UnitelmaSapienza
Università degli Studi di Roma

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Secure Systems Design Principles

F. Parisi Presicce

Secure Systems Design Principles

According to Saltzer and Schroeder – 1975
http://web.mit.edu/Saltzer/www/publications/protection/index.html

- Overview
- Principles
    - Least Privilege
    - Fail-Safe Defaults
    - Economy of Mechanism
    - Complete Mediation
    - Open Design
    - Separation of Privilege
    - Least Common Mechanism
    - Psychological Acceptability

- More principles

Overview

Simplicity

- Less to go wrong, less to check
- Fewer entities communicating thru possible inconsistent interfaces
- Easy to understand

Restriction

- Minimize access/interactions
  - E.g.; “need to know”
- Inhibit communication
  - E.g.; writing

1. Least Privilege

“A subject/program should be given only the minimum set of privileges necessary to complete its task”

- Function, not identity, determines controls
- Rights added as needed, discarded after use

Examples:

- System operator should not perform security administration functions
- Execution of an operating system utility by application program only performs at privilege level of application program.
- UNIX network server to access port <1024 needs root; if no longer needed (SMTP server), drop
- (counter) In UNIX editor to edit single file has all privileges of account that called it, including reading or deleting other files

Compartmentalization

- Weakness of some Microsoft Applications: IIS 5 runs under the Local System account, equivalent to root privileges. Apache may run as “nobody” under UNIX; under Windows the equivalent procedure is possible but convoluted (and rarely done).
- Technique to separate the code in different parts, so that each part runs with least privilege.
  - if a part is compromised, others are still OK
  - Example: Separating a user interface from the program running with special privileges (e.g., root)

2. Fail-Safe (permission based) Defaults

"Unless a subject is given explicit access to an object, it should be denied access to that object"

- Basic access decisions are made on permissions rather than exclusion.
- Default action is to deny, not grant, access
- If action fails, system as secure as when action began

Examples:

- Omitting a parameter from a system call should result in less permission (e.g.; format string)
- New file is accessible only to its creator - not to the world
- If firewall suffers failure, no packet is forwarded
- Apache access control through .htaccess: first (default) rule: deny from all, allow from

3. Economy of Mechanism

“Security mechanisms should be as simple as possible”

Complex mechanisms may not be correctly:

- understood
- modeled
- configured
- implemented
- used

Simpler means less can go wrong

- when errors occur, easier to understand and fix

Economy of Mechanism (cont.)

Keep the design, implementation, operation, interaction with other components as simple as possible, so that it can be analyzed, verified, tested, etc.

- KISS Principle

Examples:

- Program flaws are easier to detect with small modules of code.
- The security Kernel can be validated if kept small (formal validation)
- (negative) IPSEC

4. Complete Mediation

“All accesses to objects must be checked to ensure that they are allowed”

Performance vs. security issue

- Results of access check are often cached
- What if permissions have changed since the last check?
- Mechanisms to invalidate or flush caches after a change are often missing

Architecture issue

- Capability granting and management
  - How did a capability given to Alice end up in Malory's hands?

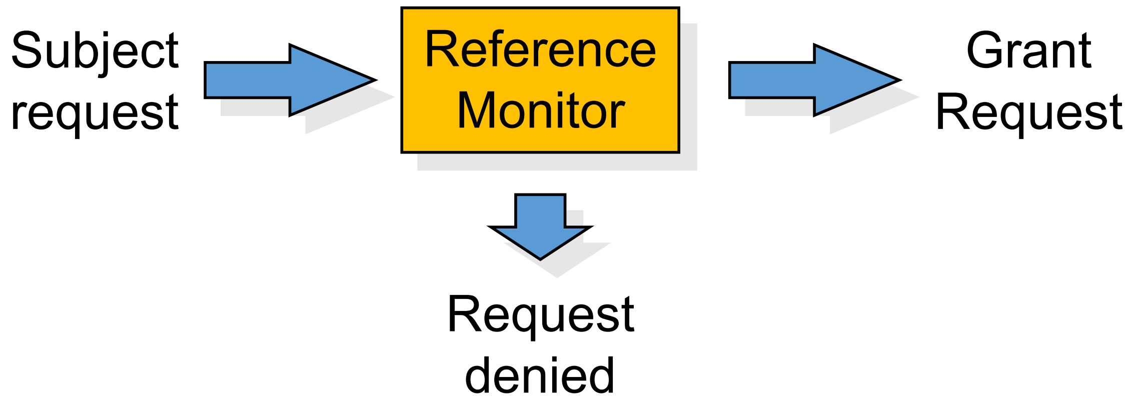
Complete Mediation (cont.)

