

Overview

Applying Formal Methods, Part III

Overview

- Security mechanisms
 - Mechanisms that provide the means to enforce security policies
- Security policies
 - Policies that define what is allowed and what is prohibited

Overview (cont.)

- Three ways to classify security policies
 1. When and to whom the policies will apply
 - Discretionary or mandatory
 2. The contexts on which the policies are based
 - Military or commercial
 3. Which aspects the policies address
 - Confidentiality, integrity, or availability

C I A

Overview

The End

Access-Control Mechanisms: Lists

Access-Control Lists

Lists

- An access-control mechanism can make use of a *list* of principals, say L , with the right to protected objects.
 - The list L : *access-control list* (ACL)
 - The mechanism: *access-control list scheme*
 - Different from ticket-oriented access control
- The principals do not possess the credential that says they have a right to access a protected resource.

Access-Control Lists (ACLs)

- Example: access to a government facility
 - Scientist X arrives at Y (a government lab) and requests to enter Y, where there are guards at the entrance.
 - The guards check X's identity and the guest list they possess before granting access to X.
- An access-control list can be represented as a principal in access-control logic

General Form of Access-Control Lists

- Form of a subject making a request:

subject says $\langle \text{accessright}, \text{object} \rangle$

ϕ

- Authority's jurisdiction to set policy

Authority controls (subject controls $\langle \text{accessright}, \text{object} \rangle$)

- General form of a simple access-control list entry:

\uparrow
list ACL says $\left\{ \begin{array}{l} \text{subject}_1 \text{ controls } \langle \text{accessright}_1, \text{object}_1 \rangle \wedge \\ \text{subject}_2 \text{ controls } \langle \text{accessright}_2, \text{object}_2 \rangle \wedge \\ \vdots \\ \text{subject}_n \text{ controls } \langle \text{accessright}_n, \text{object}_n \rangle \end{array} \right.$
 \downarrow

- Trust assumption: $\text{ACL} \Rightarrow \text{Authority}$

Using Access-Control Lists

Recall that we have the following derived inference rules:

$$\text{Simplify Says 1} \quad \frac{P \text{ says } (\varphi_1 \wedge \varphi_2)}{P \text{ says } \varphi_1} \quad \text{Simplify Says 2} \quad \frac{P \text{ says } (\varphi_1 \wedge \varphi_2)}{P \text{ says } \varphi_2}$$

Using these rules, we can always derive

$$\longrightarrow \quad ACL \text{ says } (\underbrace{subject_i \text{ controls } \varphi_i})$$

from the larger ACL:

$$ACL \text{ says } \left\{ \begin{array}{l} \vdots \\ \underbrace{subject_i \text{ controls } \varphi_i} \wedge \\ \vdots \end{array} \right.$$

Example: Invite List

Erika, Darnell, and George are on the guest list for an exclusive club.

To gain entry to the club, Erika identifies herself to the person guarding the door.

- | | |
|--|------------------------|
| 1. Erika says $\langle \text{enter, dining room} \rangle$ | Erika's request |
| 2. Manager controls (Erika controls $\langle \text{enter, dining room} \rangle$) | Access policy |
| 3. $\text{ACL} \Rightarrow \text{Manager}$ | Trust assumption |
| 4. ACL says $\left\{ \begin{array}{l} \text{Erika controls } \langle \text{enter, dining room} \rangle \wedge \\ \text{Darnell controls } \langle \text{enter, dining room} \rangle \wedge \\ \text{George controls } \langle \text{enter, dining room} \rangle \end{array} \right.$ | Guest List |
| 5. ACL says (Erika controls $\langle \text{enter, dining room} \rangle$) | 4 simplify says |
| 6. $\langle \text{enter, dining room} \rangle$ | 1, 2, 3, 5 ticket rule |

Ticket Rule

- The inference rule

$$\text{Ticket Rule} \frac{\begin{array}{l} \text{subject says } \varphi \quad \text{authority controls (subject controls } \varphi) \\ \text{ticket} \Rightarrow \text{authority} \quad \text{ticket says (subject controls } \varphi) \end{array}}{\varphi}$$

- Hypotheses used are:

