories, etc.)** are assigned labels **at creation**, either:
 - **Explicitly** (stored as part of the object's attributes)
 - **Implicitly** (inherited from the parent directory)



- **Administrative Region**
 - The highest security level.
 - Stores **audit logs and system databases** that only administrators can access.
- **User Region**
 - Where normal users store their **files, applications, and data**.
 - Less restrictive but still controlled by security policies.
- **Virus Prevention Region**
 - Special security zone for **trusted system files**.
 - Prevents modification of critical files to stop malware attacks.

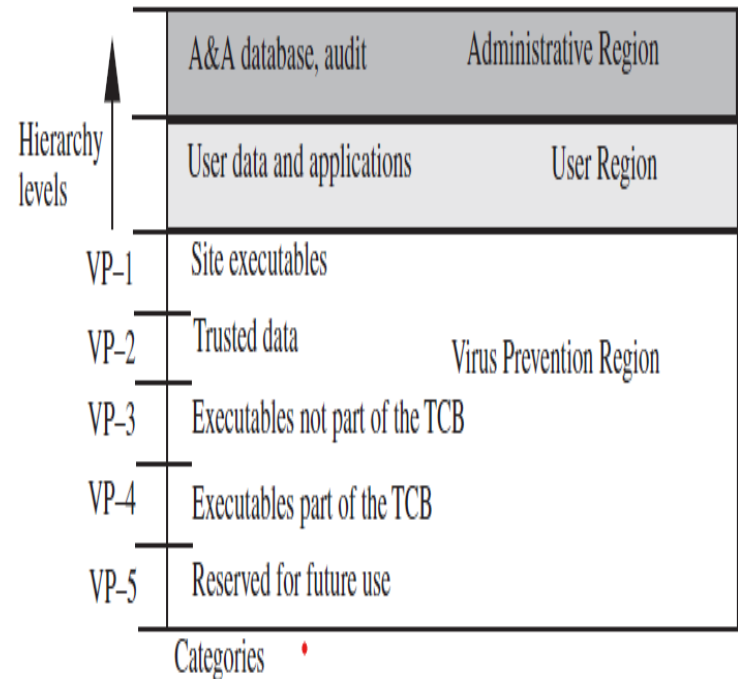


Figure 5-3 The three MAC regions in the MAC lattice (modified from the DG/UX Security Manual [230], p. 4-7, Figure 4-4). TCB stands for “trusted computing base.”

Security Levels (VP-1 to VP-5)

- Each **level (VP-1 to VP-5)** represents a different level of security

Level	Purpose
VP-1	Site executables (regular programs users can run).
VP-2	Trusted data (important files that shouldn't be changed easily).
VP-3	Executables not part of the Trusted Computing Base (TCB) (less critical programs).
VP-4	Trusted Computing Base (TCB) executables (core system programs that must be secure).
VP-5	Reserved for future security improvements.



MAC Lattice & Multilevel Directories in DG/UX

- In a **multilevel security (MLS) system** like DG/UX, each file and process has a **Mandatory Access Control (MAC) label**. This creates a challenge when different security levels interact in the same directory.
- **The Multilevel Security Problem**
- Let's say we have a **directory (/tmp)** where different users with different security levels create files.
- **MAC_A**: A user or process with MAC label **MAC_A** creates a file **/tmp/file.txt**.
- **MAC_B**: Another user or process with **MAC_B** (where **MAC_B** dominates **MAC_A**) tries to create **/tmp/file.txt**.
- **Problem: Name Conflict**
- Since the file already exists under **MAC_A**, creating another **file.txt** under **MAC_B** would **fail** due to name conflicts.



Solution: Multilevel Directories

- DG/UX **solves this** by introducing **hidden subdirectories**, one for **each MAC label**.
- **How It Works**
- Instead of storing all files in **one shared directory**, DG/UX **separates them into hidden subdirectories** based on MAC labels.
- When a **process with MAC_A** creates a file in **/tmp**, it actually goes into **/tmp/MAC_A/**.
- When a **process with MAC_B** creates a file in **/tmp**, it actually goes into **/tmp/MAC_B/**.
- **Each process only sees files within its security level.**
 - A process with **MAC_A** sees **/tmp/file.txt** from **/tmp/MAC_A/**.
 - A process with **MAC_B** sees **/tmp/file.txt** from **/tmp/MAC_B/**.



