Security Principles and Policies
CS 136
Computer Security
Peter Reiher
January 15, 2008

Outline

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

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

Policy vs. Mechanism

That's a policy

That's a mechanism

That's a different type of mechanism

Trust

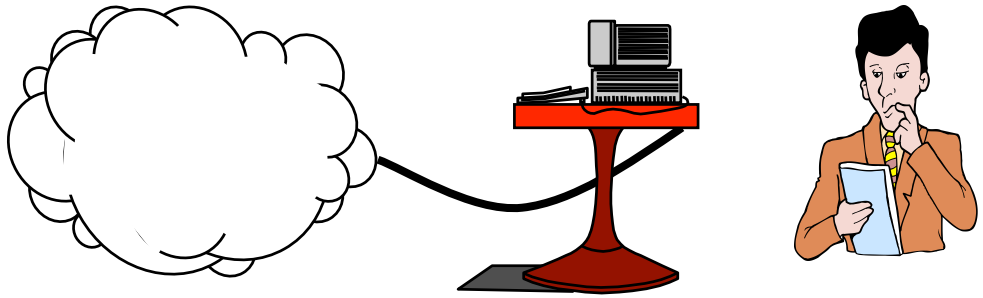
- An extremely important security concept
- You do certain things for those you trust
- You don't do them for those you don't
- Seems simple, but . . .

Problems With Trust

- How do you express trust?
- Why do you trust something?
- How can you be sure who you're dealing with?
- What if trust is situational?
- What if trust changes?

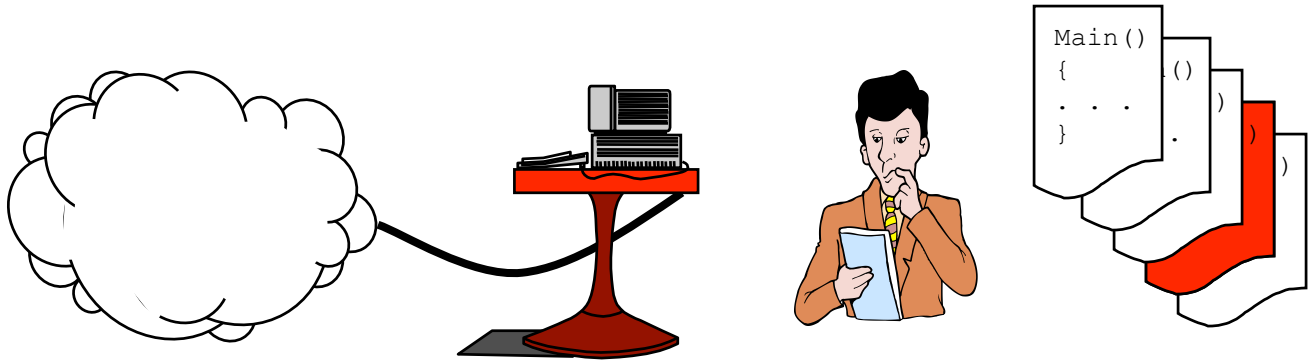
An Important Example

Consider a typical home computer



Let's say it only has one user
What's trust got to do with this case?
What if it connects to the Internet?
Do we treat everything out on the Internet as trusted?
If not, how do we tell what to trust?
And just how much?

Continuing Our Example



And what about the software it runs?

Is it all equally trusted?

If not, how do we determine exactly what each program should be allowed to do?

Trust Is Not a Theoretical Issue

- Most vulnerabilities that are actually exploited are based on trust problems
- Attackers exploit overly trusting elements of the computer
  - From the access control model to the actual human user
- Taking advantage of trust that shouldn't be there
- Such a ubiquitous problem that some aren't aware of its existence

A Trust Problem

- A user is fooled into downloading a Trojan horse program
- He runs it and it deletes a bunch of files on his computer
- Why was this program *trusted* at all?
- Why was it *trusted* to order the deletion of unrelated files?

Another Example of Trust Problems

- Phishing
- Based on fooling users into clicking on links leading to bad sites
- Usually resulting in theft of personal information
- Why does the user (or his computer) *trust* those sites?
  - Or the email message telling him to go there?

A Third Example of Trust Problems

- Buffer overflows
- A mechanism for taking control of a running program
- Allows attacker to tell the program to do things it wasn't designed to
- Why should that program be able to do arbitrary things, anyway?
- Why did we *trust* it to do things it didn't need to do?

Transitive Trust

I trust Alice

Alice trusts Bob

So do I trust Carol?

Should I?

