Cyber Security

Formal Models of Computer Security

Dr. Ramakrishna Dantu

Associate Professor, BITS Pilani

Disclaimer and Acknowledgement

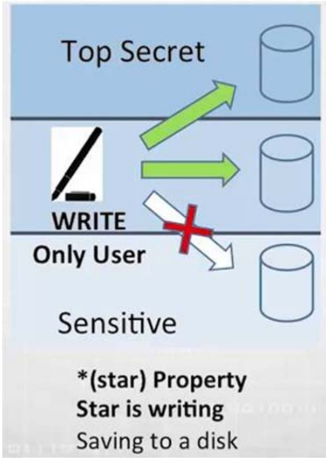
- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Agenda

- The CIA Classification:
  - Confidentiality Policies:
    - Bell-LaPadula Model
  - Integrity Policies:
    - The Biba Model
    - Lipner's Integrity Matrix Model
    - Clark-Wilson Integrity Model
    - Trust Models
  - Availability Policies:
    - Deadlock
    - Denial of Service Models

Bell LaPadula Model

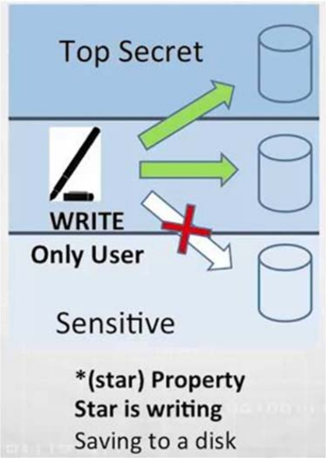
Maximum Security Level & Current Security Level



- At times, a subject must communicate with another subject at a lower level
- This requires the higher-level subject to write into a lower-level object that the lower-level subject can read
- Example:
  - A colonel with (SECRET, {NUC, EUR}) clearance needs to send a message to a major with (SECRET, {EUR}) clearance
  - The colonel must write a document that has at most the (SECRET, {EUR}) classification
  - But this violates the *-property, because (SECRET, {NUC, EUR}) *dom* (SECRET, {EUR})

Bell LaPadula Model

Maximum Security Level & Current Security Level



- The model provides a mechanism for allowing this type of communication
- A subject has a *maximum security level* and a current security level
- The maximum security level must dominate the current security level
- A subject may (effectively) decrease its security level from the maximum in order to communicate with entities at lower security levels
- Example
  - The colonel's maximum security level is (SECRET, {NUC, EUR})
  - She changes her current security level to (SECRET, {EUR})
  - This is valid, because the maximum security level dominates the current security level
  - She can then create the document at the major's clearance level and send it to him

Bell LaPadula Model

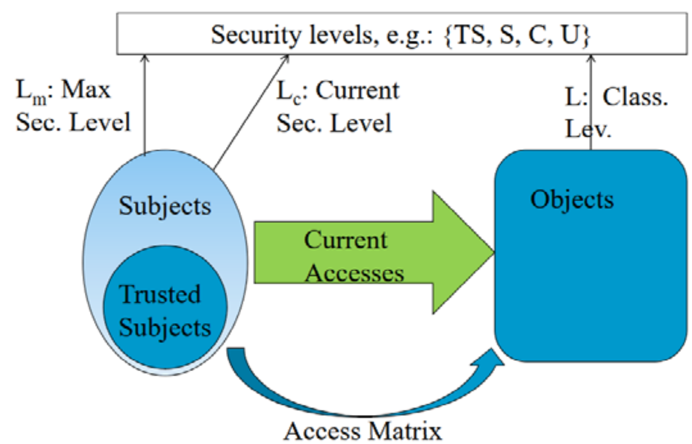
How can a system is considered secure?

- Use state-transition systems to describe computer systems
- A system as secure iff. every reachable state satisfies 3 properties
  - simple-security property
  - *-property
  - discretionary security property

Bell LaPadula Model

How can a system is considered secure?

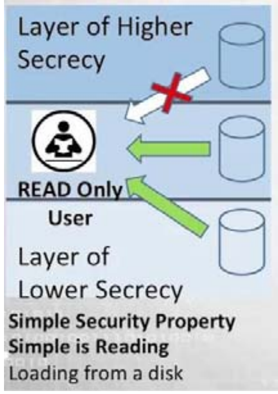
- A computer system is modeled as a state transition system
  - There is a set of subjects; some are designated as trusted
  - Each state has objects, an access matrix, and the current access information
  - There are state transition rules describing how a system can go from one state to another
  - Each subject s has a maximal sec level $L_m(s)$ and a current sec level $L_c(s)$
  - Each object has a classification level

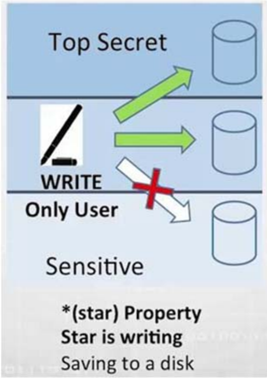


Bell LaPadula Model

How can a system is considered secure?

- A state is considered secure if it satisfies
  - Simple Security Condition (no read up):
    - S can read O iff $L_m(S) \geq L(O)$
  - The Star Property (no write down): for s that is not trusted
    - S can read O iff $L_c(S) \geq L(O)$
    - S can write O iff $L_c(S) \leq L(O)$
  - Discretionary-security property
    - every access is allowed by the access matrix
- A system is secure if and only if every reachable state is secure





Bell LaPadula Model

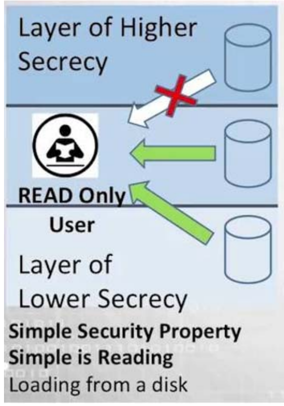
Is BLP Notion of Security Good?

