NYIT
NEW YORK INSTITUTE OF TECHNOLOGY

CSCI 620 – Operating Systems Security

Chapter 5

Chapter 5

- Verifiable Security Goals

Chapter Overview

- Information Flow Confidentiality Models:
  - Denning's Lattice Model
  - Bell-LaPadula Model
- Information Flow Integrity Models:
  - Biba Model
  - Low-Water Mark Model
  - Clark-Wilson
  - Challenges
- Covert Channels: types and controls

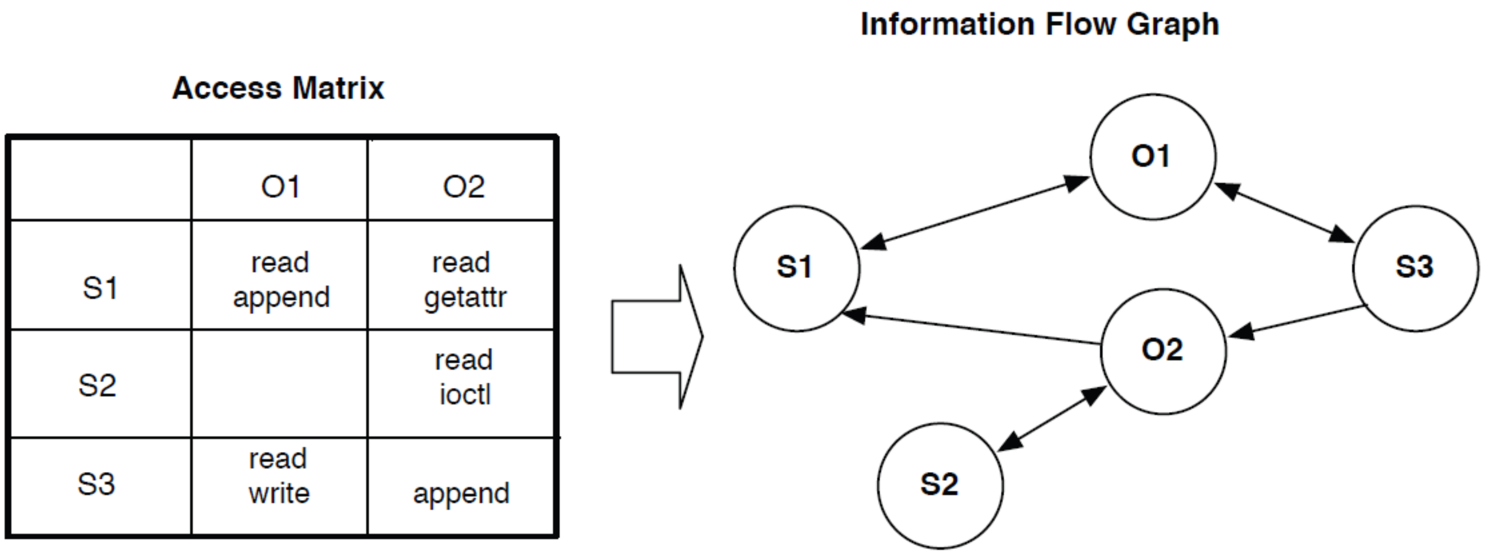
Information Flow

- Concept of information flow.
- Always between a subject (*s*) and an object (*o*).
- Cases:
  - Read: $s \leftarrow o$
  - Write: $s \rightarrow o$
- Note that every operation on an object is either a read or a write or a combination (an execution is a read, for example). Thus, every operation of a subject on an object can be seen as a an arrow from one to the other or a double-ended arrow.

Information flow graph

- Given a protection state, we can associate an *information flow graph which* is a directed graph where:
  - Nodes: union of subjects and objects
  - Edges: There is an edge from node *a* to node *b* if there is possible information flow from *a* to *b*. If the flow can go both ways, we simply assign both edges.

Information Flow Graph example



Access Matrix

	O1	O2
S1	read append	read getattr
S2		read ioctl
S3	read write	append

Figure 5.1: The information flow graph on the right represents the information flows described by the access control matrix on the left.

