Information Security CS3002 (Sections BDS-7A/B) Lecture 27

Instructor: Dr. Syed Mohammad Irteza

Assistant Professor, Department of Computer Science

25 November, 2024

Previous Lecture

- Firewalls
  - Maps to Chapter 9 (some sections) in Computer Security: Principles and Practices (William Stallings)

CHAPTER 9

Firewalls and Intrusion Prevention Systems

Before Final Exam

Remaining Lectures (Content)

- ***Theoretical Models of Access Control (1 lecture)***
- Cybercrime Laws and Ethics (1 lecture)
- Project Presentations (2 lectures at least)

Security Issues

- *Complexity and human error*: writing firewall rules that implement the security policy is difficult for large networks
- *Bypassing security policies* using tunnels
- *Bypassing firewalls using other* networks (WiFi, mobile) or devices (laptop, USB)

Sandboxing

- The process of *isolating a program on the hard drive* in order to minimize or eliminate the exposure to other apps and critical system.
- Usually programs and applications interact with multiple parts of operating system and use *shared resources* like storage, memory and CPU sometimes causing conflicts.
- A malware, if present, *can utilize such vulnerabilities to cause a disaster*.
- *Sandboxing actually helps to reduce the impact that an individual program* will have on the system.

Examples of Sandboxing

Browser sandboxing

- *Google Chrome and Opera run in their own sandboxes*
- Others have an option of selective sandboxing e.g. *Mozilla*

Virtual Machines

- It is also called *manual sandboxing* to purposely configure the system to sandbox an application.
- Examples: VirtualBox, VMware

- Windows Sandbox
  - A temporary instance of host machine (*built into Windows 10 and Windows 11*)

Penetration Testing

- Penetration testing is the process of *evaluating the strengths of all security controls on a computer system or network.*
- Penetration tests evaluate *procedural*, *operational* as well as *technological controls*

External vs. Internal

- Penetration Testing can be performed from the viewpoint of an external attacker or a malicious employee.

Overt vs. Covert

- Penetration Testing can be performed with or without the knowledge of the IT department of the company being tested.

Penetration Testing

- ***Reconnaissance and Information Gathering***
  To discover as much information about a target (individual or organization) as possible without actually making network contact with said target

- ***Network Enumeration and Scanning***
  To discover existing networks owned by a target: i.e., active hosts, open ports and running services

- ***Vulnerability Testing and Exploitation***
  To check hosts for known vulnerabilities and to see if they are exploitable, as well as to assess the potential severity of said vulnerabilities

- ***Reporting***

Information Gathering

1. Find *domain* and *sub-domain* of the target
2. Find *similar* and *parallel domain names*
3. Web searches using *advanced operators*
4. Footprint the target using Shodan (*search engine for IoT devices*)
5. Find the *geographical location of company*
6. List *employees* and their *email addresses*
7. Identify the *key email addresses through email harvesting*
8. Find *key personnel* of the company
9. Browse social network websites to find *information about company* and *employees*
10. Identify the types of *network devices* used in organization

Information Gathering

11. Search the *archive.org* for old information about the company
12. Examine the *source code of web pages*
13. Perform *whois* lookup (e.g., [Free Whois Lookup - Whois IP Search & Whois Domain Lookup | Whois.com]())
14. Find IP *addresses block allocated* to organization
15. Find *DNS records* for domain
16. Perform *reverse lookup*
17. Perform *DNS zone transfer*
18. Draw a *network diagram* using traceroute analysis

Penetration testing types

- Black box
    - little or no information is provided about the specified target
- White box
    - where almost all the information about the target is provided
- Gray box
    - some information is being provided and some hidden

Theoretical Models of Access Control

- Confidentiality policies (BLP Model)
- Integrity policies (Biba Model)
- Integrity policies (Clark-Wilson Model)
- Hybrid policies (Chinese Wall Model)

Confidentiality Policy

- Goal: prevent the unauthorized disclosure of information
  - Deals with information flow
  - Integrity incidental

- Multi-level security models are best-known examples
  - ***Bell-LaPadula*** Model basis for many, or most, of these

Bell-LaPadula (BLP) Model

- Security levels arranged in linear ordering
  - Top Secret: *highest*
  - Secret
  - Confidential
  - Unclassified: *lowest*

- Levels consist of security clearance **L(s)**
- Objects have security classification **L(o)**