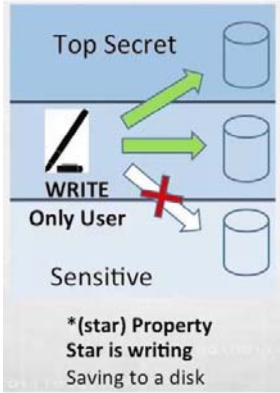
- The objective of BLP security is to ensure
  - a subject cleared at a low level should never read information classified high
  - The ss-property and the *-property are sufficient to stop such information flow at any given state
  - What about information flow across states?

Bell LaPadula Model

Is BLP Notion of Security Good?

- Consider a system with s_1,s_2,o_1,o_2
- And the following execution
  - s_1 gets access to o_1, read something, release access, then change current level to low, get write access to o2, write to o2
- Every state is secure, yet illegal information exists
- Solution:
  - Tranquility principle: subject cannot change current levels

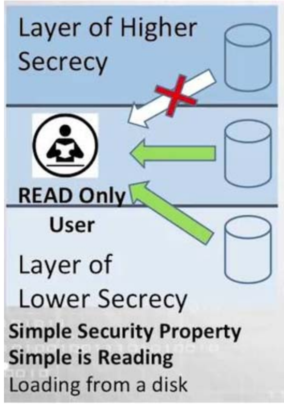


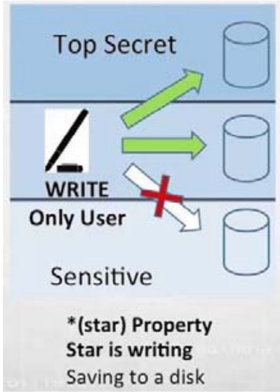


Bell LaPadula Model

Principle of Tranquility

- Strong Tranquility
  - The clearances of subjects, and the classifications of objects, do not change during the lifetime of the system
- Weak Tranquility
  - The clearances of subjects, and the classifications of objects, do not change in a way that violates
    - the simple security condition or the *-property during the lifetime of the system





Bell LaPadula Model
Formal Description

Bell LaPadula Model

Formal Description

- The state is described by the 4-tuple (b, M, f, H), where
- b = Current access set
  - This is a set of triples of the form (subject, object, access mode)
  - A triple (s, o, a) means that
    - subject "s" has current access to "o" in access mode "a"
  - Note that this does not simply mean that s has the access right a to o
  - The triple means that s is currently exercising that access right
    - that is s is currently accessing o by mode a

Bell LaPadula Model

Formal Description

- M = Access matrix
  - The matrix element M_{ij} records the access modes in which subject S_i is permitted to access object O_j

SUBJECTS	OBJECTS →	File 1	File 2	File 3	File 4
	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

(a) Access matrix

Bell LaPadula Model

Formal Description

- f = Security Level function
  - This function assigns a security level to each subject and object
  - It consists of three mappings:
    - $f_o(O_j)$ is the classification level of object O_j
    - $f_s(S_i)$ is the security clearance of subject S_i
    - $f_c(S_i)$ is the current security level of subject S_i
  - The security clearance of a subject is the maximum security level of the subject
  - The subject may operate at this level or at a lower level
  - Thus, a user may log onto the system at a level lower than the user's security clearance
  - This is particularly useful in a role-based access control system.
- H = Hierarchy
  - This is a directed rooted tree whose nodes correspond to objects in the system
  - The model requires that the security level of an object must be greater than or equal to its parent

Bell LaPadula Model

Formal Description

- For every subject S_i and every object O_j, the requirements can be stated as follows:
- ss-property:
  - Every triple of the form (S_i, O_j, read) in the current access set b has the property $f_c(S_i) \geq f_o(O_j)$
- *-property:
  - Every triple of the form (S_i, O_j, append) in the current access set b has the property $f_c(S_i) \leq f_o(O_j)$
  - Every triple of the form (S_i, O_j, write) in the current access set b has the property $f_c(S_i) = f_o(O_j)$
- ds-property:
  - If (S_i, O_j, A_x) is a current access (is in b), then access mode A_x is recorded in the (S_i, O_j) element of M
  - That is, (S_i, O_j, A_x) implies that $A_x \in M[S_i,O_j]$

Bell LaPadula Model

Formal Description

- A secure system is characterized by the following:
  1) The current security state of the system (b, M, f, H) is secure if and only if every element of b satisfies the three properties
  2) The security state of the system is changed by any operation that causes a change any of the four components of the system, (b, M, f, H)
  3) A secure system remains secure so long as any state change does not violate the three properties

Bell LaPadula Model

Abstract Operations

- The BLP model includes a set of rules based on abstract operations that change the state of the system. The rules are as follows:
- **Get access:**
  - Add a triple (subject, object, access-mode) to the current access set b
  - Used by a subject to initiate access to an object in the requested mode
- **Release access:**
  - Remove a triple (subject, object, access-mode) from the current access set b
  - Used to release previously initiated access
- **Change object level:**
  - Change the value of $f_o(O_j)$ for some object O_j
  - Used by a subject to alter the security level of an object

Bell LaPadula Model

Abstract Operations

- Change current level:
  - Change the value of $f_c(S_i)$ for some subject S_i
  - Used by a subject to alter the security level of a subject
- Give access permission:
  - Add an access mode to some entry of the access permission matrix M
  - Used by a subject to grant an access mode on a specified object to another subject
- Rescind access permission:
  - Delete an access mode from some entry of M
  - Used by a subject to revoke an access previously granted.