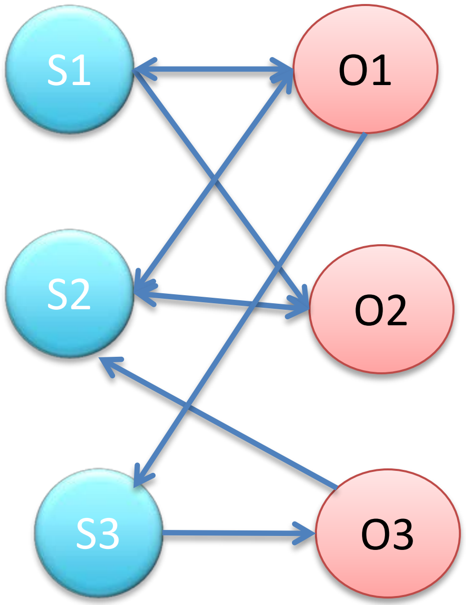
Exercise

- Develop the information flow graph for:

	O1	O2	O3
S1	R/W	W	
S2	R/W	R/W	R
S3	R		W

Solution

	O1	O2	O3
S1	R/W	W	
S2	R/W	R/W	R
S3	R		W

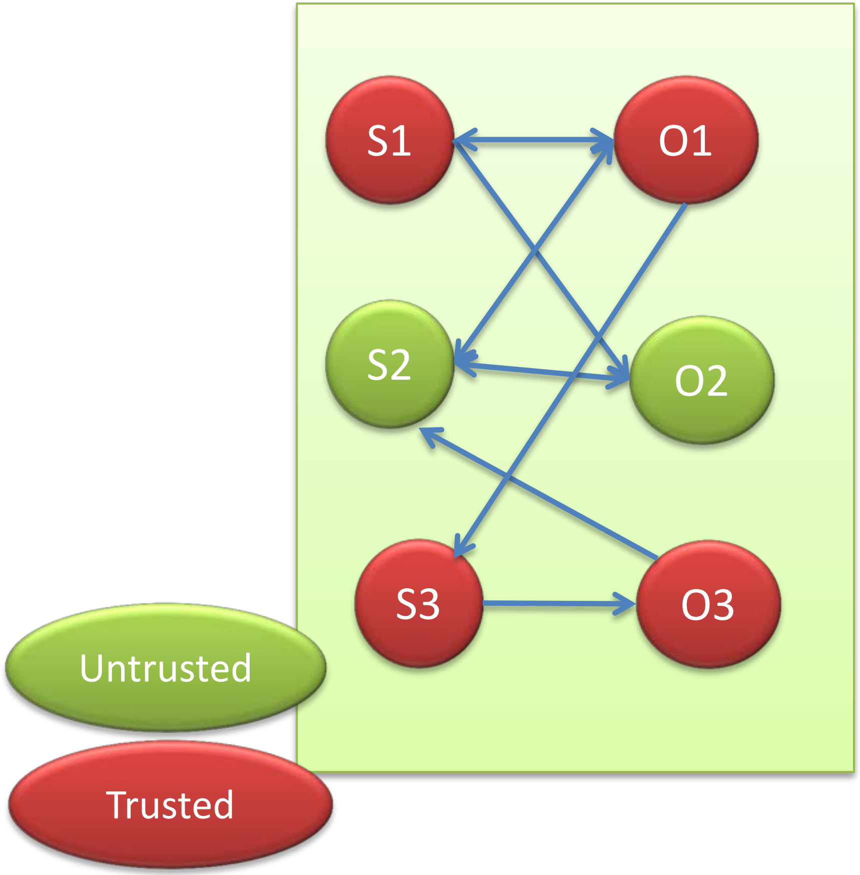


Application of Information Flow Graphs

- Information Flow graphs can be used to check for confidentiality and integrity:
  - For confidentiality, we can check whether there is a path from an object *o* which should be kept secret, to a subject *s* which should not read it. If there is such a path, we say there is a *leak* in the protection state.
  - For integrity, we can check whether there is a path from a low integrity subject *s1* (a possible attacker) to a high integrity subject *s2*. In this case we say that *s2 depends on s1*.