Bell-LaPadula (BLP) Model

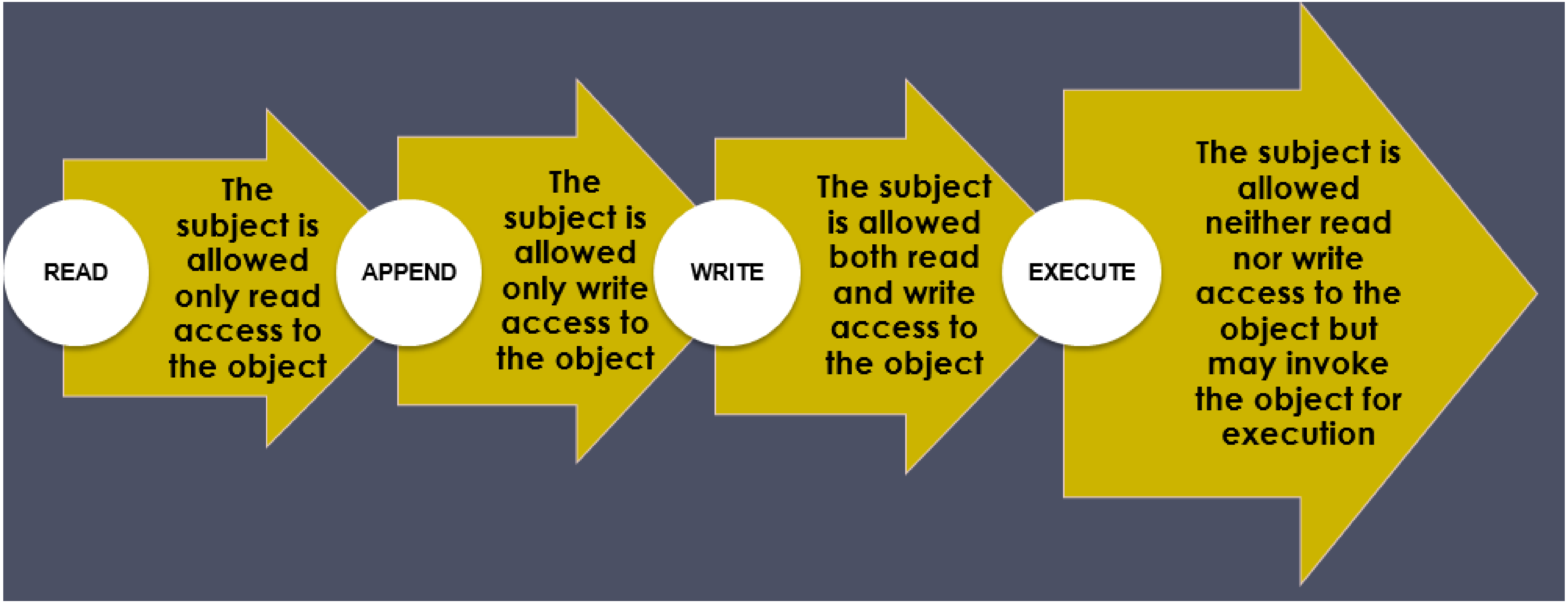
- Formal model for access control
- ***Subjects*** and ***objects*** are assigned a security class
- Form a hierarchy and are referred to as security levels
- A subject has a *security clearance*
- An object has a *security classification*
- Security classes control the manner by which a subject may access an object

A BLP Example

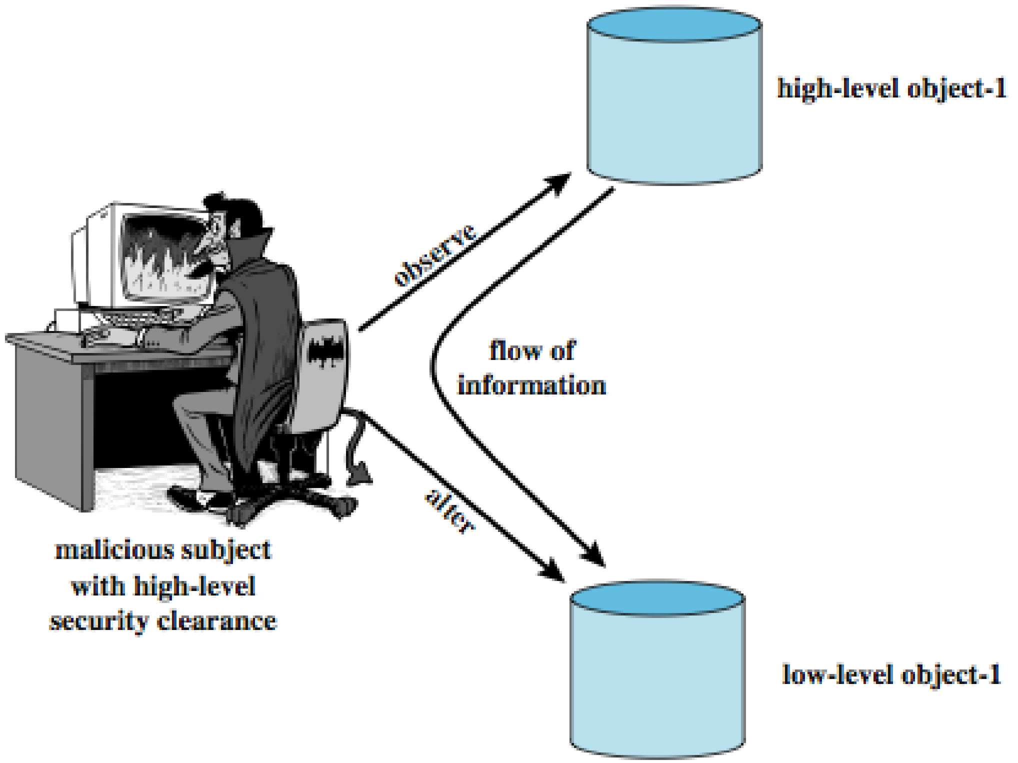
Security level	***Subject***	***Object***
Top Secret	Tamim	Personnel Files
Secret	Sohail	E-Mail Files
Confidential	Kaleem	Activity Logs
Unclassified	Jamal	Telephone Lists