		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

(a) Access matrix

Bell LaPadula Model

Abstract Operations

- Create an object:
  - Attach an object to the current tree structure H as a leaf
  - Used to create a new object or activate an object that has previously been defined but is inactive because it has not been inserted into H
- Delete a group of objects:
  - Detach from H an object and all other objects beneath it in the hierarchy
  - This renders the group of objects inactive
  - This operation may also modify the current access set b because all accesses to the object are released

Bell LaPadula Model

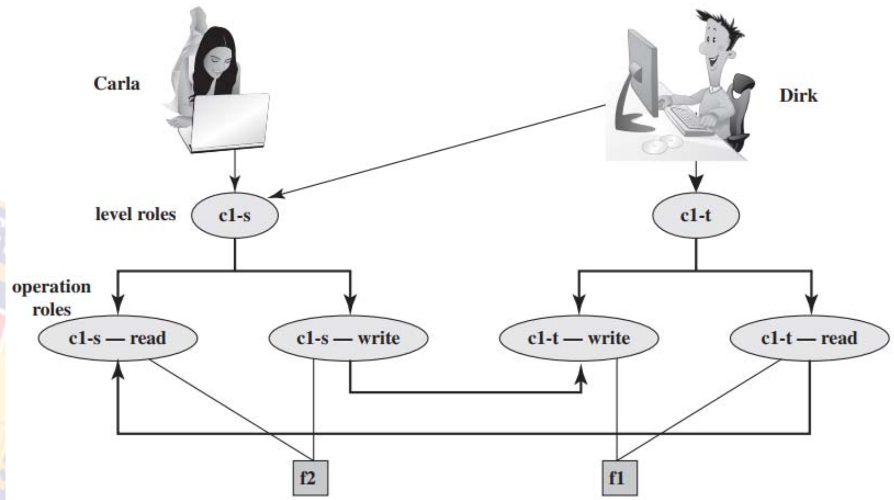
Example of BLP Use

- Carla is a student (s) in course c1
- Dirk is a teacher (t) in course c1 but may also access the system as a student; thus two roles are assigned to Dirk:
  - Carla: (c1-s)
  - Dirk: (c1-t), (c1-s)
- The student role is assigned a lower security clearance and the teacher role a higher security clearance
- Let us look at some possible actions:

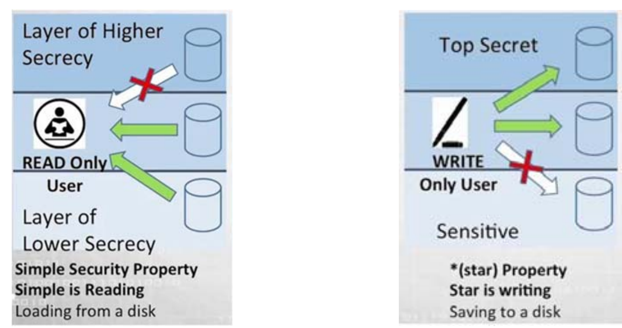
Bell LaPadula Model

Example of BLP Use – Step 1

- Dirk creates a new file f1 as c1-t
- Carla creates file f2 as c1-s
- Carla (Student):
  - Can read and write to f2
  - Cannot read f1, because it is at a higher classification level (teacher level)
- Dirk (Teacher):
  - Can read and write f1
  - Can read f2 if Carla grants access to f2



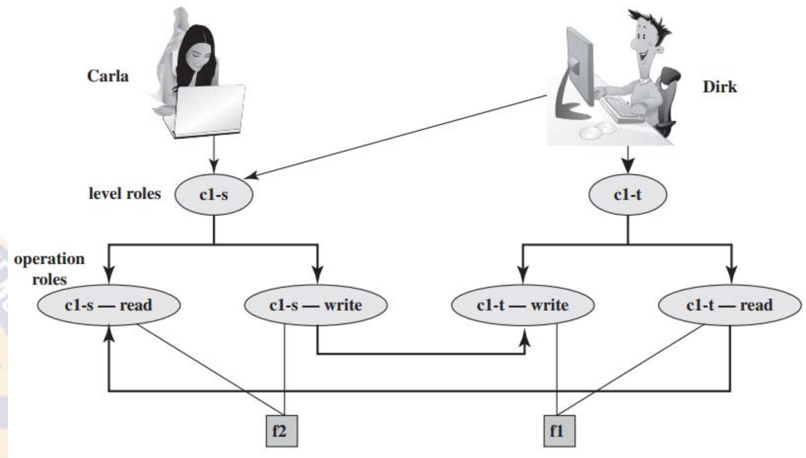
(a) Two new files are created: f1: c1-t; f2: c1-s



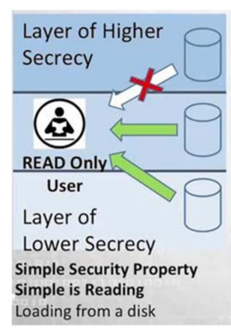
Bell LaPadula Model

Example of BLP Use – Step 1

- However, Dirk as a teacher cannot write f2 because of the *-property
- Neither Dirk nor a Trojan horse on his behalf can downgrade data from the teacher level to the student level
- Only if Dirk logs in as a student can he create a c1-s file or write to an existing c1-s file, such as f2
- In the student role, Dirk can also read f2



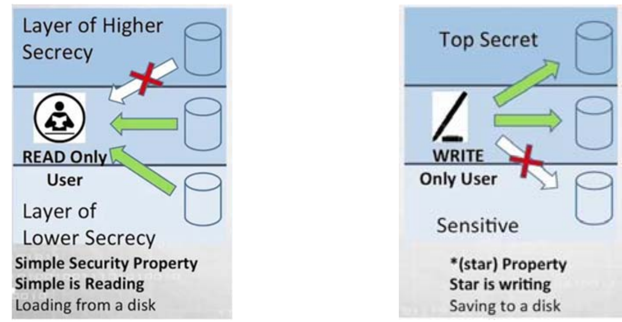
(a) Two new files are created: f1: c1-t; f2: c1-s

