Database Security EndSem 2024

Q1 (a)

We are given:

- **Doctors**
  - Read & write patient records
  - Read & write prescriptions
- **Nurses**
  - Read & write prescriptions only
  - Must learn *nothing* about patient records
- Must prevent **information flow from patient records → prescriptions**

This is an **information flow control** problem.

(i) Capturing the Policy in a Lattice Model

🔷 **Step 1: Define Security Levels**

Let:

- LH = Patient Records (Higher confidentiality)
- LL = Prescriptions (Lower confidentiality)

Define ordering:

$$LL \leq LH$$

Meaning:
Patient records are more sensitive than prescriptions.

🔷 **Step 2: Assign Labels to Objects**

- Patient records → LH
- Prescriptions → LL

- 🔷 **Step 3: Assign Labels to Subjects**
- Doctors → clearance LH
- Nurses → clearance LL

- 🔷 **Step 4: Enforce Bell–LaPadula Rules**
- Bell–LaPadula (confidentiality model):
- **No Read Up**
- **No Write Down**

🔷 Nurses

- Clearance = LLL_LLL
- Cannot read patient records (LHL_HLH)
- So they learn nothing about patient records

✓ Requirement satisfied

🔷 Doctors

- Clearance = LH
- Can read patient records (same level)
- Can write prescriptions?

Since prescriptions are LL
Doctor writing from LH to LL is:

Write Down → prohibited

Thus:
Information flow from patient records → prescriptions is prevented.

🔷 **Lattice Sketch (Text Form)**

```
L_H  (Patient Records, Doctors)
 ↑
 |
L_L  (Prescriptions, Nurses)
```

Higher cannot write to lower.
Lower cannot read higher.

(ii) Most Appropriate Security Model

The most appropriate model is:

✅ **Bell–LaPadula Model**

Why?

- This is a **confidentiality policy**
- Main requirement:
    - Prevent information flow from patient records → prescriptions
- Bell–LaPadula enforces:
    - No Read Up
    - No Write Down
- Exactly matches the requirement

Other models like Biba focus on integrity, not confidentiality.

(iii) Prevent Doctor Prescribing for Herself

Now add: A doctor must not write a prescription for herself.

This is NOT confidentiality.
This is a **conflict-of-interest / integrity rule**.

Bell–LaPadula alone cannot enforce this.

🔷 How to Augment the Model

Option 1: Use Clark–Wilson Model (Best Answer)

Clark–Wilson enforces:

- Well-formed transactions
- Separation of duty
- Integrity constraints

Add constraint:

$$\text{Doctor\_ID} \neq \text{Patient\_ID}$$

Prescription transaction must check that the doctor is not the patient.

Option 2: Add Separation-of-Duty Constraint

- Prevent subject from acting as:
    - Doctor role
    - Patient role
      simultaneously for same identity.

Final Exam-Ready Summary

(i) Define two security levels: PatientRecords (High) and Prescriptions (Low). Assign doctors high clearance and nurses low clearance. Use Bell–LaPadula model so nurses cannot read up and doctors cannot write down, preventing information flow from patient records to prescriptions.

(ii) Bell–LaPadula is most appropriate because it enforces confidentiality and prevents improper information flow.

(iii) To prevent doctors prescribing for themselves, augment the model using Clark–Wilson or a separation-of-duty constraint ensuring Doctor_ID ≠ Patient_ID in prescription transactions.

Q1 (b) (i)

Object: PrintServer

We treat the **print server as a single object**, and its methods are the possible operations:

Operations:

- print
- queue
- topQueue
- start
- stop
- restart
- status
- readConfig
- setConfig

Access Control List (ACL)

User	Allowed Operations
Alice (Manager)	print, queue, topQueue, start, stop, restart, status, readConfig, setConfig
Bob (Technician)	start, stop, restart, status, readConfig, setConfig
Cecilia (Power User)	print, queue, topQueue, restart
David	print, queue
Erica	print, queue
Fred	print, queue
George	print, queue

Explanation of Each User

🔷 Alice (Manager)

- Has **all operations**.
- Full administrative control.

🔷 Bob (Service Technician)

- Can:
  - start
  - stop
  - restart
  - status
  - readConfig
  - setConfig
- Cannot print or manage queue.

🔷 Cecilia (Power User)

- Can:
  - print
  - queue
  - topQueue
  - restart (when stuck)
- Cannot change configuration or stop/start directly.

🔷 David, Erica, Fred, George (Ordinary Users)

- Can only:
  - print
  - queue (view queue)
- No administrative privileges.