- Tamim can read all files
- Kaleem cannot read Personnel or E-Mail Files
- Jamal can only read Telephone Lists

Access Privileges



Multilevel Security



- Multiple levels of security and data
- Subject at a high level may not convey info to a subject at a non-comparable level:
  - ***No read up (ss-property)***: a subject can only read an object of less or equal security level
  - ***No write down (*-property)***: a subject can only write into an object of greater or equal security level

BLP Formal Description

- Based on current state of system (*b, M, f, H*):
  - Current access set *b (subject, object, access-mode)*; it is the current access (not permanent)
  - Access matrix *M* (S_i *is permitted to access* O_j)
  - Level function *f: assigns security level to each subject and object*; a subject may operate at that (or lower) level
  - Hierarchy *H: a directed tree whose nodes are objects*:
    - Security level of an object must dominate (must be greater than) its parents

BLP Properties

- Three BLP properties: (c = current)
  - ss-property: (S_i, O_j, read) has $f_c(S_i) \geq f_o(O_j)$
  - *-property: $(S_i, O_j, \text{append})$ has $f_c(S_i) \leq f_o(O_j)$ and
    (S_i, O_j, write) has $f_c(S_i) = f_o(O_j)$
  - ds-property: (S_i, O_j, A_x) implies $A_x \in M[S_i,O_j]$

