# Design Principles for Secure Systems

# Driving Ideas for Security Principles

- Saltzer and Schroeder (1975) defined 8 principles that are based on the ideas of simplicity and restriction
- Simplicity
  - Less to go wrong
  - Fewer possible inconsistencies
  - Easy to understand
- Restriction
  - Minimize access – an entity can access only information it needs (also known as “need to know” principle)
  - Inhibit communication – an entity can communicate with other entities only when necessary, and in few (and narrow) ways as possible

# Design Principles

- Economy of Mechanism
- Open Design
- Principle of Least Privilege
- Complete Mediation
- Separation of Privilege
- Failsafe Defaults
- Least Common Mechanism
- Psychological Acceptability
- Additional principles
  - Diversity of Mechanism
  - Multiple Lines of Defense

# Economy of Mechanism

- Use simple and straightforward mechanisms wherever possible
  - Simplicity means
    - Less can go wrong
    - When errors occur, they are easier to understand and fix
  - Interfaces and interactions
    - Interfaces to other modules are crucial because modules often make implicit assumptions about input or output parameters or the current system state
- Avoid a lot of extra features.  Bells and whistles add complication, and that introduces errors
- The simplest mechanism that does the job is the best

# Economy of Mechanism

- Consider reusing components whenever possible, as long as the components to be reused are believed to be of good quality.
  - Why would anyone want to re-implement AES or SHA-1, when there are several widely used libraries available?
- Do not implement unnecessary security mechanisms.
  - An example is file encryption supporting the access control service that in turn supports the goals of confidentiality and integrity by preventing unauthorized file access.
  - If file encryption is a necessary part of accomplishing the goals, then the mechanism is appropriate.
  - However, if these security goals are adequately supported without inclusion of file encryption, then that mechanism would be an unneeded system complexity.

# Open Design

- No security through obscurity. The security of a mechanism should not depend on the secrecy of its design and implementation (i.e., attacker's ignorance of how the mechanism works or is built)
  - If the strength of a program's security depends on the ignorance of user, a knowledgeable user can defeat the security mechanism
    - "Security through obscurity" is not a good principle
  - How about passwords or cryptographic keys?
- This principle is controversial.
  - Showing a design or source code to attackers certainly makes their task easier. NSA, for instance, refused to publish their analysis that the DES encryption algorithm is secure. The lack of information, however, decreases people's assurance in the security of DES.
  - Publicizing the design give security researchers the opportunity to find and fix the flaws before the attackers do.

# Open Design

- Issues of proprietary software and trade secrets complicate the application of this principle
- In some cases, companies do not want their designs made public to protect them from competitors
- The principle then requires that the design and implementation be available to people barred from disclosing it outside the company

# Principle of Least Privilege

- Every subject/program should be given the minimum set of privileges necessary to complete the job.
  - The function of a subject, and not its identity, should control the assignment of rights
  - Rights should be added as needed, discarded after use
- This limits the damage that can result from an accident or error
- It limits the number of privileged programs (which could be compromised) in the system.
- It also helps in debugging, is good for increasing assurance, and allows isolation of small critical subsystems.

# Complete Mediation

- Complete Mediation. Every access to every object is checked.
  - In practice, this is relaxed/violated! Why?
  - Usually done once, on first action
    - UNIX: access checked on OPEN, not checked thereafter (READ)
    - If permissions change after, the user may get unauthorized access
  - Example: process for user A opens a file; user A is terminated revoking all his privileges; process accesses file which has been open for days and user privilege is not verified.

# Separation of Privilege

- Two keys are better than one. Each privilege should require a separate secret.
  - A system should not grant permission based on a single condition
  - More than one condition must be verified in order to gain access
- Separate passwords for separate objects.

# Separation of Privilege

- Example: company cheques for more than $100k must be signed by two officers of the company
- Example: On Berkeley-based version of Unix, a user is not allowed to change from his account to the root account unless two conditions are verified: (i) the user knows the root password; (ii) the user is in the wheel group (with GID 0)
- This allows finer-grained control of access to the system, and limits what can be compromised if a single secret is revealed.
- Can be overly cumbersome for the user.

# Failsafe Defaults

- No access by default. It is much better (and less prone to error) to define who *can* have access than to directly define who *cannot*.
  - Problem: user needs a privilege that was not anticipated so his work is delayed while the privilege is authorized.
- If the subject is not able to complete its action or task, it should undo those changes it made in the security state of the system before it terminates. If the program fails, the system is still safe.
  - What happens if the program crashes, not fails?

# Least Common Mechanism

- Mechanisms used to access resources should not be shared
  - Information can flow along shared channels
  - Covert channels
- This principle is implemented by isolation mechanisms
  - Virtual machines
  - Sandboxes

# Psychological Acceptability

- Security mechanisms should not make the resource more difficult to access than if the security mechanisms were not present.
  - Hide complexity introduced by security mechanisms
  - Ease of installation, configuration, use
  - Human factors critical here
  - On the other hand, security requires that the messages impart no unnecessary information
    - For example, if a user supplies the wrong password, the system should reject the attempt with a message saying that the login failed
      - If it were to say that the password was incorrect, the user would know that the account name was legitimate

# Other principles: Diversity of Mechanism

- Diversity of Mechanism  Security mechanisms that have the same design or follow similar logic are likely to fail in similar ways (hence, at the same time or fall to the same trick of an attacker). Diverse mechanisms are unlikely to share vulnerabilities.
- With diverse mechanisms, the odds are increased that no single vulnerability is common to all.

# Other principles: Multiple Lines of Defense

- Unfortunately, no security mechanism is totally secure.  Plan for something to fail and have a second (and third) line of defense.
- Example: Try to keep the bad guys out (firewall) but if they do get in, minimize the harm they can do (strict access controls), and if they manage to get access, have good audit logs so you can track them down and prosecute.

# Key Points

- Principles of secure design underlie all security-related mechanisms
- They encompass not only technical details but also human interaction
- They require
  - Good understanding of
    - The goal of the security mechanism and
    - The environment in which it is to be used
  - Careful analysis and design
  - Careful implementation