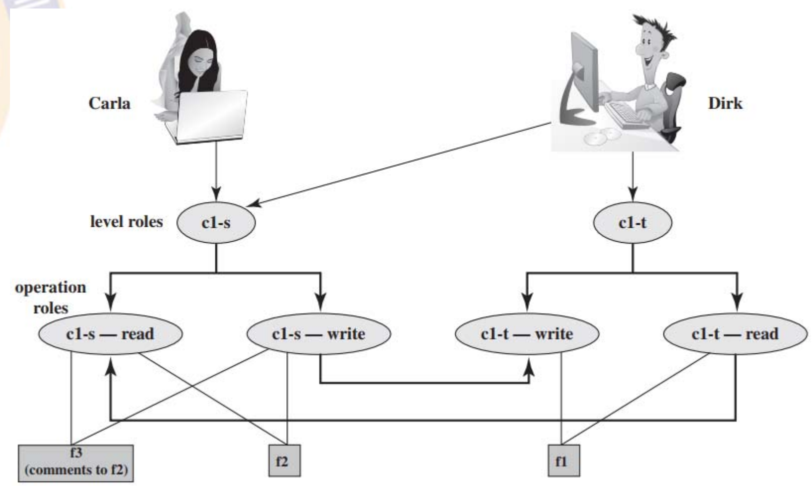


Bell LaPadula Model

Example of BLP Use – Step 2

- Dirk reads f2 and wants to create a new file with comments to Carla as feedback
- Dirk must sign in student role c1-s to create f3 so that it can be accessed by Carla
- In a teacher role, Dirk cannot create a file at a student classification level



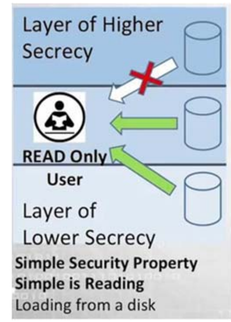


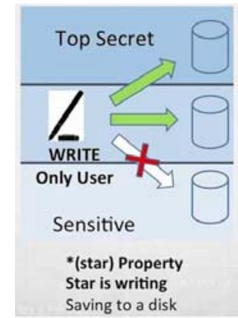
(b) A third file is added: f3: c1-s

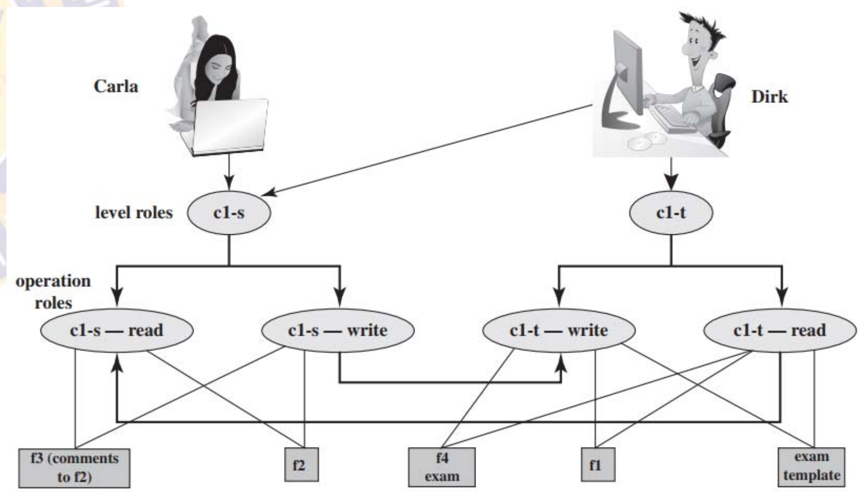
Bell LaPadula Model

Example of BLP Use – Step 3

- Dirk creates an exam based on an existing template file store at level c1-t
- Dirk must log in as c1-t to read the template and the file he creates (f4) must also be at the teacher level
- Dirk wants Carla to take the exam and so must provide her with read access
- However, such access would violate the ss-property





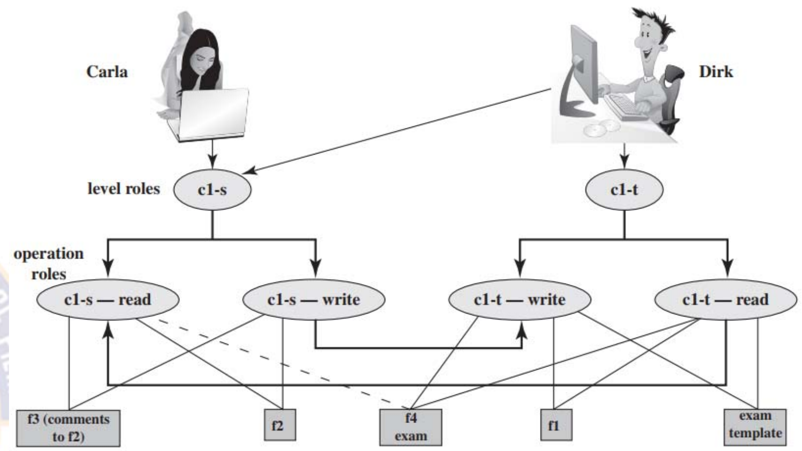


(c) An exam is created based on an existing template: f4: c1-t

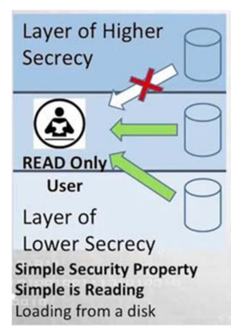
Bell LaPadula Model

Example of BLP Use – Step 4

- Dirk must downgrade the classification of f4 from c1-t to c1-s
- Dirk cannot do this in the c1-t role because this would violate the *-property
- Therefore, a security administrator must have downgrade authority and must be able to perform the downgrade outside the BLP model
- The dotted line connecting f4 with c1-s-read indicates that this connection has not been generated by the default BLP rules but by a system operation