<i>subject</i> says φ	Access request
<i>authority</i> controls (<i>subject</i> controls φ)	Access policy
<i>ticket</i> \Rightarrow <i>authority</i>	Trust assumption
<i>ticket</i> says (<i>subject</i> controls φ)	Ticket

The Says Simplification Rules

(Two derived inference rules taken from figure 3.5)

Says
Simplification (1)

$$\frac{P \text{ says } (\varphi_1 \wedge \varphi_2)}{P \text{ says } \varphi_1}$$

conjunction

Says
Simplification (2)

$$\frac{P \text{ says } (\varphi_1 \wedge \varphi_2)}{P \text{ says } \varphi_2}$$

Access-Control Mechanisms: Lists

The End

Discretionary Security Policies

Access-Control Matrix and Access-Control Lists

The Background of the C-I-A Triad

Consider your personal data (e.g., grade related) in our university's computer system.

1. Unauthorized individuals cannot access it.
2. The data come from a trusted source and are uncorrupted.
3. The data are available to authorized individuals or parties.

Confidentiality, Integrity, and Availability

Policies

- Specify what is *allowed* and what is *prohibited*
- Specify permissible *actions* that *subjects* can perform on *objects*

Confidentiality

- Deals with the *inaccessibility* of information
- Policy states who may know or possess information

Confidentiality, Integrity, and Availability, continued

Integrity

- Deals with *accuracy, credibility, or quality* of information or resource
- Gasoline comes in various grades: 87, 89, or 91 octane
- Bonds are rated from AAA (best) to C (junk)

Availability


- Deals with *quality of service*
- Bandwidth or speed of internet connections
- Long or short lines for boarding planes at airports based on ticket class or mileage levels

Access-Control Matrix

- Represents (desired) access-control behavior of a system
- Describes basic access-control information of a system; not sufficient as authentication; trust assumptions are not addressed

- Example

- ACST: Table 4.1



	file ₁	file ₂	file ₃	file ₄
alice	read	read, write		execute
bob	⋮	read	execute	
carol	read, write		execute	execute

Table 4.1: Example of an access-control matrix

Access-Control Matrix M

- Subjects (i.e., principals): alice, bob, carol
- Objects: file₁, file₂, file₃, file₄ (protected objects)
- An entry of M, $M(s, o)$, specifies the access rights subject s with respect to object o

		Objects			
		file ₁	file ₂	file ₃	file ₄
Subjects	alice	read	read, write		execute
	bob		read	execute	
	carol	read, write		execute	execute

Handwritten annotations: "Access rights" with an arrow pointing to the "execute" cell for (alice, file₄); "General form" with an arrow pointing to the same cell.

Table 4.1: Example of an access-control matrix

execute (alice, file₄)

Access-Control Lists

- In table 4.1, the access-control information of each object o can be obtained from the respective column and can be represented as a list (ACL: an access-control list).
- By contrast, the capability of a subject s to access the system's resource can be obtained from the respective row in the access-control matrix.
- The access-control information stored in the list can be expressed in ACL logic.

e.g.

alice

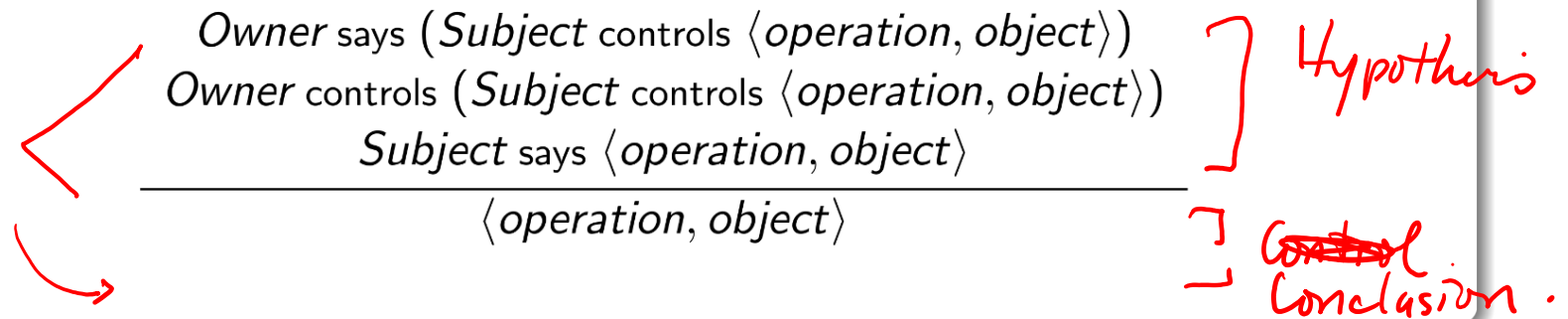
 ← row →
↑
Capability
list

Discretionary Security Policies

Dynamic: can change (typically under user control)