Final ACL Representation (Formal Form)

PrintServer:

Alice → {print, queue, topQueue, start, stop, restart, status, readConfig, setConfig}

Bob → {start, stop, restart, status, readConfig, setConfig}

Cecilia → {print, queue, topQueue, restart}

David → {print, queue}

Erica → {print, queue}

Fred → {print, queue}

George → {print, queue}

Q1) (ii) Identify Roles, Define Role Hierarchy and Permissions

We now convert the previous ACL into an **RBAC model**.

🔷 Step 1: Identify Roles

From the problem, the logical roles are:

1. **Manager**
2. **Technician**
3. **PowerUser**
4. **OrdinaryUser**

🔷 Step 2: Define Permissions for Each Role

Recall available operations:

- print
- queue
- topQueue
- start
- stop
- restart
- status
- readConfig
- setConfig

🔷 **OrdinaryUser**

- print
- queue

$$\text{OrdinaryUser} \rightarrow \{\text{print}, \text{queue}\}$$

🔷 **PowerUser**

Power users:

- Can print
- Manage queue (queue, topQueue)
- Restart server

$$\text{PowerUser} \rightarrow \{\text{print}, \text{queue}, \text{topQueue}, \text{restart}\}$$

Since PowerUser includes all OrdinaryUser capabilities plus more:

$$\text{PowerUser} \geq \text{OrdinaryUser}$$

🔷 **Technician**

Technician:

- start
- stop
- restart
- status

- readConfig
- setConfig

  Technician → {start, stop, restart, status, readConfig, setConfig}

Technician and PowerUser are NOT directly hierarchical because they have different permission sets.

🔷 **Manager**

Manager (Alice):

- Has all operations

Instead of redefining everything, define hierarchy:

$$\text{Manager} \geq \text{PowerUser}$$
$$\text{Manager} \geq \text{Technician}$$

Manager inherits permissions from both.

So:

$$\text{Manager} \rightarrow \text{All permissions}$$

🔷 **Step 3: Role Hierarchy Diagram**

```
        Manager
       /      \
  PowerUser   Technician
      |
 OrdinaryUser
```

🔷 **Step 4: Assign Users to Roles**

- Alice → Manager
- Bob → Technician
- Cecilia → PowerUser
- David → OrdinaryUser
- Erica → OrdinaryUser

- Fred → OrdinaryUser
- George → OrdinaryUser

Final Formal RBAC Definition

Roles:

R = {Manager, Technician, PowerUser, OrdinaryUser}

Permissions:

OrdinaryUser → {print, queue}

PowerUser → {topQueue, restart} + inherits OrdinaryUser

Technician → {start, stop, restart, status, readConfig, setConfig}

Manager → inherits PowerUser and Technician

Final Exam-Ready Summary

Define four roles: OrdinaryUser, PowerUser, Technician, and Manager. OrdinaryUser can print and view queue. PowerUser inherits OrdinaryUser and additionally has topQueue and restart. Technician has administrative operations (start, stop, restart, status, readConfig, setConfig). Manager inherits both PowerUser and Technician, thus having all permissions.

(iii) Role Reassignment After Staff Changes

Situation:

- Bob (Technician) leaves.
- George takes over as Technician.
- Two new employees:

  - **Henry** → Ordinary user
  - **Ida** → Power user (same role as Cecilia)

We are using the RBAC model defined in (ii).

🔷 Step 1 — Remove Bob

Since Bob leaves the company:

Revoke Bob from role Technician

Formally:

$$UA = UA - (\text{Bob}, \text{Technician})$$

(No need to delete role itself.)

🔷 Step 2 — Assign George as Technician

Remove him from OrdinaryUser and assign Technician:

$$UA = UA - (\text{George}, \text{OrdinaryUser})$$
$$UA = UA \cup (\text{George}, \text{Technician})$$

So George now inherits:

- start
- stop
- restart
- status
- readConfig
- setConfig

🔷 Step 3 — Assign Henry as Ordinary User

$$UA = UA \cup (\text{Henry}, \text{OrdinaryUser})$$

Henry gets:

- print
- queue

🔷 Step 4 — Assign Ida as PowerUser

$$UA = UA \cup (\text{Ida, PowerUser})$$

Since PowerUser ≥ OrdinaryUser, Ida gets:

- print
- queue
- topQueue
- restart

🔷 Important RBAC Insight

Notice:

✓ No permission changes were required
✓ Only user-role assignments changed
✓ Policy remains consistent

This is the advantage of RBAC — administrative changes are simple.

Final Exam-Ready Answer

When Bob leaves, his assignment to the Technician role is removed. George is removed from the OrdinaryUser role and assigned to the Technician role. Henry is assigned to the OrdinaryUser role. Ida is assigned to the PowerUser role. No changes to the role definitions or permissions are needed; only user-role assignments are updated.