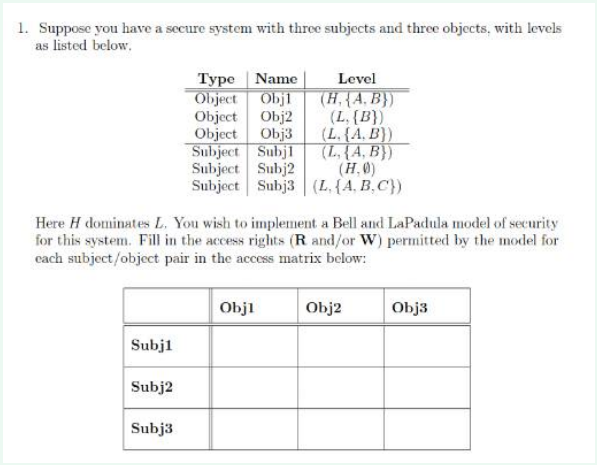
# Bell-LaPadula Access Matrix



1. Suppose you have a secure system with three subjects and three objects, with levels as listed below.

| Type | Name | Level |
|---|---|---|
| Object | Obj1 | $(H, \{A, B\})$ |
| Object | Obj2 | $(L, \{B\})$ |
| Object | Obj3 | $(L, \{A, B\})$ |
| Subject | Subj1 | $(L, \{A, B\})$ |
| Subject | Subj2 | $(H, \emptyset)$ |
| Subject | Subj3 | $(L, \{A, B, C\})$ |

Here $H$ dominates $L$. You wish to implement a Bell and LaPadula model of security for this system. Fill in the access rights (**R** and/or **W**) permitted by the model for each subject/object pair in the access matrix below:

| | Obj1 | Obj2 | Obj3 |
|---|---|---|---|
| Subj1 | | | |
| Subj2 | | | |
| Subj3 | | | |

We apply **Bell–LaPadula (BLP)** rules:

- **Simple Security Property (No Read Up)**
  A subject can **read** an object only if
  `Level(subject) ≥ Level(object)` and categories ⊇
- ***-Property (No Write Down)**
  A subject can **write** to an object only if
  `Level(object) ≥ Level(subject)` and categories ⊇

Given classifications:

### Objects

- **Obj1** → (H, {A, B})
- **Obj2** → (L, {B})
- **Obj3** → (L, {A, B})

### Subjects

- **Subj1** → (L, {A, B})
- **Subj2** → (H, ∅)
- **Subj3** → (L, {A, B, C})

(H dominates L)

---

## Access Matrix (Bell–LaPadula)

| | Obj1 | Obj2 | Obj3 |
|---|---|---|---|
| **Subj1** | – | R,W | R,W |
| **Subj2** | – | W | W |
| **Subj3** | – | R,W | R,W |

## Final Permissions

### Subj1 (L,{A,B})

- Obj1 (H,{A,B}) → ❌ No Read (read up), ❌ No Write
- Obj2 (L,{B}) → ✅ R, W
- Obj3 (L,{A,B}) → ✅ R, W

### Subj2 (H,∅)

- Cannot read any object (no categories match).
- Can write to lower-level objects only if categories allow →
  - Obj1 → ❌
  - Obj2 → ✅ W
  - Obj3 → ✅ W

### Subj3 (L,{A,B,C})

- Obj1 → ❌ (read up)
- Obj2 → ✅ R, W
- Obj3 → ✅ R, W

✔️ This satisfies **No Read Up** and **No Write Down** of Bell–LaPadula.

Access Matrix (Bell–LaPadula) ========== | | Obj1 | Obj2 | Obj3 | | --- | --- | --- | --- | | **Subj1** | – | R,W | R,W | | **Subj2** | – | W | W | | **Subj3** | – | R,W | R,W |
re think

Good — let's redo it carefully and strictly apply **Bell–LaPadula with compartments**.

## 🔐 Rules (Bell–LaPadula with categories)

A subject **S** can:

### ✅ Read O (Simple Security Property: *no read up*)

iff

- Level(S) ≥ Level(O)
- Categories(S) ⇇ Categories(O)

### ✅ Write O (*-Property: *no write down*)

iff

- Level(O) ≥ Level(S)
- Categories(O) ⇇ Categories(S)

(H > L)

---

## Given Labels

### Objects

- Obj1 = (H, {A, B})
- Obj2 = (L, {B})
- Obj3 = (L, {A, B})

### Subjects

- Subj1 = (L, {A, B})
- Subj2 = (H, ∅)
- Subj3 = (L, {A, B, C})