IBAC: Identity-based access control

- Decisions made on subject's identity
- Owner usually has control



Example

Access-control matrix for *foo* under Susan's control

	<i>foo</i>
April	read
Bill	read, write
Carla	execute

object
foo

Formalization of ACL

- $1 \wedge 2 \wedge 3 \wedge 4$ {
- 1 Susan says (*April* controls $\langle \text{read}, \text{foo} \rangle$) \wedge
 - 2 Susan says (*Bill* controls $\langle \text{read}, \text{foo} \rangle$) \wedge
 - 3 Susan says (*Bill* controls $\langle \text{write}, \text{foo} \rangle$) \wedge
 - 4 Susan says (*Carla* controls $\langle \text{execute}, \text{foo} \rangle$)

Example, continued

Susan's authority to determine the ACL

Susan controls (*April* controls $\langle \text{read}, \text{foo} \rangle$) \wedge
Susan controls (*Bill* controls $\langle \text{read}, \text{foo} \rangle$) \wedge
Susan controls (*Bill* controls $\langle \text{write}, \text{foo} \rangle$) \wedge
Susan controls (*Carla* controls $\langle \text{execute}, \text{foo} \rangle$)

Justification for granting Carla's execute request

Susan says (*Carla* controls $\langle \text{execute}, \text{foo} \rangle$)
Susan controls (*Carla* controls $\langle \text{execute}, \text{foo} \rangle$)
Carla says $\langle \text{execute}, \text{foo} \rangle$

 $\langle \text{execute}, \text{foo} \rangle$

e.g. -
of
Justification
for Carla request

Demonstration

The inference rule pertaining to Carla's request is correct.

Principal says (Subject controls $\langle \text{operation}, \text{object} \rangle$)

Principal controls (- - - - -)

Subject says $\langle \text{operation}, \text{object} \rangle$

$\langle \text{operation}, \text{object} \rangle$

Discretionary Security Policies

The End

Vulnerability: A Trojan Horse Example

An Extended Example

Background

Trojan horse

The Trojan horse used in the film *Troy* (Çanakkale, Turkey): It is made of plywood—by Fredrik Posse (from Wikipedia)

The Trojan horse story is a well-known story regarding ancient Greek history (e.g., *The Odyssey*).

The Greek soldiers were able to take the city of Troy after a fruitless 10-year siege by hiding in a giant horse supposedly left as an offering to the goddess Athena.



Trojan Horse

- Within the context of computer security, what is a Trojan horse?
 - It is a computer program with an apparently useful function, but it also has hidden functions that exploit the legitimate authorizations of the invoking process.
 - Viruses are usually transmitted as Trojan horses.
 - How does a Trojan horse causes harm to a system?

Trojan Horse (cont.)

- How does a Trojan horse cause harm to a computer system?
- Through the improper use of any authorizations of the invoking user; for example:
 - It could delete all files of the user.
 - It could obtain confidential information from the user without proper consent.

Discretionary Access Control

- Some observations
 - Users versus subjects operating on their behalf
 - No external control on the flow of information
- A subject operating on behalf of a user may carry out malicious activities that leak information, as there is no external control

An Example

Background

- Company: Troy, a manufacturing company
- Two users: Alice (manager) and Bob (assistant to Alice)
- File secret (owned by Alice): contains the company confidential information
- File apps (owned by Alice, Bob has “write” permission): an application that Alice uses for daily routines

An Example (cont.)

Access-control policies are specified by the matrix M.