- BLP gives formal theorems
  - Theoretically possible to prove system is secure

ss-property: *simple security*

*-property: pronounced *star*

ds-property: *discretionary security*

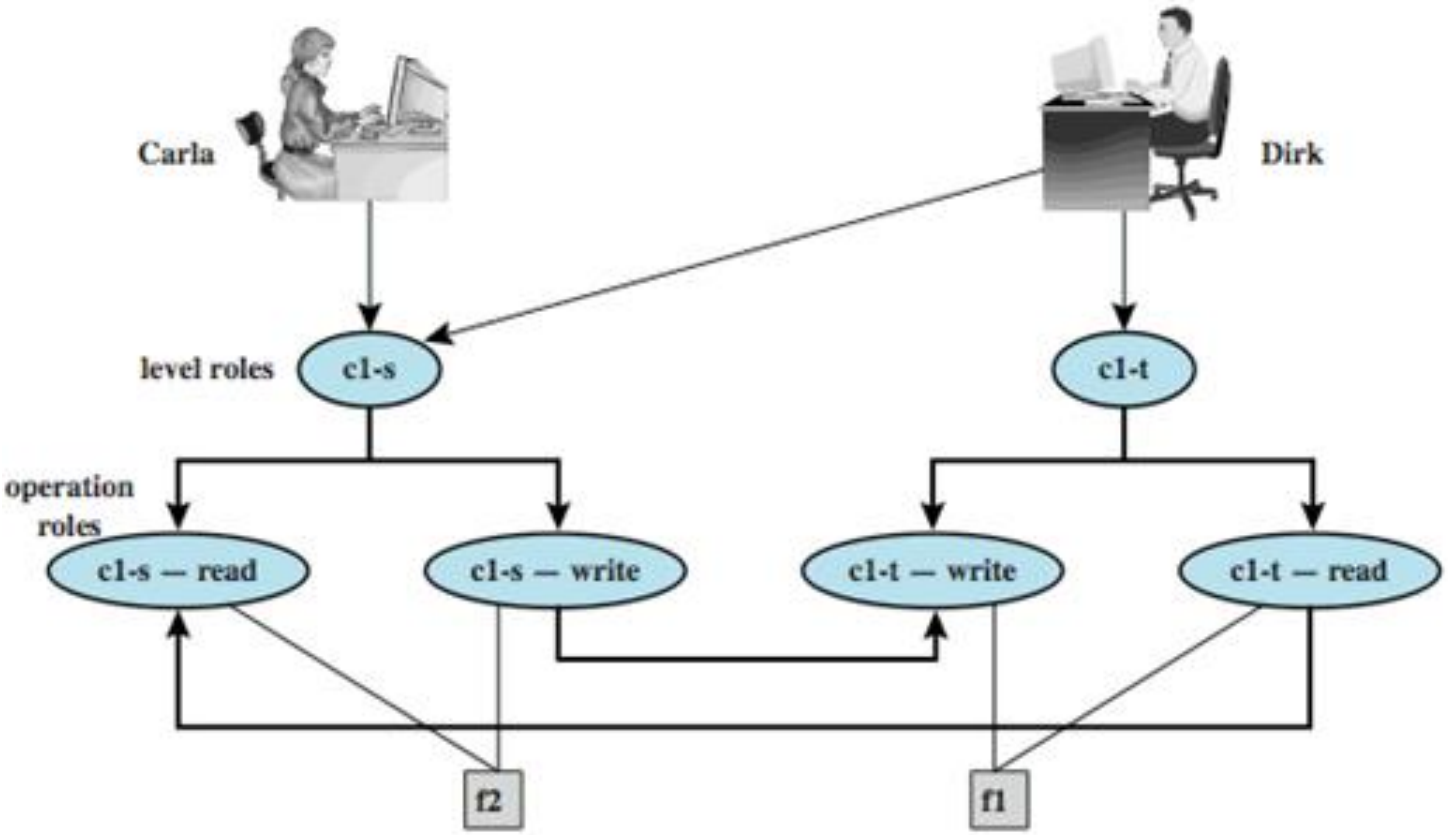
BLP Operations

1. get access: add (subject, object, access-mode) to b
   i. used by a subject to initiate an access to an object
2. release access: remove (subject, object, access-mode)
3. change object level
4. change current level (subject)
5. give access permission: Add an access mode to M (matrix)
   i. used by a subject to grant access mode on an object to another subject
6. rescind access permission: reverse of 5
7. create an object
8. delete a group of objects

BLP Example

- A role-based access control system
- Two users:
  - Carla → student (s) in course c1
  - Dirk → teacher (t) in course c1
- Classes
  - Carla (Class: s)
  - Dirk (Class: t); can also login as a student, thus (Class: s)
- A student role has a *lower security clearance*
- A teacher role has a *higher security clearance*

