Protection and Security

Questions answered in these notes:
- How can the system authenticate a user?
- How are access rights specified?
- What are common security problems?

Reading for topic: Chapter 19 & 20

Components of Protection Mechanism

Authentication
- Make sure system knows which user is doing what action

Authorization determination
- Determine what the user is and is not allowed to do
- Separate policy and mechanisms

Access enforcement
- Make sure there are no loopholes in the system

Slightest flaw in any areas ruin entire protection mechanism

Motivation

Protection more important as computer systems develop
- Multiple users have access to same resources
- Computers connected to network
- Increasing importance of electronic commerce

Goals
- Prevent accidental misuse
  Example: Mistakenly overwrite command interpreter & no one can log in
  Relatively easy to solve by making likelihood small

- Malicious abuse
  Example: Break the password for accounting system & transfers $3 billion
  Hard to completely eliminate

Authentication

Authentication most often performed with passwords
- Secret piece of information used to establish the identity of a user
- Should not be stored in readable form
  One-way transformations should be used
- Passwords should be relatively long and obscure
  Short passwords are easy to crack
  Long passwords are easily forgotten and usually written down
- Disadvantage: Relatively weak form of protection

Another form of identification: key
- Does not have to be kept secret
- Should not be forgeable or able to be copied
  If stolen, the owner is aware
- Disadvantage: Key must be cheap to make, yet hard to duplicate
Authorization Determination

Access rights represented with *access matrix*

One domain (e.g., user or process) per one row
One resource per column
Each entry indicates the privileges of that domain on that object

| Domain  | File A | Printer | TTY 3 | ...
|---------|--------|---------|-------|-----
| Domain 1 | R      | W       | RW    |     |
| Domain 2 | --     | W       | ---   |     |
| Domain 3 | RW     | W       | ---   |     |
| Domain 4 | RWX    | ---     | ---   | --- |
| Domain 5 | ---    | ---     | ---   | --- |

Additional considerations
- Can process change from one domain to another?
- Can access rights of entry be changed?
- Can rights be copied from one domain to another?

Representation of Access Matrix

Full access matrix is sparsely populated
- Information condensed in two forms: Access lists and capabilities

Access Lists
- For each resource, indicate users that can perform operations
  - Column of access matrix
- Most general: each object has list of <user, privilege> pairs

Disadvantage
- Tedious to have separate entry for every user
  - Optimization: Group users into classes
  - UNIX Example: Three classes: self, group, everyone else

Advantages
- Access lists are simple and used in almost all file systems
- Revocation is trivial

Revocation of Access Rights

Potentially difficulty: Capabilities distributed throughout system

Re-acquisition
- Periodically delete all capabilities
- Must reacquire capability to access again

Back-pointers
- Each object has pointers back to all capabilities
- Follow pointers back to remove capability

Indirection
- Capability does not point to object, but to entry in global table
- Delete entry in table

Keys: Unique, un-modifiable bit pattern
- Associate value of master key with local key
- Change master key value
Access Enforcement

Security kernel responsibilities
- Protecting identification and authorization information
- Enforcing access controls

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

Paradox
1. More powerful protection mechanism
2. Larger and more complex security kernel
3. More likely to have implementation bugs

Levels of protection
- Most systems let entire OS run in all-powerful mode
  Such systems are not very secure

Common Security Problems

Abuse of valid privileges
- Privileges are not fine grained enough
- Example: Super-user can do anything

Listener
- Eavesdrop on terminal wire or local network to steal information
- Set Ethernet card to promiscuous mode

Denial of Service or Spoiler
- Consume all resources and make system crash or unusable
- Example: Grab all file space or create many processes

More Security Problems

Leverage Covert Channels
- Information that leaks outside of normal interface
- Example: Tenex page-fault caper
  System checked password only until no match
  Cracked passwords by placing input string across page boundaries
  Measured time for password check
  Solution: System touches entire password

<table>
<thead>
<tr>
<th>password</th>
<th>Page faults:</th>
<th>Page faults:</th>
</tr>
</thead>
<tbody>
<tr>
<td>a a a a a</td>
<td>0</td>
<td>c a a a</td>
</tr>
<tr>
<td>b a a a a</td>
<td>0</td>
<td>c b a a</td>
</tr>
<tr>
<td>c a a a a</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- Impossible to confine all information (e.g., power consumption)

Imposter or Trojan Horse
- Program that misuses its environment
- Many examples
  - Program looks like login process, remembers passwords
  - Editor that reads unauthorized files
  - ATMs
  - Fake deposit slips

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!
  - Compiler code also show trap door, so have special compiler for compiler
  - Simple compiler only distributed in binary!
More Security Problems

**Virus**
- Fragment of code embedded in legitimate code
- Problem for personal computers
- Spread by copying infected programs over network or floppy disk

**Worm**
- Process that is capable of spreading itself from machine to machine
- Example: Disabled thousands of computers in Fall of 1988
- Sendmail attack
  - Leverage 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**
- Not all possible to tell that security violation occurred
  - Villain can remove all traces from log files
- Hooks could have been left around for the imposter to regain control
- Cannot restore system from backup tapes
  - Attack could have occurred earlier than suspected

**Only solution**
- Remove all files from disk and reinstall all software