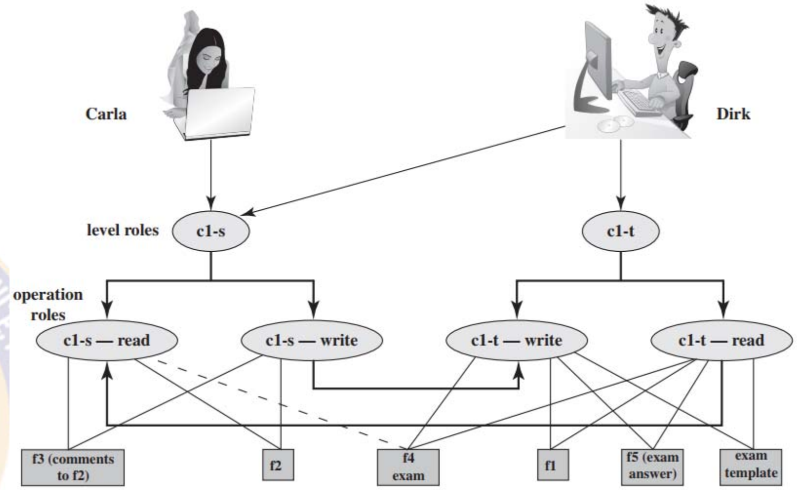
(d) Carla, as student, is permitted acess to the exam: f4: c1-s



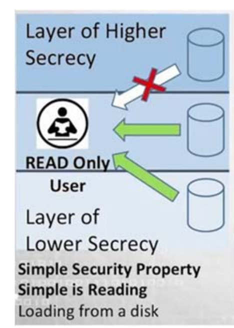
Bell LaPadula Model

Example of BLP Use – Step 5

- Carla writes the answers to the exam into a file f5
- She creates the file at level c1-t so that only Dirk can read the file
- This is an example of writing up, which is not forbidden by the BLP rules
- Carla can still see her answers at her workstation but cannot access f5 for reading



(e) The answers given by Carla are only accessible for the teacher: f5: c1-t



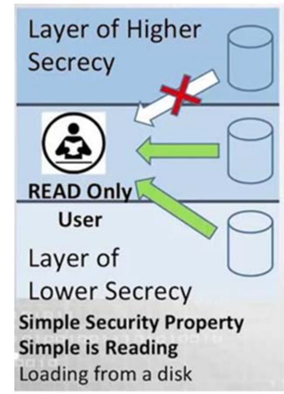
Limitations of BLP Model

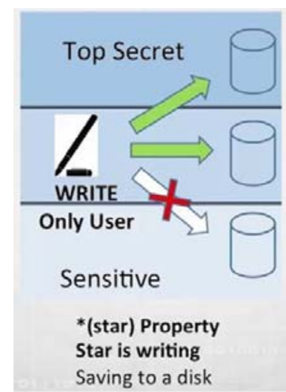
- No provision to "downgrade" the objects
  - As noted in step 4, the BLP model has no provision to manage the "downgrade" of objects
  - Although the requirements for multilevel security recognize that such a flow of information from a higher to a lower level may be required
    - provided it reflects the will of an authorized user
  - Hence, any practical implementation of a multilevel system has to support such a process in a controlled and monitored manner

Bell LaPadula Model

Limitations of BLP Model

- Classification creep
  - A subject constrained by the BLP model can only be "editing" (reading and writing) a file at one security level while also viewing files at the same or lower levels
  - If the new document consolidates information from a range of sources and levels, some of that information is now classified at a higher level than it was originally
  - This is known as *classification creep* and is a well-known concern when managing multilevel information
  - Again, some process of managed downgrading of information is needed to restore reasonable classification levels





Integrity Policies

Integrity Policies

Overview

- Requirements
  - Very different than confidentiality policies
- Biba's models
  - Strict Integrity policy
- Lipner's model
  - Combines Bell-LaPadula, Biba
- Clark-Wilson model
- Trust models
  - Policy-based
  - Reputation-based

The Biba Model

The Biba Model

Overview

- The Biba Model or Biba Integrity Model developed by Kenneth J. Biba in 1975
- The model is based on information flow, and the objects and subjects are grouped into ordered levels of integrity
- The Biba model was designed after the BLP model
  - sometimes called the Bell-LaPadula upside down model
- The model is designed so that subjects may not corrupt data in a level ranked higher than the subject, or be corrupted by data from a lower level than the subject.
- The model is also built on state transition system of computer security policy that describes a set of access control rules designed to ensure data integrity

The Biba Model

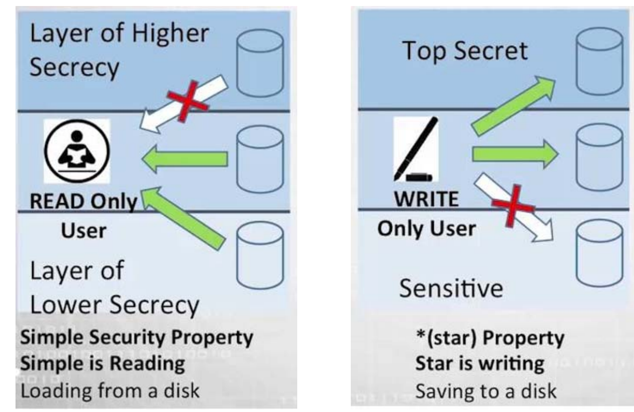
Overview

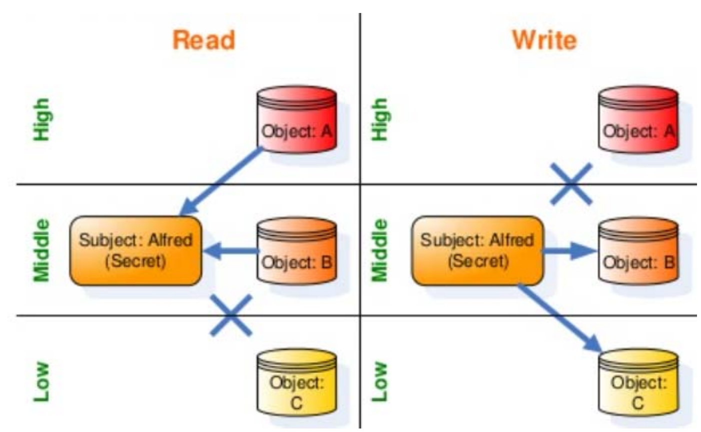
- Like other models, the Biba model supports the access control of both subjects and objects.
  - Subjects: (are users or processes acting on behalf of the users)
    - they are the active elements in the system that can access information
  - Objects:
    - are the passive system elements for which access can be requested (files, programs, etc.).
- Each subject and object will have a integrity level associated with it
  - denoted as I(S) and I(O) for subject S and object O, respectively
- A simple hierarchical classification uses a strict ordering of levels from lowest to highest
- Biba was designed to address three integrity issues:
  - Prevent modification of objects by unauthorized subjects.
  - Prevent unauthorized modification of objects by authorized subjects.
  - Protect internal and external object consistency

