Design Principles

We will be looking at eight principles for the design and implementation of security mechanisms. These principles draw on the ideas of simplicity and restriction. Simplicity makes designs and mechanisms easy to understand. Less can go wrong with simple designs. Minimizing the interaction of system components minimizes the number of sanity checks on data being transmitted from one component to another. Simplicity also reduces the potential for inconsistencies within a policy or set of policies. Restriction minimizes the power of an entity. The entity can access only information it needs. Entities can communicate with other entities only when necessary and in as few and narrow ways as possible. Communications is used in its widest possible sense, including that of imparting information by not communicating.

Describe how the sendmail system works and some of the issues that could develop as a result of how it is designed.

The eight design principles are:

1. Principle of Least Privilege

A subject should be given only those privileges that it needs in order to complete its task.

The function of a subject should control the assignment of rights, not the identity of the subject. This means that if your boss demands root access to a UNIX system that you administer, she should not be given that privilege unless she absolutely has a task that requires such level of access. If possible, the elevated rights of an identity individual should be removed as soon as those rights are no longer required.

e.g.

sudo
su programs
set uid only when needed

2. Principle of Fail-Safe Defaults

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

This principle restricts how privileges are initialized when a subject or object is created. Basically, this principle is similar to the “Default Deny” principle that we talked about in the 6 dumbest ideas in computer security. Whenever access, privilege, or some other security related attribute is not granted, that attribute should be denied by default.

3. Principle of Economy of Mechanism

Security mechanisms should be as simple as possible.

This principle simplifies the design and implementation of security mechanisms. If the design and implementation are simple, fewer possibilities exist for errors. The checking and testing process is less complex. Interfaces between security modules are suspect area and should be as simple as possible.

4. Principle of Complete Mediation

All accesses to objects should be checked to ensure that they are allowed.

This principle restricts the caching of information, which often leads to simpler implementations of mechanisms. Every time that someone tries to access an object, the system should authenticate the privileges associated with that subject. What happens in most systems is that those privileges are cached away for later use. The subject's privileges are authenticated once at the initial access. For subsequent accesses the system assumes that the same privileges are enforce for that subject and object. This may or may not be the case. The operating system should mediate all and every access to an object.

e.g.

DNS information is cached
What if it is poisoned?

5. Principle of Open Design

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

This principle suggests that complexity does not add security. This concept captures the term "security through obscurity". This principle not only applies to cryptographic systems but also to other computer security related systems.

e.g.

DVD player & Content Scrambling System (CSS) protection

6. Principle of Separation of Privilege

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

This principle is restrictive because it limits access to system entities. The principle is similar to the separation of duty principle that we talked about in the integrity security policy unit. Thus before privilege is granted two or more checks should be performed.

e.g.

to su (change) to root two conditions must be met

1. the user must know the root password
2. the user must be in the right group (wheel)

7. Principle of Least Common Mechanism

Mechanisms used to access resources should not be shared.

This principle is also restrictive because it limits sharing of resources.
Sharing resources provides a channel along which information can be transmitted. Hence, sharing should be minimized as much as possible. If the operating system provides support for virtual machines, the operating system will enforce this privilege automatically to some degree.

8. Principle of Psychological Acceptability

Security mechanisms should not make the resource more difficult to access than if the security mechanism were not present.

Do you believe this? This principle recognizes the human element in computer security. If security-related software or systems are too complicated to configure, maintain, or operate, the user will not employ the requisite security mechanisms. For example, if a password is rejected during a password change process, the password changing program should state why it was rejected rather than giving a cryptic error message. At the same time, programs should not impart unnecessary information that may lead to a compromise in security. In practice, the principle of psychological acceptability is interpreted to mean that the security mechanism may add some extra burden, but that burden must be both minimal and reasonable.

e.g.

When you enter a wrong password, the system should only tell you that the user id or password was wrong. It should not tell you that only the password was wrong as this gives the attacker information.