Follow on to Previous Example

- In this Information Flow Graph, identify all cases of *leak* in the protection state.
- Answer:
  - O3 to S2
  - Any others?



Information Flow Confidentiality Models

- The point is to ensure that, no matter which programs are run (i.e. malware), no information is leaked.
- Discretionary systems cannot work, since the malware can change the protection system to introduce a leak.
- Systems like Windows and Unix also have the problem that they do not account for all the information flows that may happen.

Information Flow Confidentiality Models: Outline

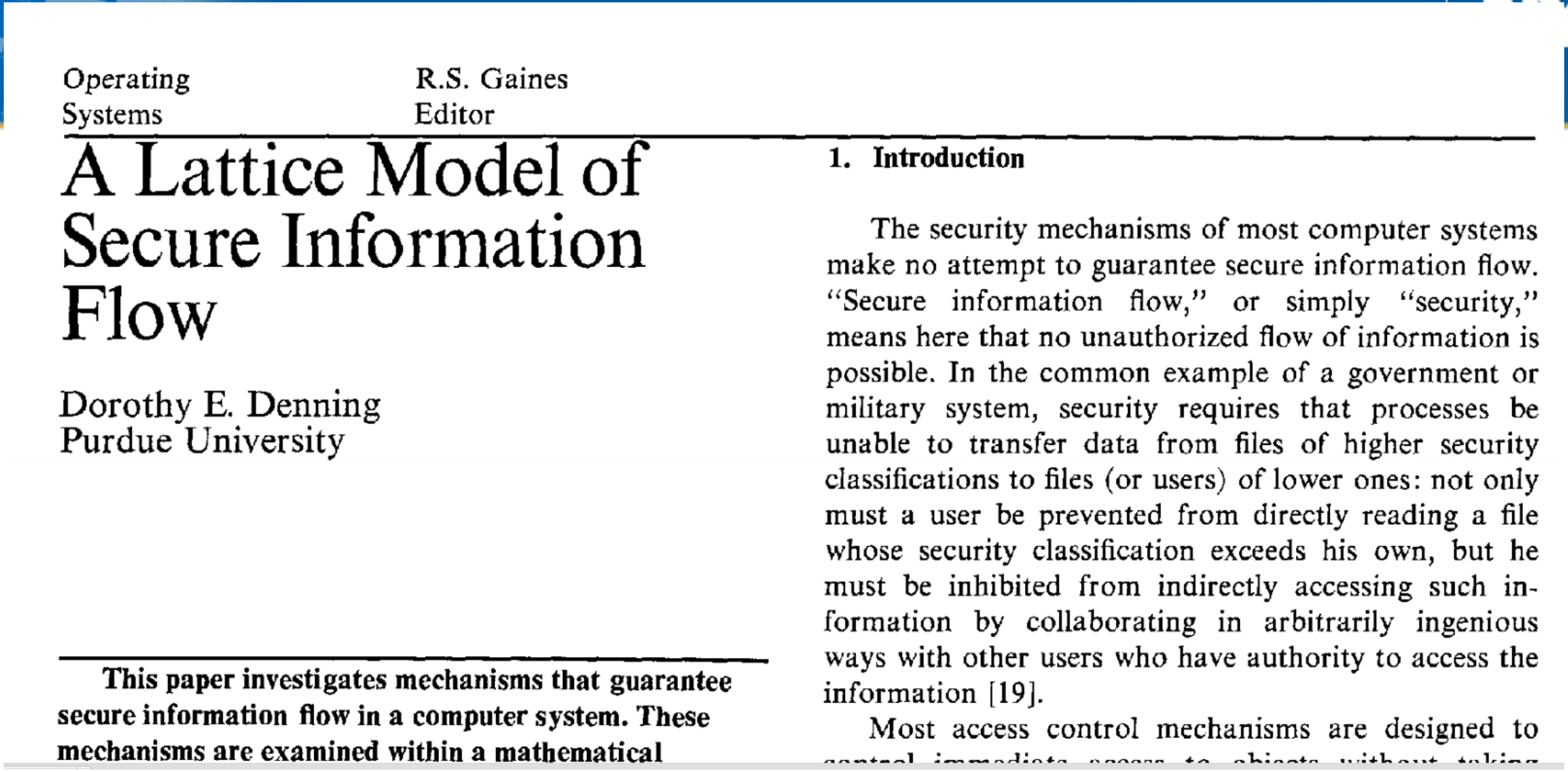
- Set up protection system so there are no leaks.
- No changes to the protection system are allowed.
- Models to be presented:
  - Denning's Lattice Model
  - Bell-LaPadula Model

