UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 537
Introduction to Operating Systems

Andrea C. Arpaci-Dusseau
Remzi H. Arpaci-Dusseau

Protection and Security

Questions answered in this lecture:

How can a system authenticate a user?

How are access rights specified?

What are common security problems?

Motivation

Protection more important as computer systems develop

- Multiple users have access to same resources
- Computers connected to network
- Increasing importance to electronic commerce

Goals: Ensure users only do what they are supposed to do

- Prevent accidental misuse
  - Example: Mistakenly overwrite command interpreter; no one can log in
  - Relatively easy to solve by making hard to do
- Prevent malicious abuse
  - Example: Break into accounting system and transfer $3billion
  - Hard to completely eliminate

Components of Protection Mechanism

Authentication

- Make sure system knows which user is doing what action

Authorization determination

- Policy
- Determine what the user is and is not allowed to do

Access enforcement

- Mechanism
- Make sure no loopholes in the system

Flaw in any area ruins entire protection mechanism

- System is only as secure as its weakest link

Authentication

How do you prove who you are?

Passwords

- Secret piece of information known only by user
- System should not store in readable form
  - One-way transformations must be used when check
- Disadvantage: Relatively easy to crack
  - Humans choose poor passwords
    - Short passwords are easy to find with brute force
    - Common words found in dictionaries

Key

- Physical possession of item proves identity
- Should not be forgeable or able to be copied
- Advantage: If stolen, user is aware
- Disadvantage: Relatively expensive to make

Authorization Determination

Access rights represented with access matrix

- One domain (e.g., user) per row
- One resource (e.g., files) per column
- Each entry indicates privileges of domain for resource

	File A	File B	File C	File D
User 1	RW	RW	RW	RW
User 2	RW	RW	-	-
User 3	RW	R	-	-
User 4	RW	R	RW	-
User 5	RW	R	RW	-

Representation of Access Matrix

Access matrix is sparsely populated

- Condense information by expressing in two forms
  - Access control list: Per column
  - Capability: Per row

Access Control Lists: (ACLs)

- For each resource, indicate users that can perform operations
  - General form: Each resource has list of <user, privilege> pairs
- Disadvantage
  - Tedious to have separate entry for every user
  - Optimization: Group users into classes
  - UNIX example:
    - Three classes of users: self, group, everyone else
    - Three privileges: read, write, execute
  - AFS example
    - Construct arbitrary groups
    - Seven privileges: rliwdka
- Advantage: Easy to revoke privileges

Representation of Access Matrix

Capabilities

- For each user, indicate resources that can be accessed
  - General form: Each user has list of <resource, privilege> pairs
- Implementation: Naming
  - Secure pointer, whose value cannot be change
  - Cannot even name objects not in your capability list
  - Users cannot construct or copy these pointers
  - Often need hardware or language support

Examples

- Virtual address space
- File descriptor for an open file
- Unlisted phone number?

Advantages

- More secure: default is no access to object

Disadvantage

- Difficult to revoke capabilities

Access Enforcement

Responsibilities of security kernel

- Protecting identification and authorization information
- Enforcing access controls

Requirements

- Must run in protected mode
- As small and simple as possible

Paradox

- More powerful protection mechanism -->
  Larger and more complex security kernel -->
  More likely to have implementation bugs -->
  More security holes

Common Security Problems

Abuse of valid privileges

- Privileges are not fine grained enough
- Example: Super-user can do anything

Listener (or snooper)

- Eavesdrop on interconnect to steal information
- Example: Set ethernet card to promiscuous mode

Denial of Service (DoS) or Spoiler

- Consume all resources to make system crash or unusable
- Example: Grab all file space or create many processes

More Security Problems

Leverage Covert Channels

- Information leaks outside of normal interface
  - Time, power, page faults, ...
- Example: Tenex page-fault caper
  - System checked passwords until character didn't match
  - Cracked passwords by placing input string across page boundary
  - Measured time for password check
    - If very slow?
  - Number of needed attempts?
- Example: Power consumption on smart cards

More Security Problems

Imposter or Trojan Horse

- Application that misuses its environment
  - Paths including "." make users more vulnerable
- Examples
  - Program looks like login process
  - Editor that reads unauthorized files
  - ATMs

Trap door

- Designer leaves hole in software to leverage later
- Example: Login makes user a super-user regardless of password file
  - Problem: Inspection of source code reveals trap door
  - Change compiler to insert special code when compiling login!

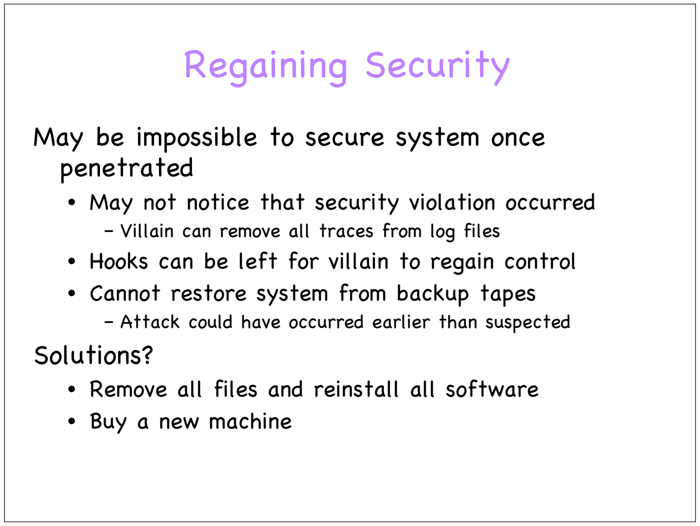
More Security Problems

Virus

- Fragment of malicious code embedded in legitimate code
- Spread by copying infected programs over network or floppy disk

Worm

- Capable of spreading itself from machine to machine
- Example: Thousands of computers disabled in Fall 1988
  - Sendmail attack:
    Debug command left enabled to execute code as super-user
  - Fingerd attack:
    Give long name to fingerd to overflow buffer and modify stack
  - Rsh: Crack passwords of local users by guessing common ones;
    Look for .rhost files for access to more machines



Regaining Security

May be impossible to secure system once penetrated

- May not notice that security violation occurred
  - Villain can remove all traces from log files
- Hooks can be left for villain to regain control
- Cannot restore system from backup tapes
  - Attack could have occurred earlier than suspected

Solutions?

- Remove all files and reinstall all software
- Buy a new machine