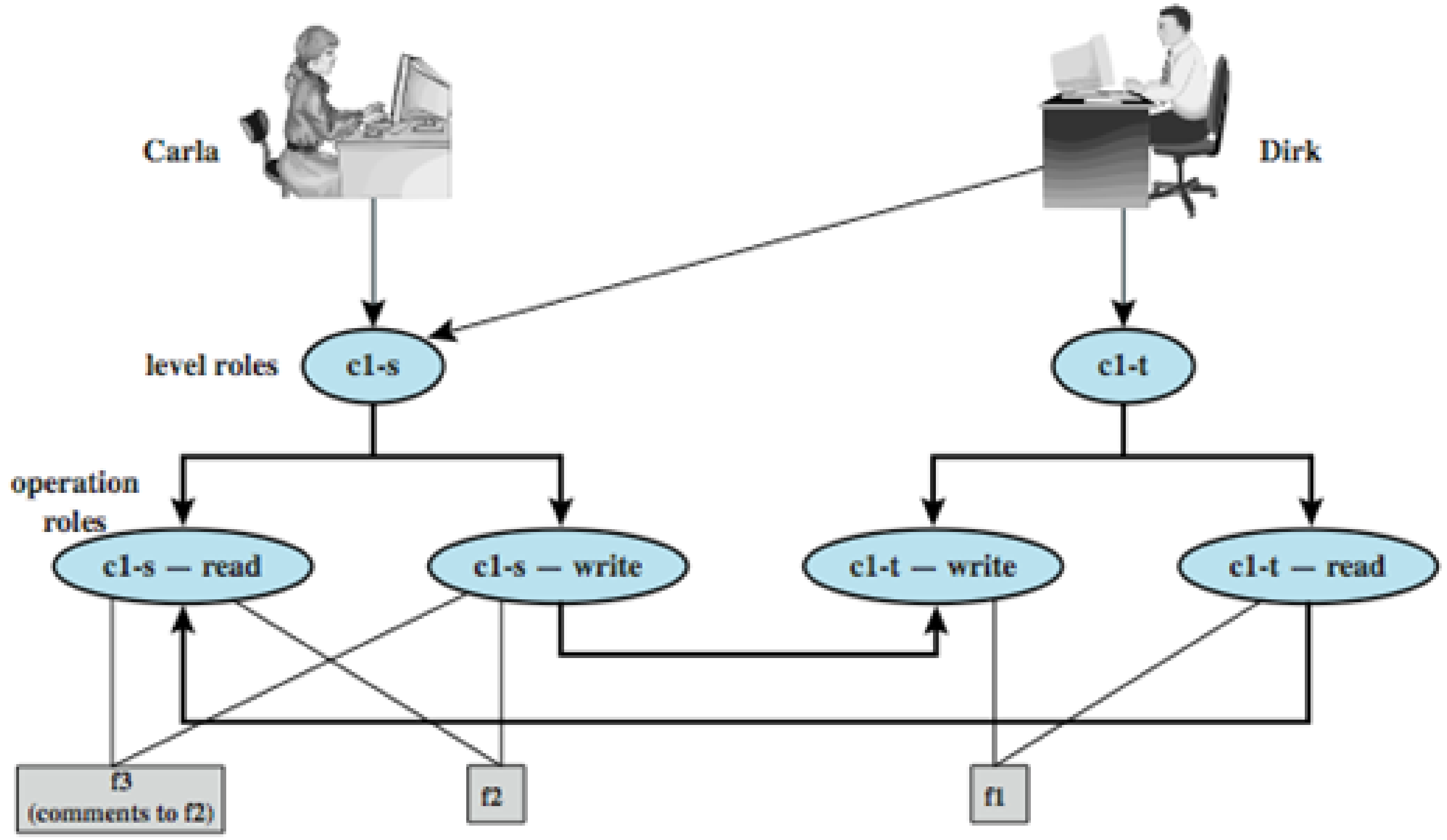
BLP Example



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

- Dirk creates **f1**; Carla creates **f2**
- Carla can read/write to **f2** but cannot read **f1**
- Dirk can read/write **f1** and **f2** (if permitted, i.e., if Carla grants access to **f2**)
- Dirk can read/write **f2** only as a student

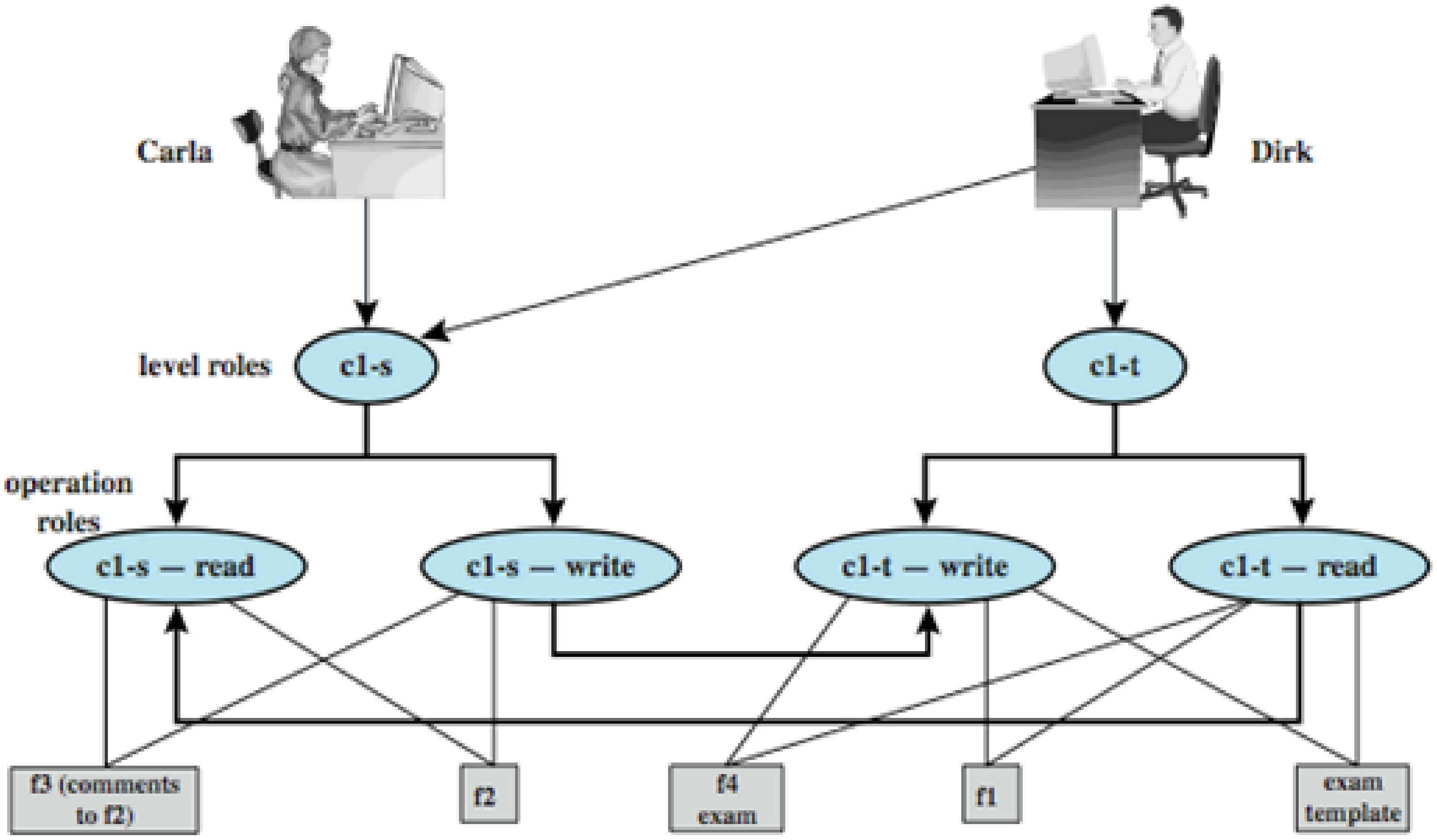
BLP Example (cont.)