The Biba Model

Properties

- Basic properties or axioms of the Biba model state machine:
  - The Simple Integrity Property
    - A subject cannot read an object at a lower integrity level (no read-down).
  - The * (star) Integrity Property
    - A subject cannot modify an object at a higher integrity level (no write-up)
  - Invocation Property
    - A subject cannot send messages (logical request for service) to object of higher integrity





The Biba Model

Access Modes

- The Biba model consists of the following access modes:
- Modify:
  - The modify mode allows a subject to write to an object
  - This mode is similar to the write mode in other models
- Observe:
  - The observe mode allows a subject to read an object
  - This command is synonymous with the read command of most other models
- Invoke:
  - The invoke mode allows a subject to communicate with another subject
- Execute:
  - The execute mode allows a subject to execute an object
  - The command essentially allows a subject to execute a program which is the object

The Biba Model

Integrity Levels

- Each integrity level is represented as $L = (C, S)$ where:
  - L is the integrity level
  - C is the classification
  - S is the set of categories.
- The integrity levels then form a dominance relationship.
- Integrity level $L_1 = (C_1, S_1)$ dominates ($\geq$) integrity level $L_2 = (C_2, S_2)$ if and only if this relationship is satisfied:
  - $C_1 \geq C_2$ and $S_2 \subseteq S_1$

The Biba Model

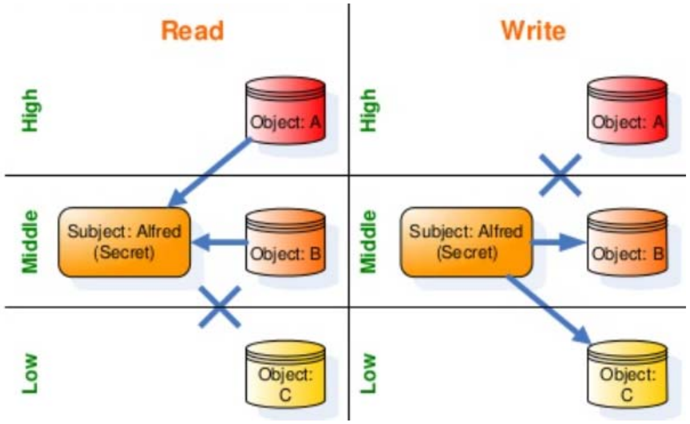
Biba Policies

- The Biba model is actually a family of different policies
- The goal of the model is to prevent the contamination of "clean" high level entities from "dirty" low level entities
- The model supports both mandatory and discretionary policies.
- The Mandatory Policies:
  - Strict Integrity Policy
  - Low-Watermark Policy for Subjects
  - Low-Watermark Policy for Objects
  - Low-Watermark Integrity Audit Policy
  - Ring Policy
- The Discretionary Policies:
  - Access Control Lists
  - Object Hierarchy

The Biba Model

Strict Integrity Policy

- Simple Integrity Condition ("no read-down"):
  - A subject can read an object only if : $I(S) \leq I(O)$.
  - $s \in S$ can observe $o \in O$ if and only if $i(s) \leq i(o)$

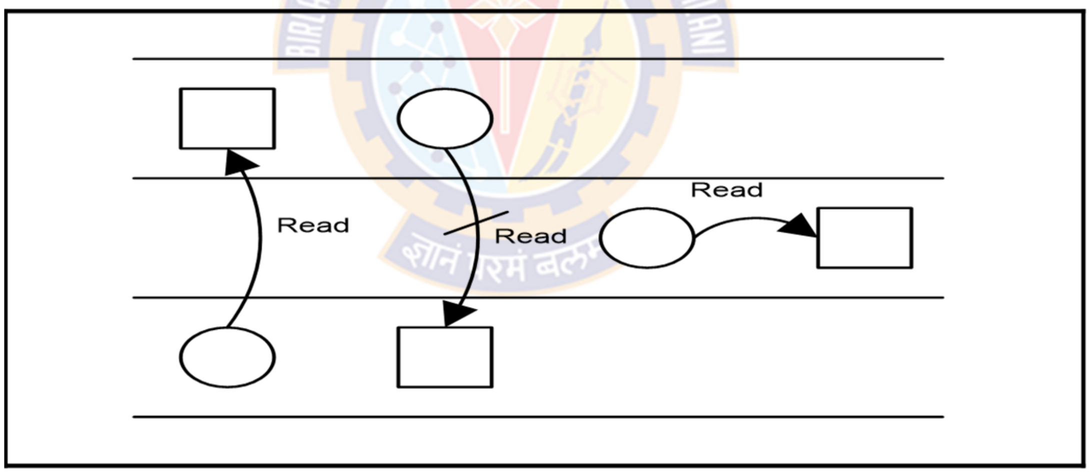


- Star Integrity Property ("no write-up"):
  - A subject can modify an object only if : $I(S) \geq I(O)$.
  - $s \in S$ can modify $o \in O$ if and only if $i(o) \leq i(s)$
- Invocation Property:
  - A subject can invoke/comm with another subject (E.g., software utility) only if : $I(S1) \geq I(S2)$.
  - $s_1 \in S$ can invoke $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$

The Biba Model

Strict Integrity Policy

- Simple Integrity Condition ("no read-down"):
  - A subject can read an object only if : $I(S) \leq I(O)$.
  - $s \in S$ can observe $o \in O$ if and only if $i(s) \leq i(o)$