Denning's Lattice Model

- Define an *information flow model* as a quintuple of:
  - objects (*N*),
  - subjects/processes (*P*),
  - a collection of *security classes* (SC),
  - a relation (→) (can flow, between members of SC)
  - a binary *join* operation on SC (⊕)

The join of two security classes is the result of combining data from the two classes.

It is customary to picture models by the security classes.



Operating Systems

R.S. Gaines Editor

A Lattice Model of Secure Information Flow

Dorothy E. Denning
Purdue University

This paper investigates mechanisms that guarantee secure information flow in a computer system. These mechanisms are examined within a mathematical

1. Introduction

The security mechanisms of most computer systems make no attempt to guarantee secure information flow. "Secure information flow," or simply "security," means here that no unauthorized flow of information is possible. In the common example of a government or military system, security requires that processes be unable to transfer data from files of higher security classifications to files (or users) of lower ones: not only must a user be prevented from directly reading a file whose security classification exceeds his own, but he must be inhibited from indirectly accessing such information by collaborating in arbitrarily ingenious ways with other users who have authority to access the information [19].

Most access control mechanisms are designed to

Two Information Flow Model Policies.

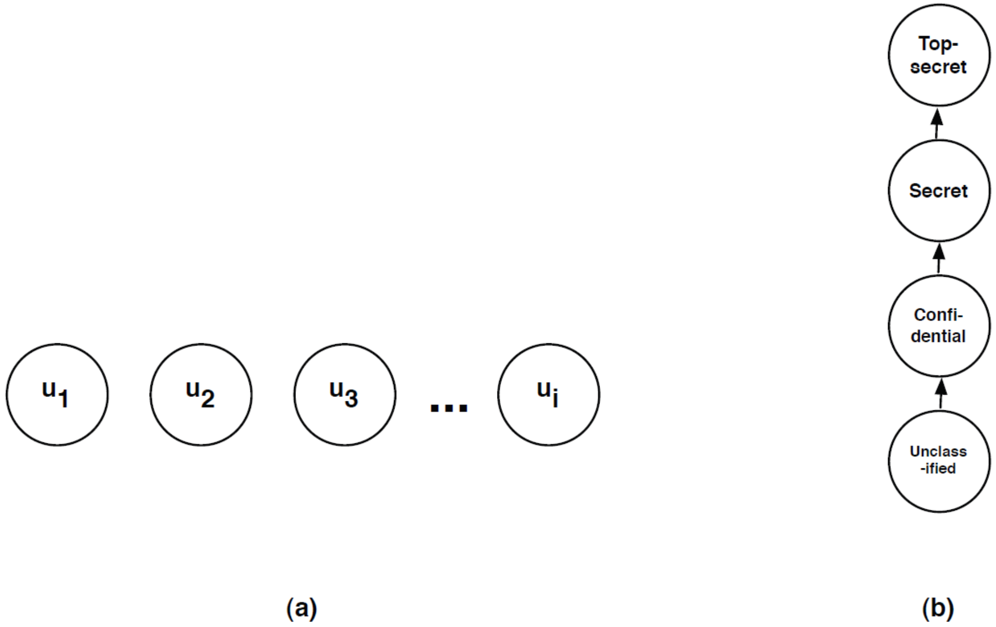


Figure 5.2: Two information flow model policies: (a) consists of isolated security class where no information flows among them and (b) is a totally-ordered sequence of security classes where information flows upwards only.