Bob trusts David

David trusts Carol

Examples of Transitive Trust

- Trust systems in peer applications
- Chains of certificates
- But also less obvious things
  - Like a web server that calls a database
  - The database perhaps trusts the web server itself
  - But does it necessarily trust the user who invoked the server?
- Programs that call programs that call programs are important cases of transitive trust

Design Principles for Secure Systems

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

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

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

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
- Obscurity can provide *some* security, but it's brittle
  - When the fog is cleared, the security disappears

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

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

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

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

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 . . .

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*)

An Example

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

Think About the Questions

- What assets are we trying to protect?
- What are the risks to those assets?
- How well does the security solution mitigate those risks?
- What other security problems does the security solution cause?
- What tradeoffs does the security solution require?

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

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

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

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
- Most systems provide only discretionary access control

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

Some Important Security Policies

- Bell-La Padula
- Biba integrity policy
- Chinese Wall policy

Bell-La Padula Model

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

Clearances

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

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

Goal of Bell-La Padula Model

- Prevent any subject from ever getting read access to objects at higher classification levels than subject's clearance
  - I.e., don't let untrusted people see your secrets
- Concerned not just with objects
- Also concerned with the objects' contents
- Includes discretionary access control
  - Which we won't cover in lecture

Bell-La Padula Simple Security Condition

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

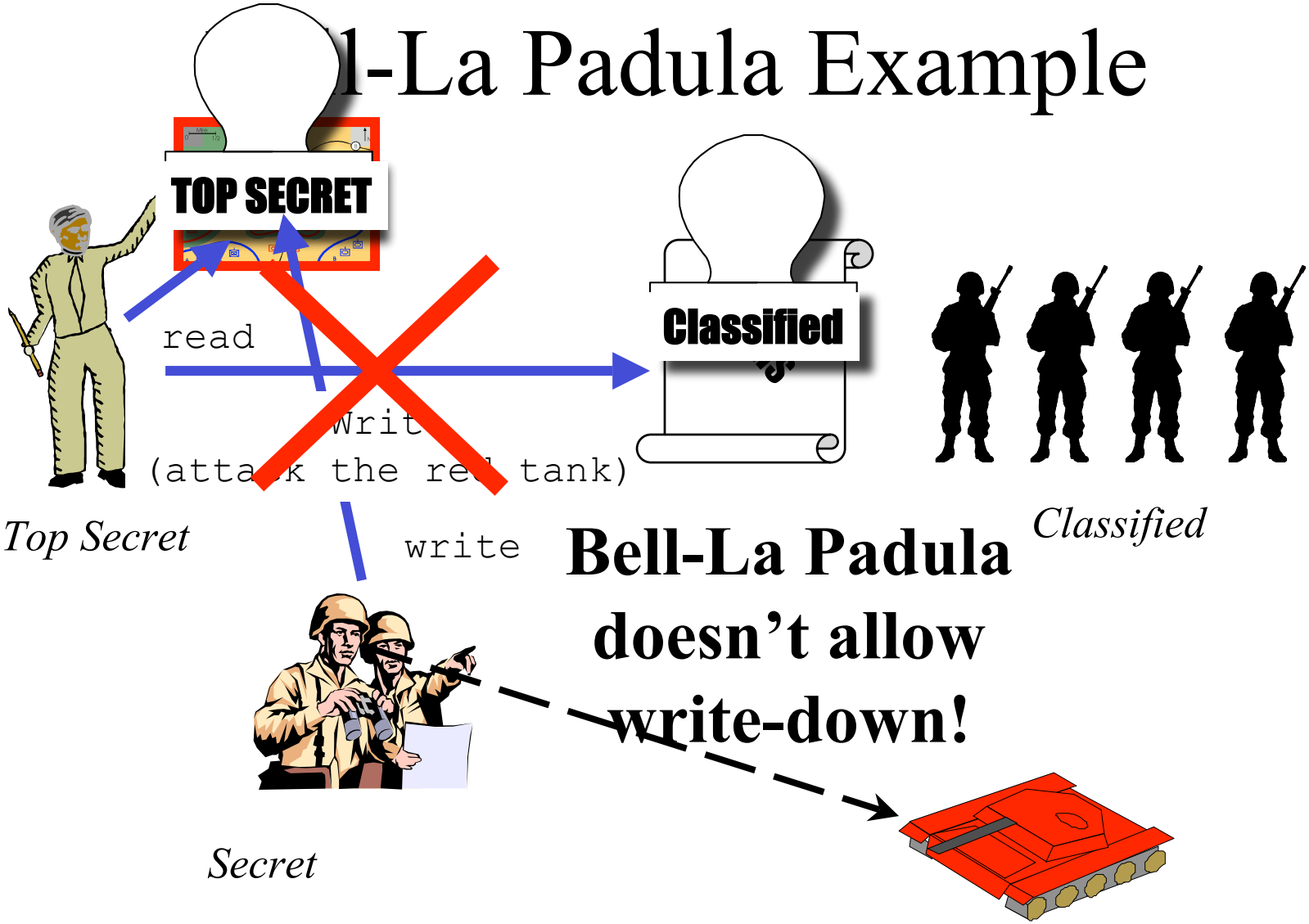
Why Aren't We Done?

