

Security Principles and Policies

CS 239

Computer Security

Peter Reiher
January 11, 2006

CS 239, Winter 2006

Lecture 2
Page 1

Outline

- Security terms and concepts
- Security policies
 - Basic concepts
 - Security policies for real systems

CS 239, Winter 2006

Lecture 2
Page 2

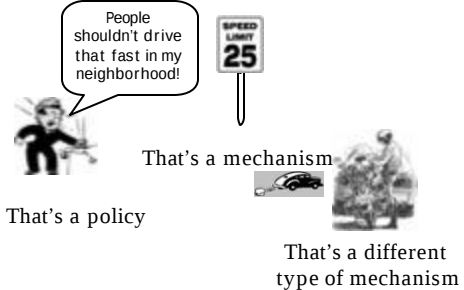
Security and Protection

- *Security* is a policy
 - E.g., “no unauthorized user may access this file”
- *Protection* is a mechanism
 - E.g., “the system checks user identity against access permissions”
- Protection mechanisms implement security policies

CS 239, Winter 2006

Lecture 2
Page 3

Policy vs. Mechanism



CS 239, Winter 2006

Lecture 2
Page 4

Design Principles for Secure Systems

- Economy
- Complete mediation
- Open design
- Separation of privileges
- Least privilege
- Least common mechanism
- Acceptability
- Fail-safe defaults

CS 239, Winter 2006

Lecture 2
Page 5

Economy in Security Design

- Economical to develop
 - And to use
 - And to verify
- Should add little or no overhead
- Should do only what needs to be done
- Generally, try to keep it simple and small

CS 239, Winter 2006

Lecture 2
Page 6

Complete Mediation

- Apply security on every access to a protected object
 - E.g., each read of a file, not just the open
- Also involves checking access on everything that could be attacked

CS 239, Winter 2006

Lecture 2
Page 7

Open Design

- Don't rely on "security through obscurity"
- Assume all potential attackers know everything about the design
 - And completely understand it
- This doesn't mean publish everything important about your security system
 - Though sometimes that's a good idea

CS 239, Winter 2006

Lecture 2
Page 8

Separation of Privileges

- Provide mechanisms that separate the privileges used for one purpose from those used for another
- To allow flexibility in security systems
- E.g., separate access control on each file

CS 239, Winter 2006

Lecture 2
Page 9

Least Privilege

- Give bare minimum access rights required to complete a task
- Require another request to perform another type of access
- E.g., don't give write permission to a file if the program only asked for read

CS 239, Winter 2006

Lecture 2
Page 10

Least Common Mechanism

- Avoid sharing parts of the security mechanism
 - among different users
 - among different parts of the system
- Coupling leads to possible security breaches

CS 239, Winter 2006

Lecture 2
Page 11

Acceptability

- Mechanism must be simple to use
- Simple enough that people will use it without thinking about it
- Must rarely or never prevent permissible accesses

CS 239, Winter 2006

Lecture 2
Page 12

Fail-Safe Designs

- Default to lack of access
- So if something goes wrong or is forgotten or isn't done, no security lost
- If important mistakes are made, you'll find out about them
 - Without loss of security
 - But if it happens too often . . .

CS 239, Winter 2006

Lecture 2
Page 13

Thinking About Security

When considering the security of any system, ask these questions:

1. What assets are you trying to protect?
2. What are the risks to those assets?
3. How well does the security solution mitigate those risks?
4. What other security problems does the security solution cause?
5. What tradeoffs does the security solution require?

(This set of questions was developed by Bruce Schneier, for his book *Beyond Fear*)

CS 239, Winter 2006

Lecture 2
Page 14

An Example

- Access to computers in the graduate workstation room
- Current security solution
 - Must provide valid CS department user ID and password

CS 239, Winter 2006

Lecture 2
Page 15

What Assets Are We Trying to Protect?

- ?

CS 239, Winter 2006

Lecture 2
Page 16

What Are the Risks to Those Assets?

- ?

CS 239, Winter 2006

Lecture 2
Page 17

How Well Does the Security Solution Mitigate Those Risks?

- ?

CS 239, Winter 2006

Lecture 2
Page 18

What Other Security Problems Does the Security Solution Cause?

- ?

CS 239, Winter 2006

Lecture 2
Page 19

What Tradeoffs Does the Security Solution Require?

- ?

