CS6501: Great Works in Computer Science

Jan. 29th 2013
Longze Chen

The Protection of Information in Computer Systems

Jerome H. Saltzer and Michael D. Schroeder

Jerry Saltzer

Michael Schroeder

The Meaning of Computer Security

- What does “Secure/Security” mean to a computer system?

- If you are a user of a system, what do you expect from it with regard to protecting your personal data?

- Security Goals
    - **Confidentiality** ensures that computer-related assets are accessed only by authorized parties.
    - **Integrity** means that assets can be modified only by authorized parties or only in authorized ways.
    - **Availability** means that assets are accessible to authorized parties at appropriate times.

- Control
    - Encryption
    - Software Control
    - Hardware Control
    - Policy and Procedures

Definitions by Saltzer & Schroeder:

- Security
    - With respect to information processing systems, the term “**Security**” describes mechanisms and techniques that control who may use or modify the computer or the information stored in it.
- Security Violations
    - Unauthorized Information Release;
    - Unauthorized Information Modification;
    - Unauthorized Denial of Use.
- Protection
    - Equivalent to Security;
    - Mechanisms and techniques that control the access of executing programs to stored information.
- Authentication
    - Mechanisms and techniques that verify the identity of a person (or other external agent) making a request of a computer system.

Levels of Information Protection

- Unprotected System
-
- Information Access
    - All-or-nothing System
        - Is this apply to the whole system or per objects?
    - Controlled Sharing
        - Who can access the data
        - Which access pattern is allowed
    - User-programmed Sharing Controls
        - Unstandard access pattern
        - Protected objects and subsystems
- Information Propagation
    - Putting Strings on Information
- “Dynamics of Use”
    - Establish or make change to who may access what?

Design Principles (in Saltzer & Shroeder)

- Why Do We Need Design Principles
    - It can guide the design and contribute to an implementation without security flaws.

- Economy of mechanism
    - Keep the design as simple and small as possible.
    - Design and implementation errors that result in unwanted access paths will not be noticed during normal use. As a result, techniques such as line-by-line inspection of software and physical examination of hardware that implements protection mechanisms are necessary. In this case, simple design is essential.

- Fail-safe Defaults
    - Base access decisions on permission rather than exclusion.
    - This principle, suggested by E. Glaser in 1965,1 means that the default situation is lack of access, and the protection scheme identifies conditions under which access is permitted.
    - A design or implementation mistake in a mechanism that gives explicit permission tends to fail by refusing permission, a safe situation, since it will be quickly detected.

- Complete mediation
    - Every access to every object must be checked for authority.
    - This principle, when systematically applied, is the primary underpinning of the protection system. It forces a system-wide view of access control, which in addition to normal operation includes initialization, recovery, shutdown, and maintenance.

- Open Design
    - The design should not be secret.
    - The mechanisms should not depend on the ignorance of potential attacks, but rather on the possession of specific, more easily protected, keys or passwords.

- Separate of Priviledge
    - Where feasible, a protection mechanism that requires two keys to unlock it is more robust and flexible than one that allows access to the presenter of only a single key.

- Least Priviledge
    - Every program and every user of the system should operate using the least set of privileges necessary to complete the job.

    - Primarily, this principle limits the damage that can result from an accident or error.
    - It also reduces the number of potential interactions among privileged programs to the minimum for correct operation, so that unintentional, unwanted, or improper uses of privilege are less likely to occur.

- Least Common Mechanism
    - Minimize the amount of mechanism common to more than one user and depended on by all users.
    - Every shared mechanism (especially one involving shared variables) represents a potential information path between users and must be designed with great care to be sure it does not unintentionally compromise security.
    - Further, any mechanism serving all users must be certified to the satisfaction of every user, a job presumably harder than satisfying only one or a few users.

- Psychological Acceptability
    - It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly.
    - Also, to the extent that the user's mental image of his protection goals matches the mechanisms he must use, mistakes will be minimized.

Extra Principles (by Cigital, Inc.)

- Defense In Depth
    - The idea behind defense in depth is to manage risk with diverse defensive strategies, so that if one layer of defense turns out to be inadequate, another layer of defense will hopefully prevent a full breach.

- Never Assuming That Your Secrets Are Safe
    - Always assume that an attacker knows everything that you know -- assume the attacker has access to all source code and all designs. Even if this is not true, it is trivially easy for an attacker to determine obscured information.

- Promotion Privacy
    - Many users consider privacy a security concern. Try not to do anything that might compromise the privacy of the user. Be as diligent as possible in protecting any personal information a user does give your program.

- Reluctant to Trust
    - Assume that the environment where your system operates is hostile.

- Securing the Weakest Link
    - Security practitioners often point out that security is a chain; and just as a chain is only as strong as the weakest link, a software security system is only as secure as its weakest component. Bad guys will attack the weakest parts of your system because they are the parts most likely to be easily broken.

General Protection Model

- Fundamental **Objects** for Protection
    - Partitioned Information
    - Uniformity of Protection

- Build an **impenetrable wall** around each distinct object.

- Construct a **door** in the wall through which access can be obtained.

- Post a **guard** at the door to control its use.
    - The guard knows which user are authorized to have access.
    - Each user can identify himself to the guard.

- **Authority check**
    - The guard demand a match between something he knows and the something the prospective user possesses.

Example Model 1: Protection Model

Example Model 2: Authentication Model

- Authentication
    - A system verify the identity of a claimed user.
    - A user verify the expected system in communication.

- Secrecy
    - Password

- Unforgeability
    - ID Cards, Physical Keys
    - Biometrics

- Two-way authentication
    - Protect against masquerading

Example Model 3: Shared Information Model

Access Control List and Capability

- Two categories of implementation of sharing protection
    - List-oriented
        - Guard
        - User
    - Ticket-oriented
        - Guard
        - User

- Access Control List
    - There is one such list for each object and the list shows all subjects who should have access to the object and what their access is.
        - “A list of principals that are authorized to have access to some object.”

- Capability
    - A capability is an unforgeable token that give permission to a subject to have a certain type of access to an object.
        - “In a computer system, capability is an unforgeable ticket, which when presented can be taken as incontestable proof that the presenter is authorized to have access to the object named in the ticket.”

- Directory
    - Each user has a file directory, which lists all the files to which that user has access.