- 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

The Bell-La Padula *-Property

- *S can write O iff* $l_S \leq l_O$
- 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"

l-La Padula Example



So How Do You Really Use The System?

- There have to be mechanisms for reclassification
- Typically, a document at a higher classification is set to a lower one
  - Usually requiring explicit operation
- Danger that reclassification process will be done incautiously

Bell-La Padula Caveats

- A provably secure Bell-La Padula system may be impossible to really use
- Says nothing about some other important security properties
  - Like integrity
- Information is generally put in different categories, in real use
  - Classifications and access permissions set separately on each category
  - "Need to know" principle

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

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

Biba Integrity Policy Rules

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

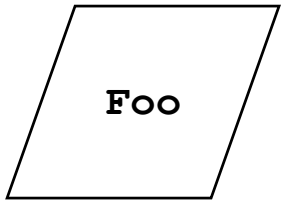
Vista and Mandatory Integrity Control

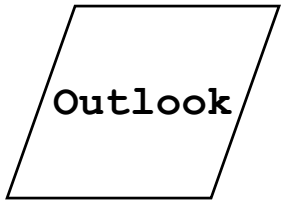
- A limited form of the Biba model in Microsoft's Vista OS
- Users have an access token with a security level
- Processes run by them run at that level
- Low-level processes can't write files marked with high integrity levels
- No read component to this access control

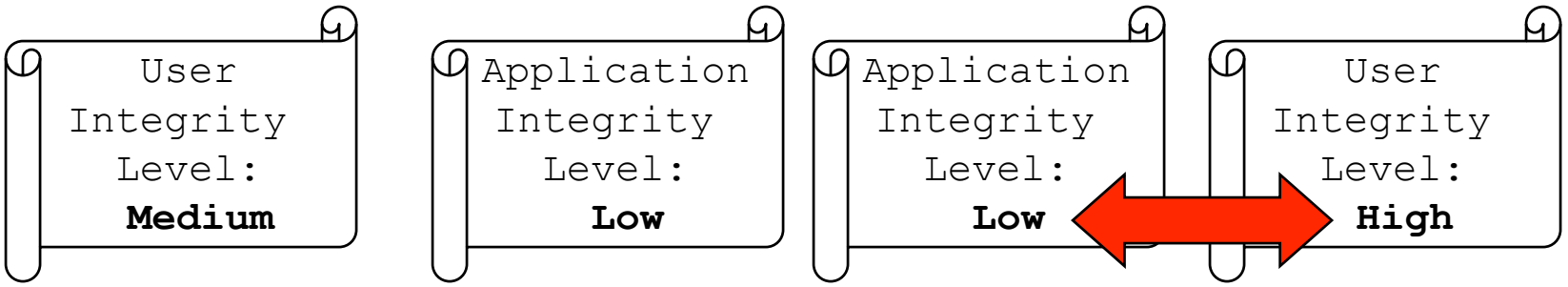
More Details on Vista MIC

- Five defined integrity levels
- Default is middle level, IE runs at next level down
- Objects created by processes inherit their level
- Can't write to files at higher integrity levels
- Failures lead to prompts asking if level should be elevated
  - Is that a good idea?
  - If not, what should they do instead?

An Example







The application downloads an executable foo

foo runs and tries to write to the Outlook executable

Vista MIC denies the write

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

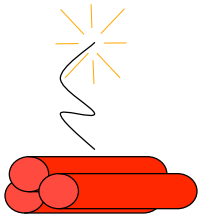
The Chinese Wall Model

- 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

Chinese Wall Security Conditions

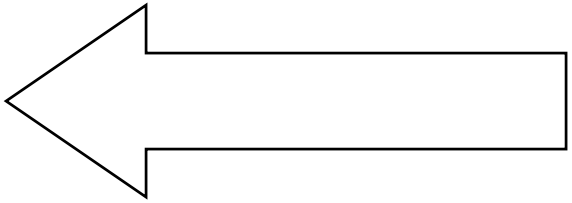
- *S* can read *O* iff any of the following holds:
  1. There is an object *O′* that *S* has accessed and *CD(O) = CD(O′)*
  2. For all objects *O′*, $O' \in PR(S) \Rightarrow COI(O') \neq COI(O)$ (*PR(S)* is the set of objects *S* has already read)
  3. *O* is a sanitized object
     - While *O* may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

Chinese Wall Example

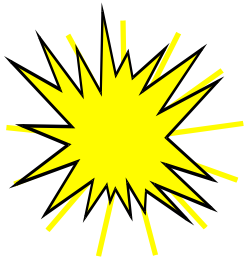


The Acme Dynamite Company

Strategic Plan



?