CS 239, Winter 2006

Lecture 2
Page 20

Security Policies

- Security policies describe how a secure system should behave
- Generally, if you don't have a clear policy, you don't have a secure system
 - Since you don't really know what you're trying to do

CS 239, Winter 2006

Lecture 2
Page 21

What Is a Security Policy?

- A complete description of the security goals the system should achieve
 - Not a description of how to achieve them
- Sometimes described informally
- Sometimes described very formally
 - Using mathematical models

CS 239, Winter 2006

Lecture 2
Page 22

Informal Security Policies

- “Users should only be able to access their own files, in most cases.”
- “Only authorized users should be able to log in.”
- “System executables should only be altered by system administrators.”
- The general idea is pretty clear
- But it can be hard to determine if a system meets these goals

CS 239, Winter 2006

Lecture 2
Page 23

Access Control Policies

- Describe who can access what resources
- *Mandatory access control*
 - The system enforces its own policy
- *Discretionary access control*
 - Policy set by individual users

CS 239, Winter 2006

Lecture 2
Page 24

Formal Security Policies

- Typically expressed in a mathematical security policy language
- Tending towards precision
 - Allowing formal reasoning about the system and policy
- Often matched to a particular policy model
 - E.g., Bell-La Padula model

CS 239, Winter 2006

Lecture 2
Page 25

Bell-La Padula Model

- Probably best-known computer security model
- Corresponds to military classifications
- Combines mandatory and discretionary access control
- Two parts:
 - Clearances
 - Classifications

CS 239, Winter 2006

Lecture 2
Page 26

Clearances

- Subjects (people, programs, etc.) have a *clearance*
- Clearance describes how trusted the subject is
- E.g., *unclassified*, *confidential*, *secret*, *top secret*

CS 239, Winter 2006

Lecture 2
Page 27

Classifications

- Each object (file, database entry, etc.) has a *classification*
- The classification describes how sensitive the object is
- Using same categories as clearances
- Informally, only people with the same (or higher) clearance should be able to access objects of a particular classification

CS 239, Winter 2006

Lecture 2
Page 28

Goal of Bell-LaPadula Model

- Prevent any subject from ever getting read access to objects at higher classification levels than subject's clearance
- Concerned not just with objects
- Also concerned with the objects' contents
- Includes discretionary access control
 - Which we won't cover in lecture

CS 239, Winter 2006

Lecture 2
Page 29

Bell-LaPadula Simple Security Condition

- *Subject S can read object O iff $l_O = l_S$*
- Simple enough:
 - If S isn't granted top secret clearance, S can't read top secret objects
- Are we done?

CS 239, Winter 2006

Lecture 2
Page 30

Why Aren't We Done?

- Remember, we really care about the information in an object
- A subject with top secret clearance can read a top secret object
- If careless, he could write that information to a confidential object
- Then someone with confidential clearance can read top secret information

CS 239, Winter 2006

Lecture 2
Page 31

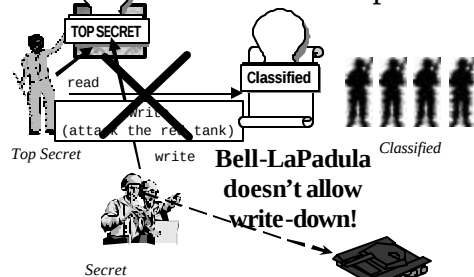
The Bell-LaPadula *-Property

- S can write O iff $I_S = I_O$
- Prevents *write-down*
 - Privileged subjects writing high-classification information to low-classification objects
 - E.g., a top secret user can't write to a confidential data file
- Can be proven that a system meeting these properties is "secure"

CS 239, Winter 2006

Lecture 2
Page 32

Bell-LaPadula Example



CS 239, Winter 2006

Lecture 2
Page 33

Bell-LaPadula Caveats

- A provably secure Bell-LaPadula system may be impossible to really use
- Says nothing about some other important security properties
 - Like integrity

CS 239, Winter 2006

Lecture 2
Page 34

Integrity Security Policies

- Designed to ensure that information is not improperly changed
- Often the key issue for commercial systems
- Secrecy is nice, but not losing track of your inventory is crucial

CS 239, Winter 2006

Lecture 2
Page 35

Example: Biba Integrity Policy

- Subject set S , object set O
- Set of ordered integrity levels I
- Subjects and objects have integrity levels
- Subjects at high integrity levels are less likely to screw up data
 - E.g., trusted users or carefully audited programs
- Data at a high integrity level is less likely to be screwed up
 - Probably because it badly needs not to be screwed up

CS 239, Winter 2006

Lecture 2
Page 36

Biba Integrity Policy Rules

- s can write to o iff $i(o) = i(s)$
- s_1 can execute s_2 iff $i(s_2) = i(s_1)$
- A subject s can read object o iff $i(s) = i(o)$
- Why do we need the read rule?