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

- Dirk reads **f2**; wants to create **f3** (comments)
- Dirk signs in as a student (so Carla can read)
- As a teacher, Dirk cannot create a file at student classification

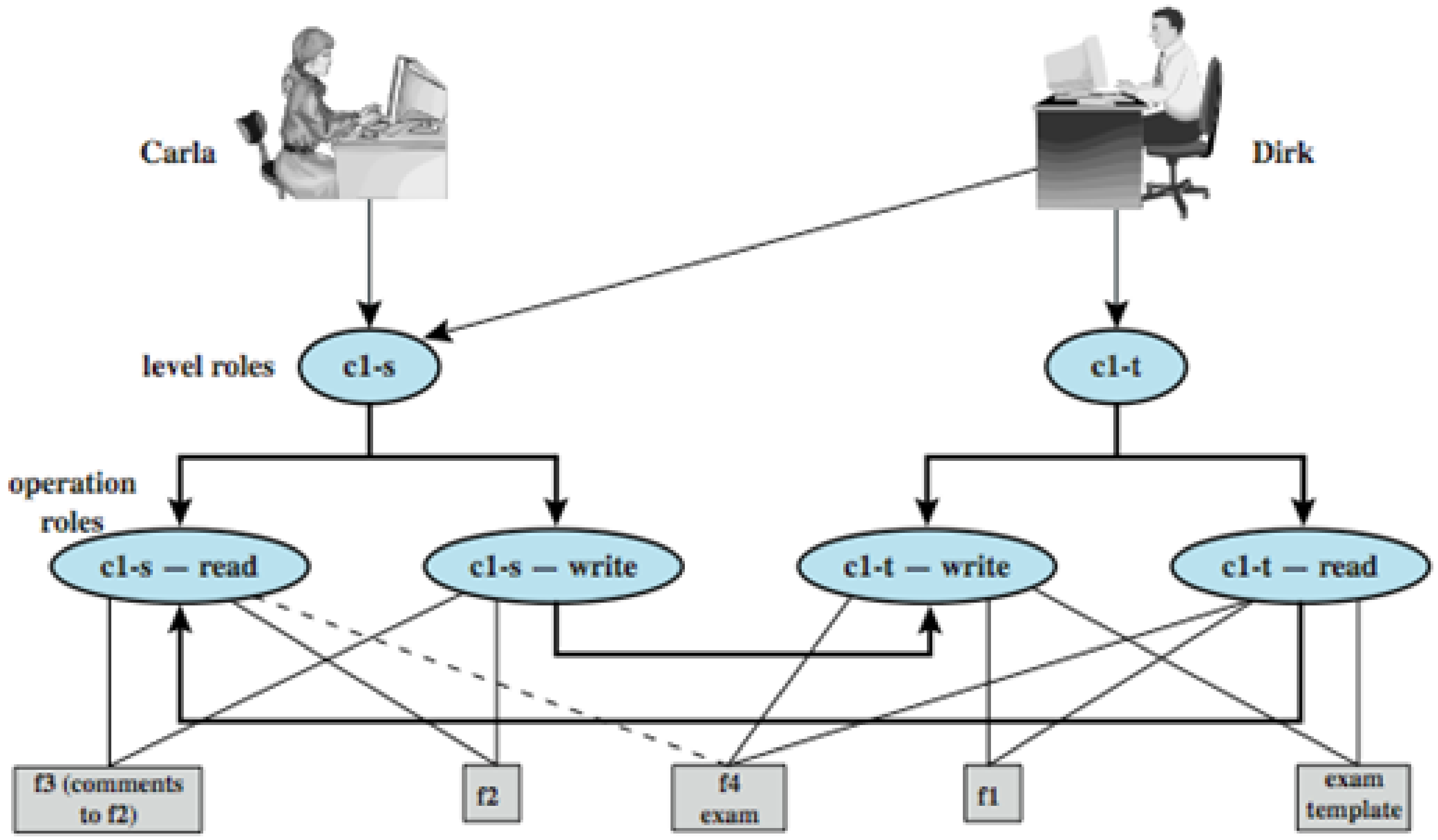
BLP Example (cont.)



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

- Dirk as a teacher creates exam (**f4**)
- Must log in as a teacher to read template