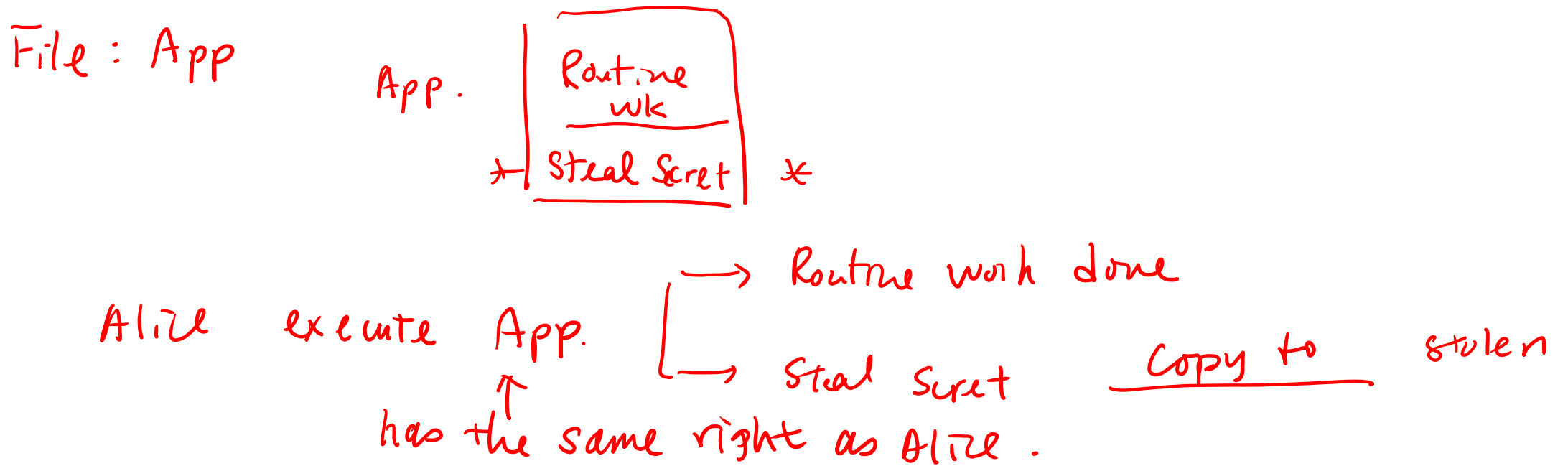
Subjects	File: secret	File: apps	File: stolen
Alice	read, write, execute	execute	
Bob		write	read, write

Access-control matrix M

Discussions

Bob steals the secret file from Alice. Outline the steps.

- What does Bob do?
- When will the file be stolen?



How to Avoid it ?

Discussions (cont.)

Can we avoid the information leak shown in this example?

- Why won't the discretionary policy help to avoid it (information leaks)?
- What are the changes that we can recommend (and why)?

Can we introduce external control ?

Vulnerability: A Trojan Horse Example

The End

Mandatory Security Policies

Security Policies Classifications

Security Policy Classifications

- Based on their nature: discretionary or mandatory
- Based on their use context: military or commercial
- Based on which aspect of security is addressed: confidentiality, integrity, or availability

Mandatory Security Policies

- Mandatory security policies
 - Policies that apply to everyone and everything all the time
 - Static and cannot be changed
 - Individuals have no discretion or control over them
- Mandatory access control (MAC) policies in computers
 - Typically implemented by the OS or by the hardware
 - Typically do not name specific subjects or principals in policy statements

Mandatory Security Policies (cont.)

- A hardware example (omitted, optional reading)
- Military security policies (protect confidentiality)
- Commercial policies (protect integrity)

Military Security Policies

Primary concern: *confidentiality*

- Information is protected on a *need to know* basis
- Flow of information is governed by classification levels, typically
 - UNCLASSIFIED, CONFIDENTIAL, SECRET, and TOP SECRET
 - $UC \leq_s C \leq_s S \leq_s TS$

Bell La Padula Model

- Subjects cannot read information at higher classification levels: “*no read up*”
- Subjects cannot write (leak) information to lower classification levels: “*no write down*”

Commercial Policies

Primary concern: *integrity*

- Protecting system and its resources from damage
 - Contamination
 - Corruption
 - Misuse

Integrity Levels: Examples

- Octane ratings on gas
- Frequent-flyer status: early access to seats and airport lounges