Example Scenario

Action

Process A (MAC_A) creates /tmp/a

Process B (MAC_B) creates /tmp/a

Process A lists /tmp

Process B lists /tmp

Both processes change directory to /tmp/a, then to .. (parent directory)

- Even though both processes think they are in **the same directory (/tmp/a)**, they are actually in **separate hidden directories based on their MAC labels**.
- This **prevents conflicts** while allowing **different security levels to coexist safely**.

What Actually Happens (Hidden Directory Mechanism)

The file is actually created in **/tmp/MAC_A/a**.

The file is actually created in **/tmp/MAC_B/a**.

It only sees **a** from **/tmp/MAC_A/**.

It only sees **a** from **/tmp/MAC_B/**.

Each process sees **their respective hidden directory (/tmp/MAC_A or /tmp/MAC_B)** as the parent.



Handling File System Labels

- DG/UX applies **Mandatory Access Control (MAC) labels** to files and directories to control access based on security levels.
- These labels **determine who can access, modify, or execute a file.**
- **Types of Labels**
 - **Explicit Labels** → **Manually assigned** to files or directories.
 - Example: A security admin assigns **MAC_High** to /secure/data.txt.
 - **Implicit Labels** → **Inherited** from the parent directory.
 - Example: If /home/user has **MAC_Medium**, any file created inside it will **automatically get** MAC_Medium unless explicitly changed.



Rules for Assigning Labels

1. Root of a File System (Mounting)

- When you mount a new file system (like a USB or network drive), it **must** have an **explicit MAC label** assigned.
- Example: If you mount a new storage /mnt/backup, the system assigns it **MAC_Low** or another label.

2. Implicit Labels (Inherited from Parent Directory)

New files take the label of their parent directory unless explicitly set.

/secure_data/ (MAC_High)

- └─ report.txt (inherits MAC_High)
- └─ logs/ (inherits MAC_High)
 - └─ error.log (inherits MAC_High)

All files and subdirectories inside /secure_data/ will **automatically** have MAC_High.



Rules for Assigning Labels

- 3. Hard Links → Forces Explicit Labels
 - **Hard links prevent files from inheriting labels** and instead **require an explicit label**.
 - In `/secure/data.txt` `/backup/copy.txt`
 - `/secure/data.txt` has **MAC_High**
 - `/backup/copy.txt` **must be explicitly labeled** instead of inheriting from `/backup/`.
 - 4. Changing Directory Labels → Forces Child Objects to Get Explicit Labels
 - If you **change** the label of a directory, all its existing files **must get explicit labels** instead of inheriting.
 - `/projects/` (original **MAC_Medium**)
 - |—— file1.txt (**MAC_Medium** - inherited)
 - |—— file2.txt (**MAC_Medium** - inherited)
- Now, if we change `/projects/` to **MAC_High**, then:
- file1.txt and file2.txt **must be explicitly assigned** **MAC_High**.
 - They **no longer inherit** from `/projects/`.



Rules for Assigning Labels

5. Symbolic Links → Use Target File's Label

- A **symbolic link (symlink)** points to another file.
- The **system checks the MAC label of the target file**, but **you still need access to the symlink itself**.

In -s /secure/data.txt /shared/link_to_data

- /secure/data.txt has **MAC_High**.
- If /shared/ has **MAC_Low**, a user with **MAC_Low** **cannot** access link_to_data because the target file (data.txt) is **MAC_High**.



Summary

Concept

Explicit Labels

Implicit Labels

Hard Links

Changing Directory Labels

Symbolic Links

Effect

Manually assigned security labels.

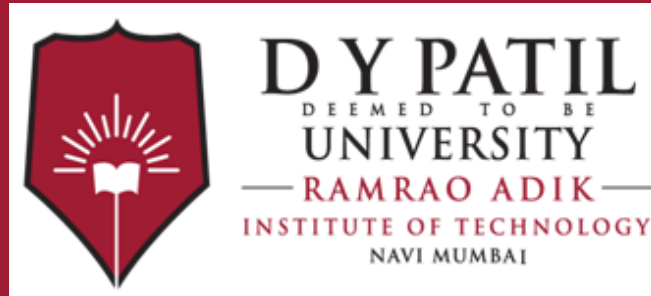
Files inherit security labels from their parent directory.

Files linked with hard links must have their own explicit labels.

Forces all files inside to get explicit labels.

The security label of the target file is checked, but access to the symlink itself also matters.





Thank You