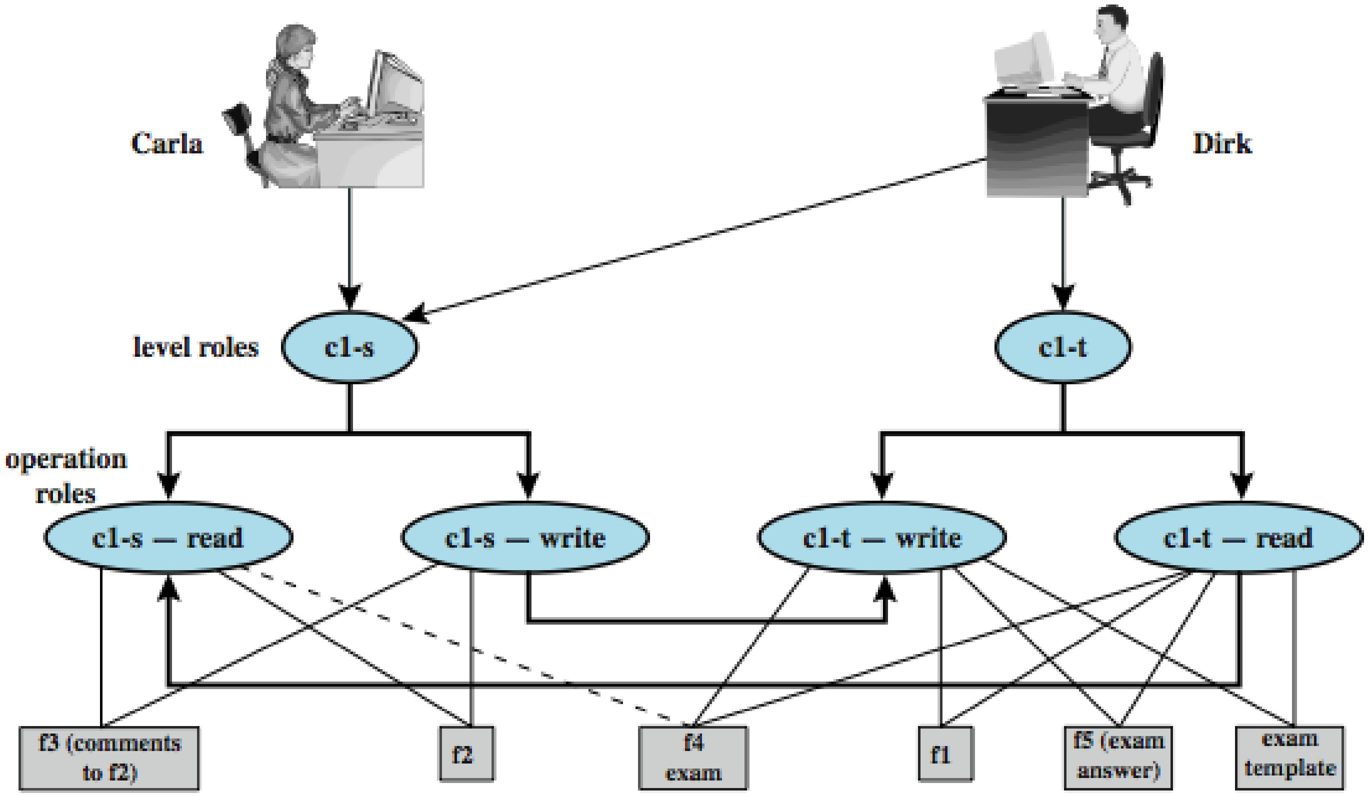
BLP Example (cont.)



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

- Dirk wants to give Carla access to read **f4**
- Dirk can't do that; an admin must do
- An admin downgrades **f4** class to c1-s

BLP Example (cont.)



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

- Carla writes answers to **f5** (at c1-t level)
  - An example of *write up*
- Dirk can read **f5**
  - Note: Carla can still see her answers at her workstation but cannot access f5 for reading

Reading Information - New

- "Reads up" disallowed, "reads down" allowed
- Simple Security Condition
  - Subject ***s*** can read object ***o*** iff ***L(s)*** dominates ***L(o)*** and ***s*** has permission to read ***o***
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "*no reads up*" rule

Writing Information - New

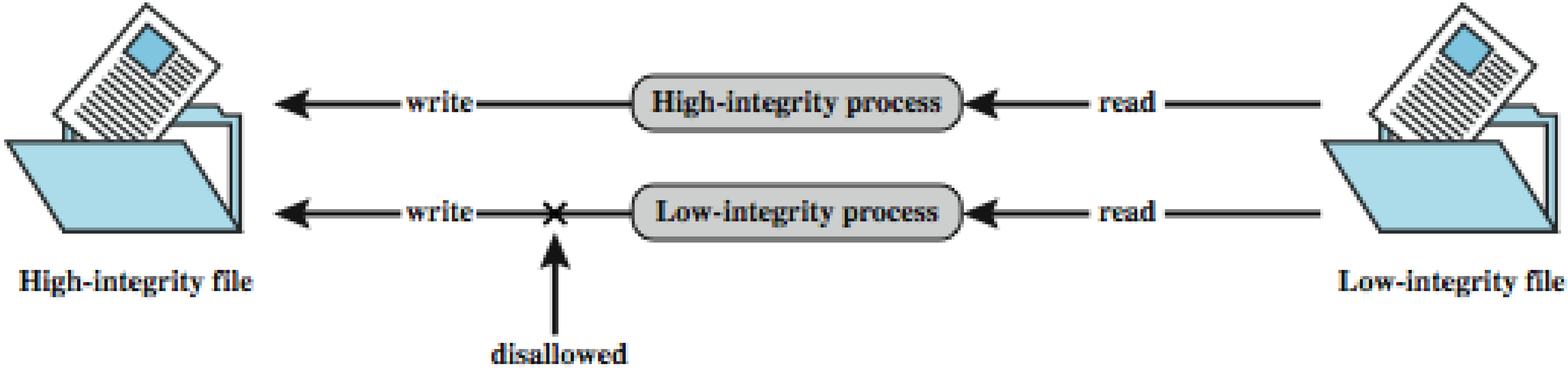
- Information flows up, not down
  - “Writes up” allowed, “writes down” disallowed
- *-Property (Step 2)
  - Subject ***s*** can write object ***o*** iff ***L(o)*** dominates ***L(s)*** and ***s*** has permission to write ***o***
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called “*no writes down*” rule