Maintaining quality rather than confidentiality

Mandatory Security Policies: Security Policies Classifications

The End

Mandatory Security Policies

Policy Specification with Access-Control (AC) Logic

Security Levels

Security (clearance) levels

- Each principal and security label is assigned to a secure level
- Example
 - Unclassified, confidential, secret, top secret
- Partially ordered

Kripke Structures with Security Levels

Security (clearance) levels

- Principal and security labels are assigned to a security level
- Examples of security labels used to define security levels
 - Unclassified, confidential, secret, top secret
- Security levels can be compared (i.e., ordered)
- Extends our Kripke structures to incorporate security levels

Extending a Kripke Structure

Syntax—need to extend the language to:

- Describe and compare security levels.
- Express the security level assigned to a particular principal.
- Define a new well-formed AC formula to support the comparisons of security levels.

Adding Security Levels to Kripke Semantics

Syntax

$\text{SecLevel} ::= \text{SecLabel} / \text{sl}(\text{PName})$

$\text{Form} ::= \text{SecLevel} \leq_s \text{SecLevel} / \text{SecLevel} =_s \text{SecLevel}$

Semantics

- $\mathcal{M} = \langle W, I, J, K, L, \preceq \rangle$
- W, I, J as before
- K is a non-empty set of security levels.
- $L : (\text{SecLabel} \cup \text{PName}) \rightarrow K$ is a mapping of security labels and simple principal names to an security level.

for every simple principal name A .

- $\preceq \subseteq K \times K$ is a partial order on K

relation

W, I, J

↳ Same as before

K, L, \leq

↳ new additions

$$L(\text{sl}(A)) = L(A),$$

Extending a Kripke Structure

Additional features in the extended models

- K : a set of security levels
- L : a labeling function
- \leq : an ordering relation over K ($\subseteq K \times K$)

Example

PName	Amy (ts), Biao (s), Sonja (c)
SecLabels	ts, s, c
K	$\{k_1, k_2, k_3, \cancel{k_4}\}$ $"\cancel{c} \leq s \leq ts"$ $\cancel{k_1} \quad k_2 \quad k_3 \quad k_4$
L	$L(c) = k_1 \quad L(s) = k_2 \quad L(ts) = k_3$ $L(Amy) = k_3 \quad L(Biao) = k_2 \quad L(Sonja) = k_1$
\preceq	$k_1 \leq k_2 \leq k_3$

Extending a Kripke Structure

Semantics—need to assign precise meanings to the newly added syntax by:

- Extending the definition of the evaluation function E_μ
- Specifying adequate properties regarding comparisons of security levels
Add
- Formulating new inference rules to support reasoning about access requests
↑ Any inference rules for reasoning with security levels.

Kripke Semantics and Inference Rules

Kripke Semantics

$$\begin{aligned} \mathcal{E}_{\mathcal{M}}[\![l_1 \leq_s l_2]\!] &= \begin{cases} W, & \text{if } L(l_1) \preceq L(l_2) \\ \emptyset, & \text{otherwise} \end{cases} \\ \mathcal{E}_{\mathcal{M}}[\![l_1 =_s l_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![l_1 \leq_s l_2]\!] \cap \mathcal{E}_{\mathcal{M}}[\![l_2 \leq_s l_1]\!]. \end{aligned}$$

Assign
meanings

Inference Rules

$$l_1 =_s l_2 \stackrel{\text{def}}{=} (l_1 \leq_s l_2) \wedge (l_2 \leq_s l_1)$$

$$\text{Reflexivity of } \leq_s \quad \frac{}{l \leq_s l}$$

$$\text{Transitivity of } \leq_s \quad \frac{l_1 \leq_s l_2 \quad l_2 \leq_s l_3}{l_1 \leq_s l_3}$$

$$sl \leq_s \quad \frac{sl(P) =_s l_1 \quad sl(Q) =_s l_2 \quad l_1 \leq_s l_2}{sl(P) \leq_s sl(Q)}$$

to new
type
of formulas

rules
& properties
that \leq_s
(relation)

Mandatory Security Policies: Policy Specification with Access-Control (AC) Logic

The End

Bell-La Padula Model

Background

Bell-La Padula model

- Military security policy: managing information flow
- Primarily concern: protecting confidentiality
- Describes the conditions under which read access can be granted
- Describes the conditions under which write access can be granted