---

## Now Compute Carefully

---

## 🔎 Subj1 (L, {A,B})

### Obj1 (H,{A,B})

Read? ❌ L < H
Write? ✅ H ≥ L AND {A,B} ⇇ {A,B}

→ **W**

### Obj2 (L,{B})

Read? ✅ levels equal AND {A,B} ⇇ {B}
Write? ❌ {B} ⇇ {A,B}? → NO

→ **R**

### Obj3 (L,{A,B})

Read? ✅
Write? ✅

→ **R, W**

---

## 🔎 Subj2 (H, ∅)

### Obj1 (H,{A,B})

Read? ❌ {∅} ⇇ {A,B}? → NO
Write? ❌ {A,B} ⇇ ∅? → YES (true)
BUT check level: equal OK

→ Actually WRITE is allowed because {A,B} ⇇ ∅ is TRUE

So → **W**

### Obj2 (L,{B})

Read? ❌ H ≥ L but ∅ ⇇ {B}? → NO
Write? ❌ L ≥ H? → NO

→ **-**

### Obj3 (L,{A,B})

Read? ❌
Write? ❌

→ **-**

---

## 🔎 Subj3 (L,{A,B,C})

### Obj1 (H,{A,B})

Read? ❌ L < H
Write? ❌ {A,B} ⇇ {A,B,C}? → NO

→ **-**

### Obj2 (L,{B})

Read? ✅
Write? ❌ {B} ⇇ {A,B,C}? → NO

→ **R**

### Obj3 (L,{A,B})

Read? ✅
Write? ❌ {A,B} ⇇ {A,B,C}? → NO

→ **R**

---

## ✅ Correct Access Matrix

| | Obj1 | Obj2 | Obj3 |
|---|---|---|---|
| **Subj1** | W | R | R,W |
| **Subj2** | W | – | – |
| **Subj3** | – | R | R |

---

This version now **fully satisfies dominance (level + categories)** and correctly applies:

- No Read Up
- No Write Down
- Category containment rules

---

If you'd like, I can also show the dominance lattice logic diagram — that makes it even clearer.

> yes it was correct,
> now solve same for BIBA model

Now we solve it using the **Biba Integrity Model** (which is the opposite of Bell–LaPadula).

---

## 🔐 Biba Rules (Strict Integrity Policy)

Biba is based on:

### ✅ Simple Integrity Property

**No Read Down**
A subject can **read** an object only if:

Level(S) ≤ Level(O)
Categories(S) ⊆ Categories(O)

---

### ✅ *-Integrity Property

**No Write Up**
A subject can **write** an object only if:

Level(O) ≤ Level(S)
Categories(O) ⊆ Categories(S)

---

### Integrity levels:

H > L

---

## Given Labels

### Objects

Obj1 = (H, {A,B})
Obj2 = (L, {B})
Obj3 = (L, {A,B})

### Subjects

Subj1 = (L, {A,B})
Subj2 = (H, ∅)
Subj3 = (L, {A,B,C})