Finite Lattice Models

- An information flow model forms a finite lattice if it is a finite lattice in the finite math sense, i.e.:
  - The set SC is finite
  - The relation $\rightarrow$ is a partial order on SC
  - SC has a lower bound w.r.t $\rightarrow$
  - The join operator is a totally defined least upper bound operator.

One more useful definition

- $A \geq B$ (A *dominates* B) if and only if $B \rightarrow A$.
- The *strictly dominates* relation is defined by $A > B$ if and only if $A \geq B$ but $A \neq B$.
- A and B are *comparable* if either $A \geq B$ or $B \geq A$. Otherwise A and B are incomparable.

Bell-LaPadula Model

- BLP actually refers to a variety of models.
- The most common variant allows for two dimensions: a sensitivity level and a need-to-know compartment.

BLP policy example

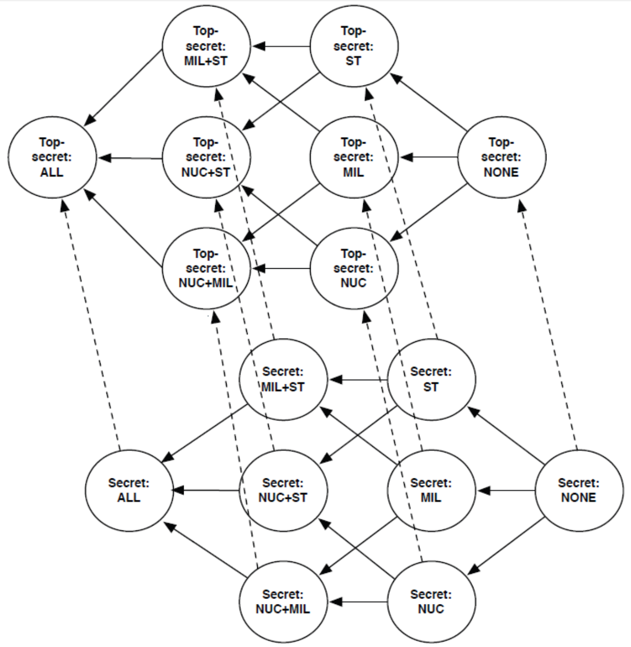


Figure 5.3: This a Haase diagram (with the information flow direction added in edges) of a Bell-LaPadula policy consisting of two sensitivity levels (*top-secret* and *secret* where *top-secret* dominates) and three categories (*NUC*, *MIL*, and *ST*). The edges show the information flows authorized by the Bell-LaPadula model for this lattice.

BLP laws

- The BLP model defines security classes SC and two properties:
  - *Simple-security property:* subject s can read object o only if SC(s) ≥ SC(o)
  - *-security property subject s can write object o only if SC(o) ≥ SC(s)
- Note that BLP assigns labels to subjects and objects; these labels may not change. That is called the *tranquility principle*.

Information Flow Integrity Models

- Define high integrity as originating from known, trusted sources.
- Define a process to be of high integrity if it does not depend on any low integrity inputs.
- Integrity can also be mapped to information flows: if information flows from a low integrity process to a high integrity process, we have an integrity compromise.

Biba Integrity Model

- In the Biba integrity model, classes of integrity (IC) are defined, from lower to higher.
- Typical level names could be trusted, system, application, user, untrusted, but there is no standard.
- The *simple-integrity property* states that a subjects can read an object o only if IC(s) ≤ IC(o) and the *-integrity property states that subject s can write object o only if IC(s) ≥ IC(o).
- Note this is the reverse of confidentiality.
- For this reason, the confidentiality levels and the integrity levels are defined differently.