Bell-La Padula Model

Observations

- Includes both mandatory and discretionary components
- Expresses confidentiality properties with respect to discretionary access and security levels
- The extended Kripke structure introduced earlier can be used to support the reasoning regarding read and write access

Bell-La Padula Model

The End

Bell-La Padula Model

Specifying Bell-La Padula Policies

Adding Security Levels to Kripke Semantics

Syntax

SecLevel ::= **SecLabel** / **sl(PName)**

Form ::= **SecLevel** \leq_s **SecLevel** / **SecLevel** $=_s$ **SecLevel**

Semantics

- $\mathcal{M} = \langle W, I, J, K, L, \preceq \rangle$
- W, I, J as before
- K is a non-empty set of *security levels*.
- $L : (\mathbf{SecLabel} \cup \mathbf{PName}) \rightarrow K$ is a mapping of security labels and simple principal names to a security level.

$$L(\text{sl}(A)) = L(A),$$

for every simple principal name A .

- $\preceq \subseteq K \times K$ is a partial order on K

Kripke Semantics and Inference Rules

Kripke Semantics



$$\begin{aligned}\mathcal{E}_{\mathcal{M}}[\![\ell_1 \leq_s \ell_2]\!] &= \begin{cases} W, & \text{if } L(\ell_1) \preceq L(\ell_2) \\ \emptyset, & \text{otherwise} \end{cases} \\ \mathcal{E}_{\mathcal{M}}[\![\ell_1 =_s \ell_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\ell_1 \leq_s \ell_2]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\ell_2 \leq_s \ell_1]\!].\end{aligned}$$

Inference Rules

$$\ell_1 =_s \ell_2 \stackrel{\text{def}}{=} (\ell_1 \leq_s \ell_2) \wedge (\ell_2 \leq_s \ell_1)$$

$$\text{Reflexivity of } \leq_s \quad \frac{}{\ell \leq_s \ell}$$

$$\text{Transitivity of } \leq_s \quad \frac{\ell_1 \leq_s \ell_2 \quad \ell_2 \leq_s \ell_3}{\ell_1 \leq_s \ell_3}$$

$$sl \leq_s \quad \frac{sl(P) =_s \ell_1 \quad sl(Q) =_s \ell_2 \quad \ell_1 \leq_s \ell_2}{sl(P) \leq_s sl(Q)}$$

Bell La Padula Policy

Simple Security Condition

Principal P can read object O if and only if:

1. P 's security level is at least as high as O 's (i.e., $sl(O) \leq_s sl(P)$), and
2. P has discretionary read access to O (i.e., P controls $\langle read, O \rangle$).

$$(sl(O) \leq_s sl(P)) \supset (P \text{ controls } \langle read, O \rangle)$$

Cond'n
in ACL

"No read up"

*-Property

Principal P can write to object O if and only if:

1. O 's security level is at least as high as P 's (i.e., $sl(P) \leq_s sl(O)$), and
2. P has discretionary write access to O (i.e., P controls $\langle write, O \rangle$).

$$(sl(P) \leq_s sl(O)) \supset (P \text{ controls } \langle write, O \rangle)$$

"No write down"

Bell-La Padula Policy

Remarks

- Catch phrase: “*no read up and no write down*”
- Both the *simple security* condition and the **-property* are “*if and only if*” statements
- Note that the *if* direction is expressible in AC logic, but the *only-if* direction is not

The Conditional Controls Rule

The intended behavior described by both simple security conditions and *-property can be captured by the derived inference rule:

$$\text{Conditional Controls} \quad \frac{\begin{array}{c} \mu \supset P \text{ controls } \varphi \\ \mu \\ P \text{ says } \varphi \end{array}}{\varphi} .$$

FX: A Fictional Example

- (ACST, Section. 5.4.3) The defense contractor example
- Contractor: DefenseSystemsRUs
- It has defense contracts across several branches of the military; it has separate contracts with both the Air Force and the Navy to develop air superiority fighters; the Air Force-funded project is called the FX-1, while the Navy-funded project is called the FX-2