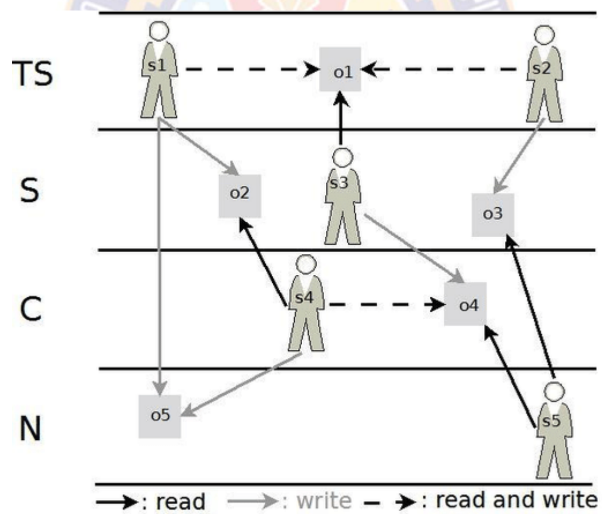


circle = subject, square = object

Biba Model

Strict Integrity Policy

- Simple Integrity Condition ("no read-down"):
  - A subject can read an object only if : I(S) ≤ I(O).
  - s ∈ S can observe o ∈ O if and only if i(s) ≤ i(o)



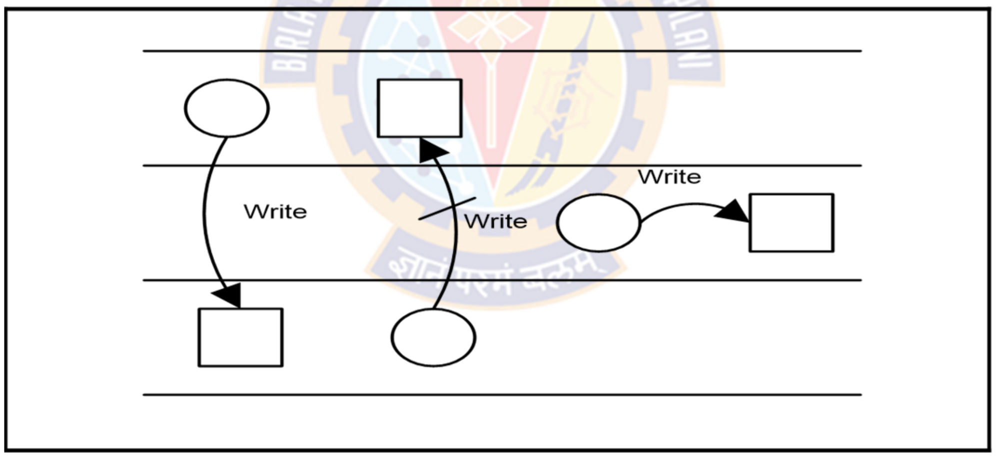
	o1	o2	o3	o4	o5
s1	read write	write			write
s2	read write		write		
s3	read			write	
s4		read		read write	write
s5			read	read	

Image Source: https://www.researchgate.net/figure/Two-Laws-of-Biba-Model-The-satisfaction-of-both-Biba-laws-prevents-the-information-flow_fig3_273706233

The Biba Model

Strict Integrity Policy

- Star Integrity Property ("no write-up"):
  - A subject can modify an object only if : $I(S) \geq I(O)$.
  - $s \in S$ can modify $o \in O$ if and only if $i(o) \leq i(s)$

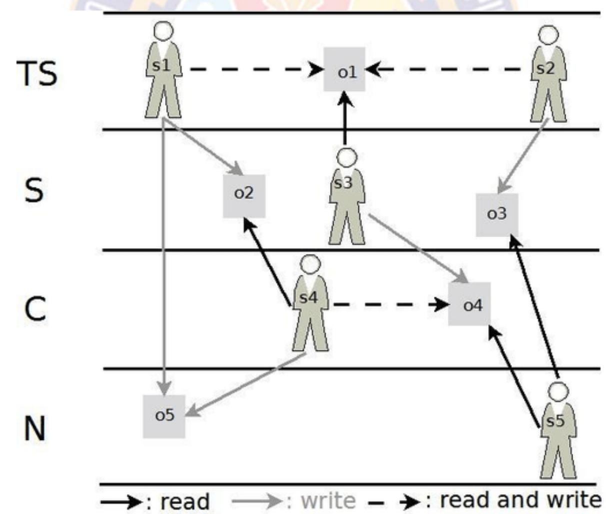


circle = subject, square =object

Biba Model

Strict Integrity Policy

- Star Integrity Property ("no write-up"):
  - A subject can modify an object only if : I(S) ≥ I(O).
  - s ∈ S can modify o ∈ O if and only if i(o) ≤ i(s)

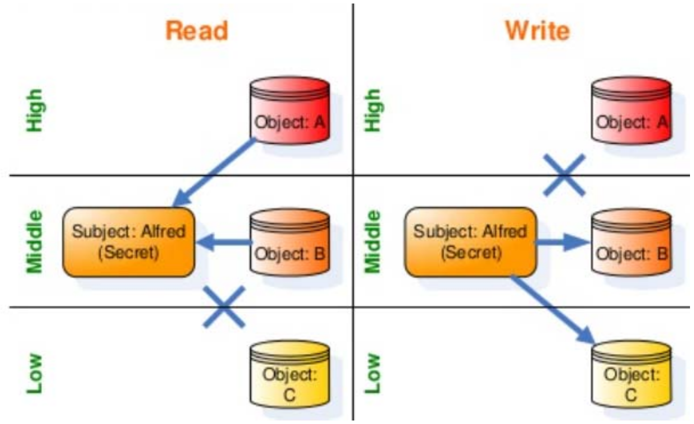


	o1	o2	o3	o4	o5
s1	read write	write			write
s2	read write		write		
s3	read			write	
s4		read		read write	write
s5			read	read	

Image Source: https://www.researchgate.net/figure/Two-Laws-of-Biba-Model-The-satisfaction-of-both-Biba-laws-prevents-the-information-flow_fig3_273706233