Biba vs. Bell-LaPadula

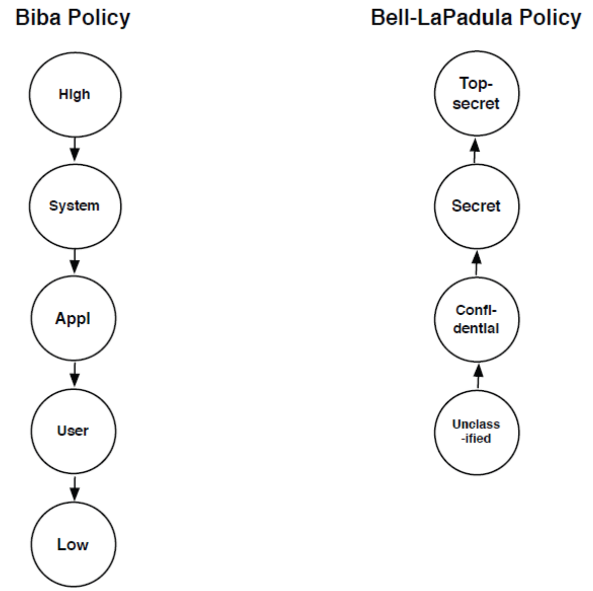


Figure 5.4: For a system that enforces both secrecy and integrity goals, Biba and Bell-LaPadula can be jointly applied. Subjects and objects will be assigned both Biba integrity classes and Bell-LaPadula secrecy classes from the set as shown.

A few more comments about Biba Policy

- No practical analogs.
- Another way to limit data transfers from low integrity items to high integrity items is by using *guards* to filter untrusted input.
- Often, guards are application specific.
- Biba model has not been applied extensively.

Low-Water Mark Integrity aka LOMAC

- Define integrity classes as before.
- Integrity changes according to inputs: a subject who reads a lower integrity object acquires its integrity; an object written by a lower integrity subject gets its own integrity class lowered.
- The idea is that low integrity items can affect fewer items.
- Limited use.

Clark-Wilson Integrity

- Unconstrained Data Items (UDI)
- Constrained Data Items (CDI)
  - Validated by Integrity verification procedures (IVP)
  - Modified by transformation procedures (TP)

Clark Wilson Rules

- Certification Rules:
  - When an IVP is executed, it must ensure the CDI's are valid
  - Each TP must transform CDI's from one valid state to another
- Enforcement Rules:
  - System must ensure only TP's certified to run on a CDI change that CDI. (Certifier may not be able to execute the TP)
  - Each TP must be associated with a set of CDI's.
  - Only the certifier of a TP may change its CDI's.
- Other rules required:separation of duty, authentication at the transaction level, logs allowing operation reconstruction, and UDI cleansing.

The Challenge of Trusted Processes

- These policies cannot be applied blindly.
- The set of trusted programs/processes is often big, even in systems desiged to be secure. e.g. SELinux/MLS has over 30 trusted subjects.
- Security-typed languages.
- There is a lot more work in progress.

Covert Channels

- In computer security, a covert channel is a type of computer security attack that creates a capability to transfer information objects between processes that are not supposed to be allowed to communicate by the computer security policy.
- The term, originated in 1973 by Lampson is defined as "(channels) not intended for information transfer at all, such as the service program's effect on system load." to distinguish it from Legitimate channels that are subjected to access controls by COMPUSEC

Covert Channels

- Two types
  - Storage channels - Communicate by modifying a "storage location"
  - Timing channels - Perform operations that affect the "real response time observed" by the receiver
  - Identification (Kemmerer)
  - Shared Resource Matrix
  - Covert Flow Trees
- Timing covert channels cannot be completely eliminated.

Timing Channels[1]

http://en.wikipedia.org/wiki/Timing_channel

- A timing channel is one example of a covert channel for passing unauthorized information, in which one process signals information to another process by modulating its own use of system resources (e.g., central processing unit time) in such a way that this manipulation affects the real response time observed by the second process.

- A service program uses timing channel to communicate by using or not using an assigned amount of computing time. In the simple case, a multi-programmed system with two processes divides time into blocks and allocates blocks of processing alternately to one process and the other.

Non-interference

- (Informal discussion only)
- Define a function *purge* which guarantees that high confidentiality objects have no effect on low confidentiality objects.
- But low confidentiality objects may have an effect on high confidentiality objects and thereby learn their state.
- Not very developed.

Questions?