Every access to every object must be validated. No path may violate this. Usually done once, on first action

- UNIX: access checked on open, not checked thereafter
- If permissions change after, may get unauthorized access

Decision procedure: whether access should be granted using a Reference monitor:

- considers requests, grant some, deny some



5. Open Design

“The security of a mechanism should not depend on the secrecy of its design or implementation.”

- If the details of the mechanism leaks (through reverse engineering, dumpster diving or social engineering), then it is a catastrophic failure for all the users at once.
- If the secrets are abstracted from the mechanism, e.g., inside a key, then leakage of a key only affects one user.

Open Design (cont)

Security should not depend on secrecy of design or implementation

- Popularly misunderstood to mean that the source code should be made public
- "Security through obscurity": correct operation is not related to the secrecy of the design
- Does not apply to information such as passwords or cryptographic keys, where secrecy is needed

6. Separation of Privilege

"A system should not grant permission based on a single condition."

- Removes a single point of failure
- Require multiple conditions to grant privilege and two or more system components work together to enforce security
  - Defense in depth

Example: two-factor authentication

- Requiring both biometric and token recognition mechanism reduces risks

Separation of Privilege (cont.)

Analogous to the separation of duty:

- By requiring multiple factors, collusion becomes necessary, and risks due to bribery (compromise of one factor) are reduced
- Dual-signature checks for amount over threshold

If one component is defeated, the system is not completely compromised

7. Least Common Mechanism

“Mechanisms used to access resources should not be shared”

Concept: You have two different services, of different priorities and value, provided to two different sets of users. The more resources they share, the more likely one can influence the other in order to:

- Transmit forbidden data (covert channels issue)
- Limit availability (denial of service, FTP and Web services on the same computer share a common thread pool)

Least Common Mechanism (cont.)

Mechanisms should not be shared

- Information can flow along shared channels that may become covert channels
- To send a 1 bit, first process uses 75% of the CPU; to send a 0 bit, it uses 25%. The other process sees how much CPU it can get and deduce what the first process uses, and hence is sending.

Isolation

- Virtual machines (to completely isolate underlying hardware/OS from application/user)
- Sandboxes (to limit access to resources)

8. Psychological Acceptability

Security mechanisms should not make the resource more difficult to access than if the security mechanism were not present (impossible !)

- Example: Commercial where users have become bald and lost (all?) their hair in order to comply with a biometric authentication mechanism requesting hair samples.
- Problem: Users look for ways to defeat the mechanisms and “prop the doors open”
- In practice, difficulty proportionate to the value of the protected asset is generally accepted

Psychological Acceptability (cont.)

Examples:

- Users write down passwords which are too difficult to remember
- .rhosts mechanism bypasses password security check

Security mechanisms should not add (too much) to difficulty of accessing resource

- Hide complexity introduced by security mechanisms
- Ease of installation, configuration, use
- Human factors critical here

More Principles

Principles applicable at many levels

- In source code
- Between applications on a machine
- At OS level
- At network level
- Within organization
- Between organizations

and vulnerabilities often exploit violations of principles

Some overlap and tension between some of these principles

More Principles

Secure weakest link

- Requires good risk analysis
- Best investment: educating users

Minimize attack surface

- Fewer open sockets, services (running by default or with high privileges), dynamic content web pages, ...

Identify the assumptions

- Especially obvious and implicit (that attacker invalidates)

Be reluctant to trust

- Trust is transitive, not good.
- All user input is evil ! Third-party software ?

Clearly assign responsibility

- At organizational level and at coding level

Key Points

Principles of secure design underlie all security-related mechanisms

Require:

- Good understanding of the goal of the mechanism and environment in which it is to be used
- Risk analysis
- Careful analysis and design
- Careful implementation