FX-1 and FX-2 Project

Documents classification levels

Document	Classification
threat scenario	TS
status report	TS
requirements	S
design	S
artist renderings	C
press releases	UC

classification
for each type
of document

FX-1 and FX-2 project personnel, functions, and clearances

Function	Clearance	FX-1	FX-2
Team Leader	TS	Amy	Arlen
Engineer	S	Biao	Burt
Artist	C	Sonja	Suresh
Public Relations	UC	Jude	Jodi

clearance level
for the personnel

Examples

FX-1 Access Matrix

FX-1 Document	Amy (ts)	Biao (s)	Sonja (c)	Jude (uc)
threat scenario (TS)	read			
status report (TS)	read, write	write	write	write
requirements (S)	read	read		
design (S)	read	read, write		
artist renderings (C)	read	read	read, write	
press releases (UC)	read	read	read	read, write

security
label
for the
personnel
included

Security policy

newly defined relation

in
ACL

$sl(status_{FX1}) \leq_s sl(Amy) \supset (Amy \text{ controls read, } status_{FX1})$
 $sl(Amy) \leq_s sl(status_{FX1}) \supset (Amy \text{ controls write, } status_{FX1})$
 $sl(Biao) \leq_s sl(status_{FX1}) \supset (Biao \text{ controls write, } status_{FX1})$
 $sl(Sonja) \leq_s sl(status_{FX1}) \supset (Sonja \text{ controls write, } status_{FX1})$
 $sl(Jude) \leq_s sl(status_{FX1}) \supset (Jude \text{ controls write, } status_{FX1})$

Bell-La Padula Model: Specifying Bell-La Padula Policies

The End

Bell-La Padula Model with AC Logic

An Extended Example

FX-1 and FX-2 Project

Documents classification levels

Document	Classification
threat scenario	TS
status report	TS
requirements	S
design	S
artist renderings	C
press releases	UC

$UC \leq C \leq S \leq TS$

FX-1 and FX-2 project personnel, functions, and clearances

Function	Clearance	FX-1	FX-2
Team Leader	TS	Amy	Arlen
Engineer	S	Biao	Burt
Artist	C	Sonja	Suresh
Public Relations	UC	Jude	Jodi



Examples

"No read up" = No violations

FX-1 Access Matrix

FX-1 Document	Amy (ts)	Biao (s)	Sonja (c)	Jude (uc)
threat scenario (TS)	read			
status report (TS)	read, write	write	write	write
requirements (S)	read	read		
design (S)	read	read, write		
artist renderings (C)	read	read	read, write	
press releases (UC)	read	read	read	read, write

"No" Write
up = No
down violations

Security policy

- $sl(status_{FX1}) \leq_s sl(Amy) \supset (Amy \text{ controls read, } status_{FX1})$
- $sl(Amy) \leq_s sl(status_{FX1}) \supset (Amy \text{ controls write, } status_{FX1})$
- $sl(Biao) \leq_s sl(status_{FX1}) \supset (Biao \text{ controls write, } status_{FX1})$
- $sl(Sonja) \leq_s sl(status_{FX1}) \supset (Sonja \text{ controls write, } status_{FX1})$
- $sl(Jude) \leq_s sl(status_{FX1}) \supset (Jude \text{ controls write, } status_{FX1})$

revised
language of A.C.
Logic

FX Example

Discussions

Verify if Bell La Padula conditions

(" No read up" , " No write down")

are satisfied . }

FX Example (cont.)

Discussions (cont.)

Formation using A.C. Logic "Language".

Write simple conditions

Q Interpret policies written in that language.

Bell-La Padula Model with AC Logic

The End

Weekly Summary

Applying Formal Methods III

Security Mechanisms and Policies

- A secured system provides precise access-control policies and have adequate mechanisms in place to enforce the policies.
- Discretionary policies can be formulated based on an access-control matrix model. They state explicit access rules with respect to the identity of the requestor to establish who can, or cannot, execute which actions on which resources.

Security Mechanisms and Policies (cont.)

- Security policies can be classified in many ways.
- The Bell-La Padula model is a notable example for military security policies. It describes policies that govern the information flow of the system.
- Policies can be formulated and expressed by an extension of access-control logic (given in chapter 2) to protect information confidentiality. It can be applied to Bell-La Padula models.

Weekly Summary

The End