CS 239, Winter 2006

Lecture 2
Page 37

Hybrid Models

- Sometimes the issue is keeping things carefully separated
- E.g., a brokerage that handles accounts for several competing businesses
- Microsoft might not like the same analyst working on their account and IBM's
- There are issues of both confidentiality and integrity here

CS 239, Winter 2006

Lecture 2
Page 38

The Chinese Wall Model

- If followed, meant to keep things that should be separated apart
- Objects O are items of information related to a company
- A company dataset CD contains all of a company's objects
- A conflict-of-interest class COI contains the datasets of companies in competition – I.e., the things needing to be kept apart

CS 239, Winter 2006

Lecture 2
Page 39

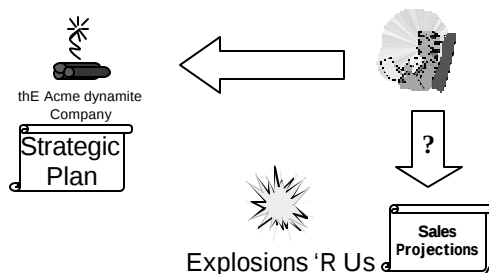
Chinese Wall Security Conditions

- S can read O iff any of the following holds:
 1. There is an object O' that S has accessed and $CD(O) = CD(O')$
 2. For all objects $O' \in PR(S)$, $COI(O') \cap COI(O) = \emptyset$ ($PR(S)$ is the set of objects S has already read)
 3. O is a sanitized object
 - While O may be in a forbidden CD for S , anything sensitive has been removed

CS 239, Winter 2006

Lecture 2
Page 40

Chinese Wall Example

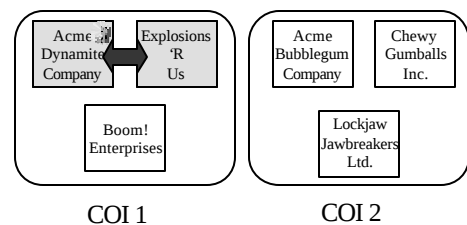


CS 239, Winter 2006

Lecture 2
Page 41

Should This Be Allowed?

This access violates CW rule 2



CS 239, Winter 2006

Lecture 2
Page 42

Designing a Policy for an Installation

- Need to determine what security goals your system has
 - Everything you mandate in the policy will have a cost
- Try to specify the minimal restrictions you really need
- But think broadly about what is important to you

CS 239, Winter 2006

Lecture 2
Page 43

For Example,

- Consider the UCLA Computer Science Department facility
- Provides computing and networking services to all faculty, staff, grad students
- Does not support undergrads
- Equipment located on 3^d and 4th floors of Boelter Hall

CS 239, Winter 2006

Lecture 2
Page 44

Services Offered by CS Facility

- Storage and compute facilities
- E-mail
- General network access (e.g., web browsing), including wireless
- Web server and department web pages
- Support for some grad class labs

CS 239, Winter 2006

Lecture 2
Page 45

What Do People Use Facility For?

- Classwork
 - Both students and professors
- Research support
- Departmental business
 - Some, not all
- Reasonable personal use

CS 239, Winter 2006

Lecture 2
Page 46

So, What Should the Department's Policy Be?

- ?

CS 239, Winter 2006

Lecture 2
Page 47

The Problems With Security Policies

- Hard to define properly
 - How do you determine what to allow and disallow?
- Hard to go from policy to mechanism that actually implements it
- Hard to understand implications of policy
- Defining and implementing policies is a lot of work

CS 239, Winter 2006

Lecture 2
Page 48

The Result?

- Security policies get a lot of lip service
- But an awful lot of places haven't actually got one
 - Even some very important places

CS 239, Winter 2006

Lecture 2
Page 49

How Policies Often Work in the Real World

- Your policy is what your tools allow by default
- Your policy is a vague version of what your sysadmin thinks is best
- Your policy is perhaps reasonably well defined, but not implemented by any real mechanisms

CS 239, Winter 2006

Lecture 2
Page 50