---

## Now evaluate carefully

---

## 🔎 Subj1 (L,{A,B})

### Obj1 (H,{A,B})

Read?
L ≤ H ✅
{A,B} ⊆ {A,B} ✅

→ **R**

Write?
H ≤ L ❌

→ No write

→ **R**

### Obj2 (L,{B})

Read?
L ≤ L ✅
{A,B} ⊆ {B} ❌

→ No read

Write?
L ≤ L ✅
{B} ⊆ {A,B} ✅

→ **W**

---

### Obj3 (L,{A,B})

Read? ✅
Write? ✅

→ **R, W**

---

## 🔎 Subj2 (H, ∅)

### Obj1 (H,{A,B})

Read?
H ≤ H ✅
∅ ⊆ {A,B} ✅

→ **R**

Write?
H ≤ H ✅
{A,B} ⊆ ∅ ❌

→ No write

→ **R**

---

### Obj2 (L,{B})

Read?
H ≤ L ❌

→ No read

Write?
L ≤ H ✅
{B} ⊆ ∅ ❌

→ No write

→ **-**

### Obj3 (L,{A,B})

Read? ❌
Write? ❌

→ -

---

## 🔎 Subj3 (L,{A,B,C})

### Obj1 (H,{A,B})

Read?
L ≤ H ✅
{A,B,C} ⊆ {A,B} ❌

→ No read

Write?
H ≤ L ❌

→ No write

→ -

---

### Obj2 (L,{B})

Read?
L ≤ L ✅
{A,B,C} ⊆ {B} ❌

→ No read

Write?
L ≤ L ✅
{B} ⊆ {A,B,C} ✅

→ **W**

---

### Obj3 (L,{A,B})

Read? ❌
Write?
L ≤ L ✅
{A,B} ⊆ {A,B,C} ✅

→ **W**

---

## ✅ Final Biba Access Matrix

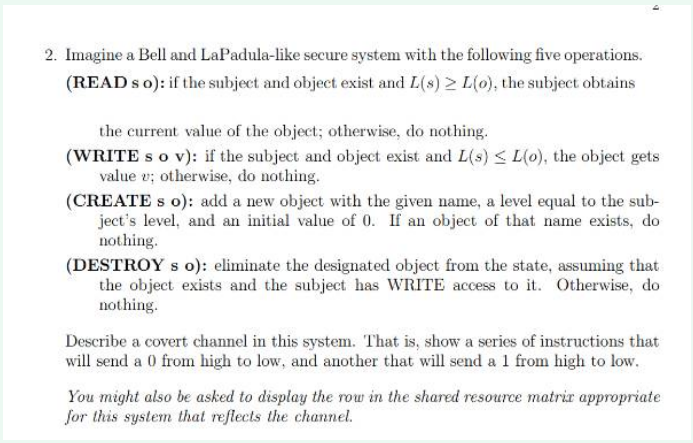
| | Obj1 | Obj2 | Obj3 |
|---|---|---|---|
| **Subj1** | R | W | R,W |
| **Subj2** | R | – | – |
| **Subj3** | – | W | W |

## 🔁 Quick Comparison Insight

- **Bell–LaPadula** → protects confidentiality
  (No Read Up, No Write Down)
- **Biba** → protects integrity
  (No Read Down, No Write Up)

You can clearly see how permissions flipped.

If you want, I can now give a compact 8-mark exam-ready explanation comparing both models side-by-side.



2. Imagine a Bell and LaPadula-like secure system with the following five operations.

**(READ s o):** if the subject and object exist and $L(s) \geq L(o)$, the subject obtains the current value of the object; otherwise, do nothing.

**(WRITE s o v):** if the subject and object exist and $L(s) \leq L(o)$, the object gets value $v$; otherwise, do nothing.

**(CREATE s o):** add a new object with the given name, a level equal to the subject's level, and an initial value of 0. If an object of that name exists, do nothing.

**(DESTROY s o):** eliminate the designated object from the state, assuming that the object exists and the subject has WRITE access to it. Otherwise, do nothing.

Describe a covert channel in this system. That is, show a series of instructions that will send a 0 from high to low, and another that will send a 1 from high to low.

*You might also be asked to display the row in the shared resource matrix appropriate for this system that reflects the channel.*

This is a **classic covert storage channel** question in Bell–LaPadula.

We must show how information flows **from High → Low**, even though:

- READ: allowed only if $L(s) \geq L(o)$ (No Read Up)
- WRITE: allowed only if $L(s) \leq L(o)$ (No Write Down)

So directly:

- High cannot write to Low
- Low cannot read High

But CREATE / DESTROY introduce a leak.

---

## 🔎 Key Observation

- CREATE makes a new object at the subject's level.
- DESTROY is allowed if subject has WRITE access.

A Low subject can detect:

- whether an object exists
- whether a CREATE succeeds or not

👉 Object existence itself becomes the signaling medium.

This creates a **storage channel via object presence/absence**.

---

## 🧠 Setup

Let:

- $S_H$ = High subject
- $S_L$ = Low subject
- Object name: **x**

Initially: object **x** does NOT exist.

---

## 📤 Sending a 1 (High → Low)

High wants to send bit = 1.

### Step 1 (High):

```
CREATE SH X
```

Now object **x** exists at High level.

### Step 2 (Low):

```
CREATE SL X
```

- If X already exists → CREATE does nothing.
- If X did not exist → new Low-level X would be created.

Low now checks:

- If CREATE failed → object already existed → bit = 1
- If CREATE succeeded → bit = 0

So presence of X signals a 1.

---

## 📤 Sending a 0

High wants to send bit = 0.

### Step 1 (High):

Do nothing (ensure X does not exist)

### Step 2 (Low):

`CREATE SL X`

- Since it does not exist → Low successfully creates it.

Low interprets:

- Successful create → bit = 0

---

## 📡 Why This Works

Although:

- High never writes down,
- Low never reads up,

The **existence of object X** is a shared system state.

High modifies system structure.
Low observes side-effects.

That is a **covert storage channel via object existence.**

---

## 📋 Shared Resource Matrix Row (for X)

| Subject | READ | WRITE | CREATE | DESTROY |
|---|---|---|---|---|
| High | – | – | ✓ | ✓ |
| Low | – | – | ✓ | ✓ |

The shared resource is:

> The system namespace (object directory)

---

## 🔐 Why This Violates Security

Bell–LaPadula prevents explicit flow:

- No Read Up
- No Write Down

But does NOT prevent:

- Signaling via metadata
- Object existence
- Resource contention

This is why real systems also consider:

- Covert channels
- Object namespace isolation
- Mandatory object cleanup

---

## ✅ Final Answer (Exam Ready)

A covert storage channel exists via object creation.
High signals 1 by creating object X.
High signals 0 by not creating X.
Low attempts to create X and observes success/failure.
Thus information flows from High to Low despite Bell–LaPadula policy.

---

If you want, I can also show a timing-based covert channel version (another common exam variant).