Software System Security

States of a Computer System

The state of a system is the collection of the current values of all components of the system: memory locations, secondary storage, registers, etc.

Protection states are those states that have to be protected.

P = set of all protection states of the system

.Q = set of all authorized protection states

The system is not secure if the current state is in P - Q

(P - Q means that all elements of set P not in set Q)

Security policy characterizes the states in Q

A security mechanism prevents the system entering a state in P - Q

Access Control Matrix Model

A model used to describe the protection states.

It characterizes the rights of each subject of the system (entity/process) regarding the objects of the system (entities/processes) in terms of a matrix.

The set of all protected entities (that is, entities that are relevant to the protection state of the system) is called the set of objects O.

The set of subjects S is the set of active objects, such as processes and users. In the access control matrix model, the relationship between these entities is captured by a matrix A with rights drawn from a set of rights R in each entry A[s, o], where $s \in S$, $o \in O$, and $A[s, o] \in R$.

The subject s has the set of rights A[s, o] over the object o. The set of protection states of the system is represented by the triple (S,O,A).

Access Control Matrix Model

	file 1	file 2	process 1	process 2
process 1	R, W, O	R	R, W, E, O	W
process 2	A	R, O	R	R, W, E, O

Here R = {Read, Write, Own, Append, Execute}

Access Control Matrix Model

	File A	File B	File C	Printer 1
Alice	RW	RW	RW	OK
Bob	R	R	RW	OK
Carol	RW			
David			RW	OK
Faculty	RW		RW	OK

Access Control Matrix Model

	File A	File B	File C	Printer 1
Alice	RW	RW	RW	OK
Bob	R	R	RW	OK
Carol	RW			
David			RW	OK
Faculty	RW		RW	OK

Access Control List

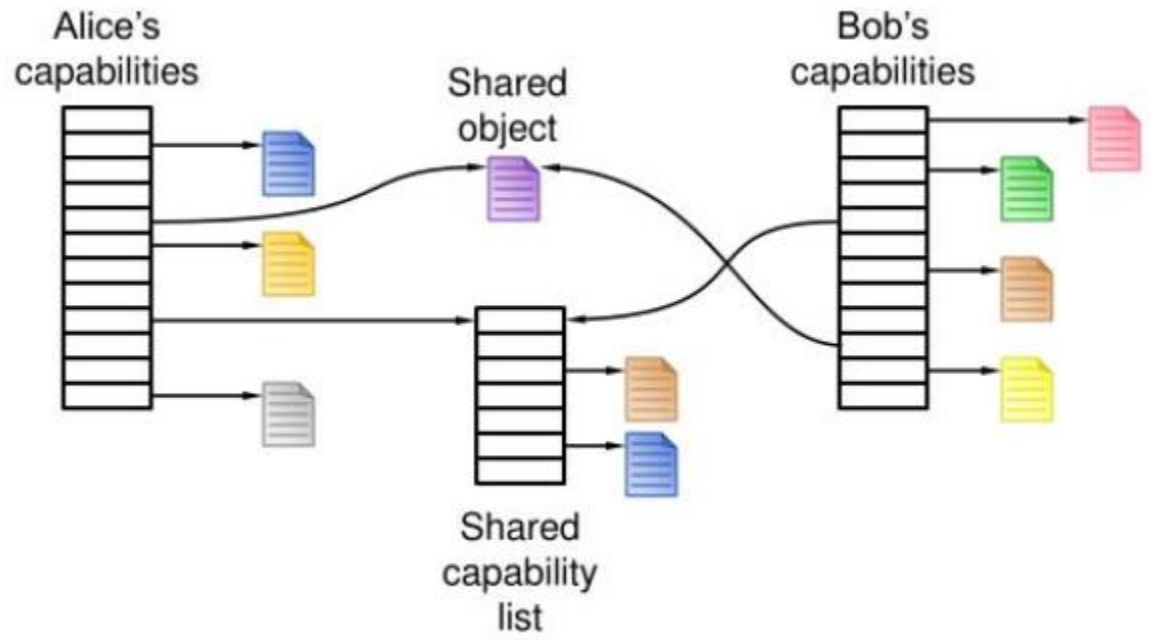
Access Control Matrix Model

	File A	File B	File C	Printer 1
Alice	RW	RW	RW	OK
Bob	R	R	RW	OK
Carol	RW			
David			RW	OK
Faculty	RW		RW	OK