Limitation of BLP model

- Incompatibility of *confidentiality* and *integrity*
- Classification of data *changes over time*
- If data needs to migrate to *higher security classification*, a *trusted user has to be downgraded*!
- In the presence of *shared resources*, **-property* may not be enforced
- A bit *complex* to implement

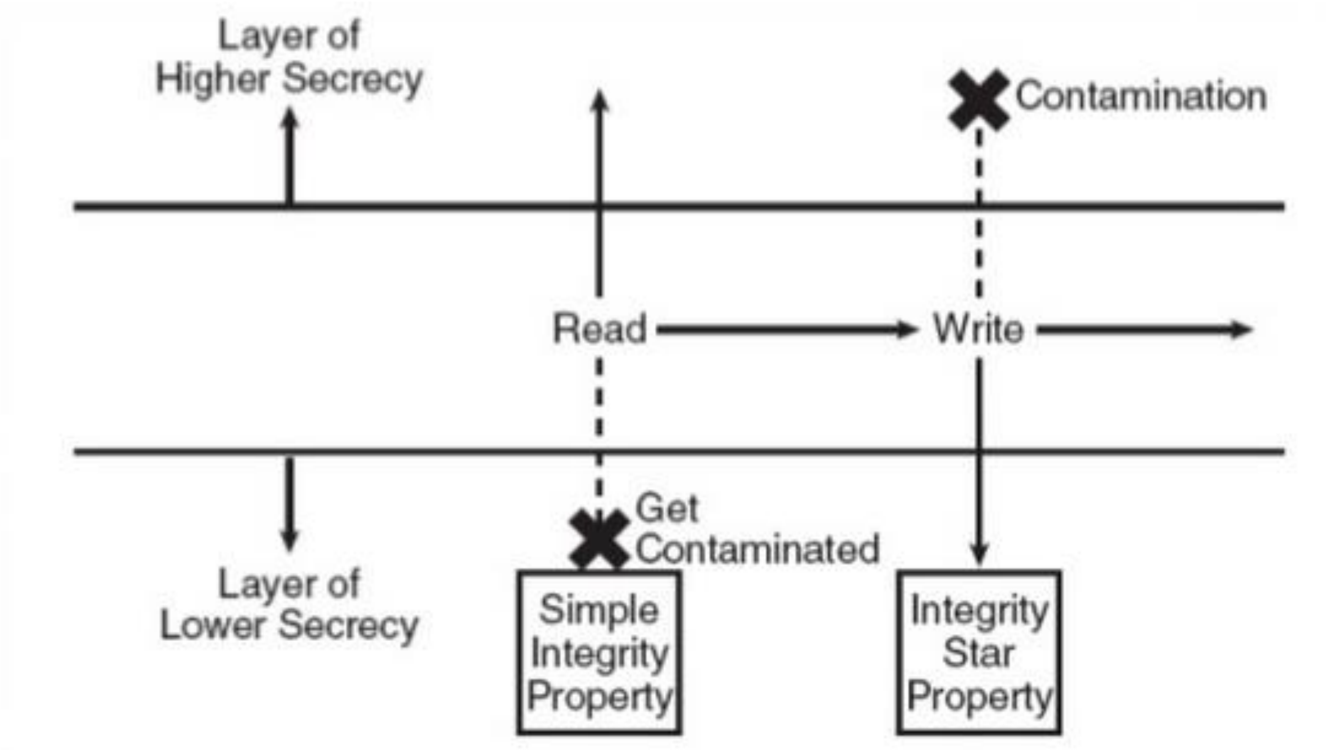
Biba Integrity Model

- Deals with integrity and deal with the case where data must be visible at multiple security levels but should be modified in a controlled way.
- ***Strict integrity policy***:
  - Simple integrity: *modify only if* $I(S) \geq I(O)$
  - Integrity confinement: *read only if* $I(S) \leq I(O)$
  - Invocation property: *invoke/comm only if* $I(S_1) \geq I(S_2)$



Biba Integrity Model

- Simple integrity: *modify only if* I(S) ≥ I(O)
- Integrity confinement: *read only if* I(S) ≤ I(O)
- Invocation property: *invoke/comm only if* I(S1) ≥ I(S2)



Appendix

- Confidentiality Model:
  - Bell LaPadula Model
- Integrity Model:
  - Foundations of Computer Security - Lecture 21: Modeling Integrity: Biba
- Sandboxing:
  - What is Sandboxing and How to Sandbox a Program | Comparitech

Acknowledgments

- Dr. Haroon Mahmood and other FAST-NU instructors