The Biba Model

Strict Integrity Policy

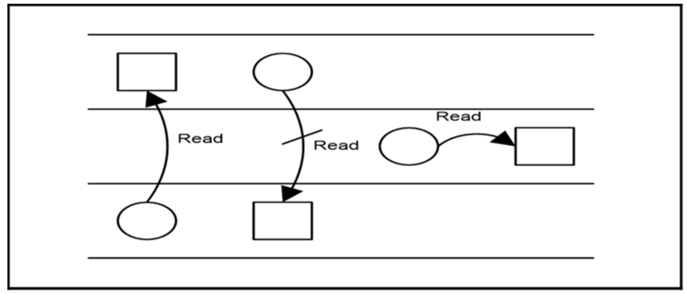


- The "no write-up" is essential because it limits the damage that can be done by malicious objects in the system
- For instance:
  - "no write-up" limits the amount of damage that can be done by a Trojan horse in the system
  - The Trojan horse would only be able to write to objects at its integrity level or lower
  - E.g., it limits the damage that can be done to the operating system.
- The "no read-down" prevents a trust subject from being contaminated by a less trusted object

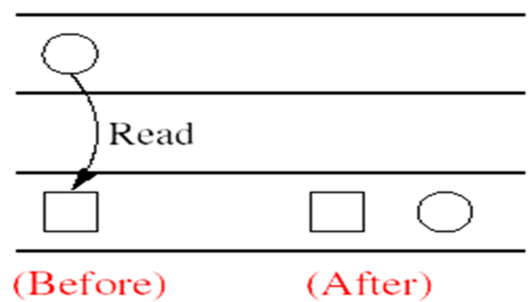
The Biba Model

Low-Water-Mark Policy

- The low-watermark policy for subjects
  - Is a relaxed "no read-down"
  - Contains these following rules:
    - Star Integrity Property:
      - s ∈ S can modify o ∈ O if and only if $i(o) \le i(s)$ ("no write-up")
    - A subject may examine any object:
      - If s ∈ S examines o ∈ O then $i'(s) = \min(i(s), i(o))$, where $i'(s)$ is the subjects integrity level after the read.
    - Invocation Property:
      - $s_1 \in S$ can invoke $s_2 \in S$ if and only if $i(s_2) \le i(s_1)$.



circle = subject, square = object
Simple Integrity Policy



circle = subject, square = object

The Biba Model

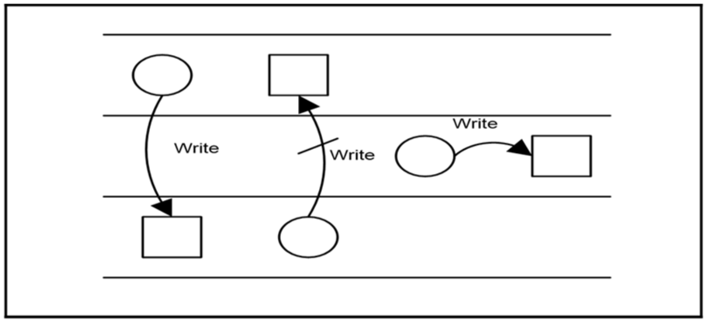
Low-Water-Mark Policy

- The low-watermark policy for subjects
  - Does nothing to restrict a subject from reading objects.
  - Is a dynamic policy, because it lowers the integrity level of a subject based on what objects are observed.
  - Drawback
    - One problem with this policy is that if a subject observes a less trusted object, it will drop the subjects integrity level to that of the object
    - Then later, if the subject needs to legitimately observe other objects, it may not be able to do so because the subjects integrity level has been lowered
    - The effect of this would be denial of service depending on the timing of the submissions.

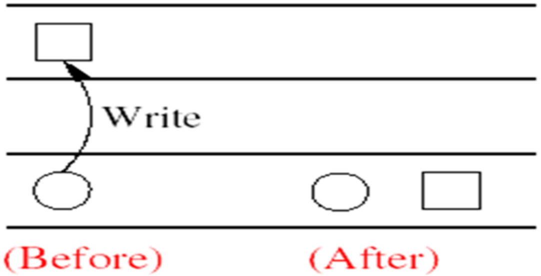
The Biba Model

Low-Water-Mark Policy

- The low-watermark policy for objects
  - Is a relaxed "no write-up"
  - Contains the following rules:
    - s ∈ S can modify any o ∈ O regardless of integrity level.
    - If s ∈ S modifies o ∈ O then
      - i'(o) = min(i(s),i(o)), where i'(o) is the objects integrity level after it is modified.



circle = subject, square =object
Integrity Star Property



circle = subject, square = object

The Biba Model

Low-Water-Mark Policy

- The low-watermark policy for objects
  - Is also a dynamic policy, similar to the low-watermark policy for subjects.
  - It does nothing to prevent an un-trusted subject from modifying a trusted object
    - In reality policy is not very practical.
  - The policy provides no real protection in a system
  - The policy simply lowers the trust placed in the objects
  - If a malicious program was inserted into the computer system it could modify any object in the system
  - This model would just lower the integrity level of objects that have become contaminated

The Biba Model

Low-Water-Mark Policy

- The low-watermark Integrity Audit Policy
  - The policy consists of the following rules:
    - Any subject may modify any object, regardless of integrity levels.
    - If a subject modifies an object at higher integrity level (a more trusted object), it results in the transaction being recorded in an audit log.
  - The drawback to this policy is it does nothing to prevent an improper modifications of an object
  - This policy is similar to the low-watermark for objects policy, except in this case the objects integrity level is not lowered, it is recorded.
  - This policy simply records that an improper modification took place.

The Biba Model

Drawbacks

- Advantages:
  - The Biba model is simple and easy to implement.
  - The Biba model provides a number of different policies that can be selected based on need.
- Disadvantages:
  - The model does nothing to enforce confidentiality.
  - The Biba model doesn't support the granting and revocation of authorization.
  - To use this model all computers in the system must support the labeling of integrity for both subjects and objects
  - To date, there is no network protocol that supports this labeling. So there are problems with using the Biba model in a network environment.

Thank You!