Capability List

Capability Lists

Sharing with Capabilities



Security Policies

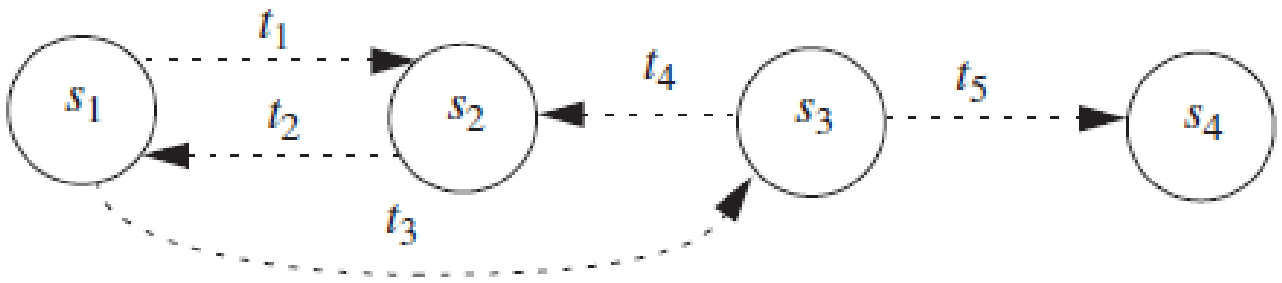
A security policy is a statement that partitions the states of the system into a set of authorized, or secure, states and a set of unauthorized, or nonsecure, states.

A secure system is a system that starts in an authorized state and cannot enter an unauthorized state.

Security Policies

Consider a finite-state machine.. The security policy partitions the states into a set of authorized states A = { s1, s2 } and a set of unauthorized states UA = { s3, s4 }.

This system is not secure, because regardless of which authorized state it starts in, it can enter an unauthorized state. However, if the edge from s1 to s3 were not present, the system would be secure, because it could not enter an unauthorized state from unauthorized state.



Security Policies

• Ease of use versus security: Virtually all security measures involve some penalty in the area of ease of use. The following are some examples. Access control mechanisms require users to remember passwords and perhaps perform other access control actions. Firewalls and other network security measures may reduce available transmission capacity or slow response time. Virus-checking software reduces available processing power and introduces the possibility of system crashes or malfunctions due to improper interaction between the security software and the operating system.

• Cost of security versus cost of failure and recovery: In addition to ease of use and performance costs, there are direct monetary costs in implementing and maintaining security measures. All of these costs must be balanced against the cost of security failure and recovery if certain security measures are lacking. The cost of security failure and recovery must take into account not only the value of the assets being protected and the damages resulting from a security violation, but also the risk, which is the probability that a particular threat will exploit a particular vulnerability with a particular harmful result.

Security mechanism

Security mechanism is an entity or procedure that enforces some part of the security policy.

Fundamental Security Design Principles

- Economy of mechanism
- Fail-safe defaults
- Complete mediation
- Open design
- Separation of privilege
- Least privilege
- Least common mechanism
- Psychological acceptability
- Isolation
- Encapsulation
- Modularity
- Layering
- Least astonishment

Fundamental Security Design Principles

- Economy of mechanism
- Fail-safe defaults
- Complete mediation
- Open design
- Separation of privilege
- Least privilege
- Least common mechanism
- Psychological acceptability
- Isolation
- Encapsulation
- Modularity
- Layering
- Least astonishment

Fundamental Security Design Principles

Economy of mechanism means that the design of security measures embodied in both hardware and software should be as simple and small as possible. The motivation for this principle is that relatively simple, small design is easier to test and verify thoroughly. With a complex design, there are many more opportunities for an adversary to discover subtle weaknesses to exploit that may be difficult to spot ahead of time. The more complex the mechanism, the more likely it is to possess exploitable flaws. Simple mechanisms tend to have fewer exploitable flaws and require less maintenance. Furthermore, because configuration management issues are simplified, updating or replacing a simple mechanism becomes a less intensive process. In practice, this is perhaps the most difficult principle to honor. There is a constant demand for new features in both hardware and software, complicating the security design task. The best that can be done is to keep this principle in mind during system design to try to eliminate unnecessary complexity.

Fundamental Security Design Principles

Fail-safe default means that access decisions should be based on permission rather than exclusion. That is, the default situation is lack of access, and the protection scheme identifies conditions under which access is permitted. This approach exhibits a better failure mode than the alternative approach, where the default is to permit access. A design or implementation mistake in a mechanism that gives explicit permission tends to fail by refusing permission, a safe situation that can be quickly detected. On the other hand, a design or implementation mistake in a mechanism that explicitly excludes access tends to fail by allowing access, a failure that may long go unnoticed in normal use. For example, most file access systems work on this principle and virtually all protected services on client/server systems work this way.

Fundamental Security Design Principles

Complete mediation means that every access must be checked against the access control mechanism. Systems should not rely on access decisions retrieved from a cache. In a system designed to operate continuously, this principle requires that, if access decisions are remembered for future use, careful consideration be given to how changes in authority are propagated into such local memories. File access systems appear to provide an example of a system that complies with this principle. However, typically, once a user has opened a file, no check is made to see of permissions change. To fully implement complete mediation, every time a user reads a field or record in a file, or a data item in a database, the system must exercise access control. This resource-intensive approach is rarely used.