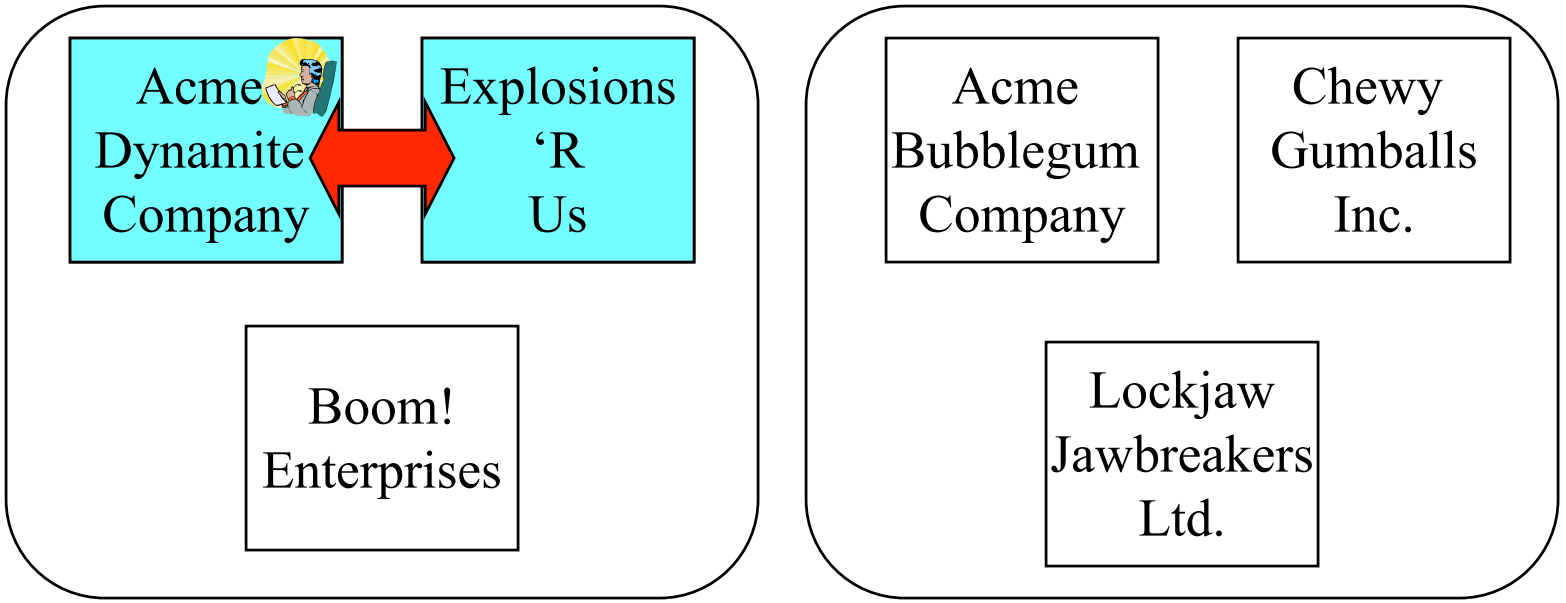


Explosions 'R Us

Sales Projections

Should This Be Allowed?

This access violates CW rule 2



What Policies Are Commonly Used?

- Most installations only use discretionary access control
- Offered by Windows, Linux, other widely used operating systems
- We'll discuss these forms of access control in more detail later

The Realities of Discretionary Access Control

- Most users never change the defaults on anything
  - Unless the defaults prevent them from doing something they want
- Most users don't think about or understand access control
- Probably not wise to rely on it to protect information you care about
  - Unless you're the one setting it
  - And you know what you're doing

Other Kinds of Policy

- Not all security policies are about access control
  - "You must keep logs of accesses"
  - "You must have a properly configured firewall"
  - "You must run a security audit every year"
  - "Every user must take a course educating him about viruses and phishing"
- Potentially very general
- Not as formally defined as access control
- But possibly even more important than access control policies

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

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

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

What Do People Use Facility For?

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

So, What Should the Department's Policy Be?

- ?

The Problems With Security Policies

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

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
- *"If you have a policy, and you don't enforce it, and you don't hold people accountable for violating it, then - do you have a policy?"* Marcus Ranum

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
- If you're in charge of security, though, treat your policy more seriously