Fundamental Security Design Principles

Separation of privilege is a practice in which multiple privilege attributes are required to achieve access to a restricted resource. A good example of this is multifactor user authentication, which requires the use of multiple techniques, such as a password and a smart card, to authorize a user. The term is also now applied to any technique in which a program is divided into parts that are limited to the specific privileges they require in order to perform a specific task. This is used to mitigate the potential damage of a computer security attack.

One example of this latter interpretation of the principle is removing high privilege operations to another process and running that process with the higher privileges required to perform its tasks. Day-to-day interfaces are executed in a lower privileged process.

Fundamental Security Design Principles

Least privilege means that every process and every user of the system should operate using the least set of privileges necessary to perform the task.

The system security policy can identify and define the various roles of users or processes. Each role is assigned only those permissions needed to perform its functions. Each permission specifies a permitted access to a particular resource (such as read and write access to a specified file or directory and connect access to a given host and port). Unless permission is granted explicitly, the user or process should not be able to access the protected resource.

More generally, any access control system should allow each user only the privileges that are authorized for that user.

There is also a temporal aspect to the least privilege principle. For example, system programs or administrators who have special privileges should have those privileges only when necessary; when they are doing ordinary activities the privileges should be withdrawn. Leaving them in place just opens the door to accidents.

Fundamental Security Design Principles

Least common mechanism means that the design should minimize the functions shared by different users, providing mutual security.

This principle helps reduce the number of unintended communication paths and reduces the amount of hardware and software on which all users depend, thus making it easier to verify if there are any undesirable security implications.

Fundamental Security Design Principles

Psychological acceptability implies that the security mechanisms should not interfere unduly with the work of users, while at the same time meeting the needs of those who authorize access. If security mechanisms hinder the usability or accessibility of resources, users may opt to turn off those mechanisms. Where possible, security mechanisms should be transparent to the users of the system or at most introduce minimal obstruction. In addition to not being intrusive or burdensome, security procedures must reflect the user's mental model of protection. If the protection procedures do not make sense to the user or if the user must translate his image of protection into a substantially different protocol, the user is likely to make errors.

Fundamental Security Design Principles

Isolation is a principle that applies in three contexts. First, public access systems should be isolated from critical resources (data, processes, etc.) to prevent disclosure or tampering. In cases where the sensitivity or criticality of the information is high, organizations may want to limit the number of systems on which that data are stored and isolate them, either physically or logically. Physical isolation may include ensuring that no physical connection exists between an organization's public access information resources and an organization's critical information. When implementing logical isolation solutions, layers of security services and mechanisms should be established between public systems and secure systems responsible for protecting critical resources. Second, the processes and files of individual users should be isolated from one another except where it is explicitly desired. All modern operating systems provide facilities for such isolation, so that individual users have separate, isolated process space, memory space, and file space, with protections for preventing unauthorized access. And finally, security mechanisms should be isolated in the sense of preventing access to those mechanisms. For example, logical access control may provide a means of isolating cryptographic software from other parts of the host system and for protecting cryptographic software from tampering and the keys from replacement or disclosure.

Fundamental Security Design Principles

Encapsulation can be viewed as a specific form of isolation based on object-oriented functionality. Protection is provided by encapsulating a collection of procedures and data objects in a domain of its own so that the internal structure of a data object is accessible only to the procedures of the protected subsystem and the procedures may be called only at designated domain entry points.

Fundamental Security Design Principles

Encapsulation can be viewed as a specific form of isolation based on object-oriented functionality. Protection is provided by encapsulating a collection of procedures and data objects in a domain of its own so that the internal structure of a data object is accessible only to the procedures of the protected subsystem and the procedures may be called only at designated domain entry points.

Fundamental Security Design Principles

Modularity in the context of security refers both to the development of security functions as separate, protected modules and to the use of a modular architecture for mechanism design and implementation. With respect to the use of separate security modules, the design goal here is to provide common security functions and services, such as cryptographic functions, as common modules. For example, numerous protocols and applications make use of cryptographic functions. Rather than implementing such functions in each protocol or application, a more secure design is provided by developing a common cryptographic module that can be invoked by numerous protocols and applications. The design and implementation effort can then focus on the secure design and implementation of a single cryptographic module, including mechanisms to protect the module from tampering. With respect to the use of a modular architecture, each security mechanism should be able to support migration to new technology or upgrade of new features without requiring an entire system redesign. The security design should be modular so that individual parts of the security design can be upgraded without the requirement to modify the entire system.

Fundamental Security Design Principles

Least astonishment means that a program or user interface should always respond in the way that is least likely to astonish the user. For example, the mechanism for authorization should be transparent enough to a user that the user has a good intuitive understanding of how the security goals map to the provided security mechanism.