Security & Protection
Security & Protection

- Deal with the control of unauthorized use and access of computer system resources
- potential security violations
  - unauthorized information release
  - unauthorized information modification
  - unauthorized denial of service
- security types
  - physical/external security deals with the devices in the system
  - internal security deals with the information in the system
- separation of policies & mechanisms
  - security=policies
  - protection=mechanism for implementing policies
Security & Protection

- The protection domain of a process
  - set of resources it can use and the types of operations it can perform on them
  - enables us to achieve the policy that a process accesses only the needed resources
Design Principles for Protection Mechanisms

- Economical to develop and use
- complete mediation for every access
- should work even if its underlying principles are known to attackers
- robustness and flexibility via separation of privileges: “two keys are needed to open a lock”
- least privilege sufficient to perform tasks
- least common mechanism among users
- simple and easy to easy to be acceptable
- fail-safe default
The Access Matrix Model

Protection model that has

- O = a set of current objects o
- S = a set of current subjects s
- S is a subset of O
- R = set of generic access rights
- P = an access matrix such that P[s,o] is a subset of R and specifies the access rights s has on o

protection state is the triplet (S, O, P)

reference monitor for each object o

which validates all access to o by any subject s
given (s,o,a), is it in P[s,o]?
Implementations of the Access Matrix Model

- Access matrix is generally sparse
- Direct implementations are wasteful of resources
- Decompose access matrix
  - By row => capability-based method
  - By column => access control list method
Capabilities

- Each subject $s$ is assigned a set of triples $(o, P[s,o])$, called capabilities, for the non-empty entries of $P$
- A subject having a capability is prima facie evidence that subject can access object in capability in the ways specified in the capability
- Capabilities must not be forgeable
- Capability-based addressing
Capability-Based Addressing

- Capability
  - Capability-id
  - Offset

- Object table
  - Base
  - Length

- Capability list
  - Access rights
  - Object-id

- Main memory
  - Object
Implementing Capabilities

Two approaches

- tagged=attach a bit to each memory location and register to distinguish data from capability content; manipulate capabilities only via privileged instructions
- partition=each object/register has two segments: one for data and one for capabilities
Advantages and Disadvantages of Capabilities:

**Advantages**
- efficient
- simple
- flexible

**disadvantages**
- controlling propagation
  - use copy bit or depth counter
- review of access is expensive
- revocation of access rights is difficult
- garbage collection is needed
Access Control Lists

- Implement the access matrix by a column-wise decomposition
- Each object o has a list of pairs (s, P[s,o]) for the non-empty P[s,o]
- Access control lists can be long
  - Slow for validating access requests
  - Takes lots of space
- However
  - Revocation of access rights is easy
  - Review of access rights is easy
- Protection groups (groups of subjects) can help to reduce the size of the lists
Access Control Lists

Who has authority to change access control list for an object?

- self-control=owner of object can modify it
- hierarchical-control=owner specifies subjects in a hierarchy that can modify it
**The Lock-Key Method**

- A hybrid between capabilities and access control lists
  - each subject has a list of capabilities \((o, k)\) for the objects in its protection domain, where \(k\) is a key (integer)
  - each object has an access control list \((l, r)\) where \(r\) is a subset of \(R\) and \(l\) is a lock (integer)

- how it works
  - at validation, a subject that wants to access \(o\) in mode \(x\) presents its capability to the reference monitor of \(o\)
  - reference monitor grants access request if \(k = l\) and \(x\) is in \(r\)

- capability-based addressing can be used
- has advantages of ACLs
Changing the system’s protection state is done with commands of the form

\[
\text{command } \langle \text{cmd-id} \rangle (<\text{params}>) \{ \\
\quad \text{if } \langle \text{conditions} \rangle \text{ then} \\
\quad \text{<list of primitive-operations>} \\
\quad \text{End} \\
\}
\]

where the set of primitive operations is

- enter r into P[s,o]
- delete r from P[s,o]
- create subject s
- create object o
- delete subject s
- delete subject o
Safety in Access Matrix Model

- Safe if a subject cannot acquire an access right to an object without consent of the object’s owner
  - impossible in the Access Matrix model
- A command leaks right r from state \( Q = (S, O, P) \) if it enters r in a cell of P that did not have r
- Safe if a subject can determine whether its actions can lead to the leakage of a right to unauthorized subjects
- State Q is unsafe for r if there exists a command that leaks r from Q; else Q is safe for r
- Safety can be decided for mono-operational systems
- It is undecidable for general protection systems
The Take-Grant Model

- Access matrix can be thought of as the adjacency matrix of a directed graph
  - an edge from x to y with label r, a subset of R, indicates that x has access rights r on y
  - two special access rights
    - Take: x can take any access rights y may have
    - Grant: y can be granted any rights x has

- state = graph
- create/delete operations add/remove nodes in the graph
- state transitions happen by executing take/grant operations
  - adding edges in the graph
The Take-Grant Model

Safety:
- given a state, is there a sequence of state transitions that lead into a graph with a specific edge?

Safety for take-grant model with general access rights/application rules is undecidable

Safety for specific access rights/application rules can be decided in polynomial time
The Bell and LaPadula Model

- Deals with information flow instead of access control
- It has
  - subjects, objects, and an access matrix
  - each subject has a clearance and a current clearance no more than its clearance
  - each object has a classification
- Access rights
  - read-only
  - append-only
  - execute
  - read-write
The Bell and LaPadula Model

- each object has a control attribute; a controller of an object can pass any access rights to any subject

Bell-LaPadula imposes the following two properties

- simple security property (reading down)
  - a subject can not read any objects with classification higher than its clearance

- star property (writing up)
  - a subject has
    - append rights only to objects with classification > its current clearance
    - read-write rights only to objects with classification=its current clearance
    - read-only rights only to objects with classification<=its current clearance
The Bell and LaPadula Model

- Information flow and access to objects is restricted not only by the access matrix but in addition by the simple security property and the star property.
- The star property supports mandatory access controls.
- The access matrix supports discretionary access controls.
The Bell and LaPadula Model

- State transitions happen via these operations
  - get access
  - release access
  - give access
  - rescind access
  - create object
  - delete object
  - change security level
- Conditions implied by access matrix and star property are enforced before operations can be performed
- Bell-LaPadula showed that these operations maintain the reading down/writing up properties in the system
- Drawbacks
  - static classification/clearances
  - star property can be too restrictive
Lattice Model of Information Flow

- a set of security classes that form a lattice
  - partial order among security classes
  - every set of security classes has a
    - single least upper bound (security class), and
    - a single greatest upper bound

- Each object $x$ has a security class $x$
Military Security Model

Military security model

- objects are
  - ranked in security levels (unclassified, confidential, secret, top secret)
  - assigned to compartments (subject relevance)
- subjects also have security levels and compartments (need-to-know)
- the class of an object is $O=(R_o, C_o)$
- the clearance of an object is $S=(R_s, C_s)$
- $S$ dominates $O$ ($O \leq S$) iff
  - $R_o \leq R_s$ and $C_o$ is subset of $C_s$
Controlling Information Flow

The dominates relation between classes of objects and clearances of subjects defines a partial order that turns out to be a lattice.

Information flows from object x to object y if:

- Information contained in x is used to derive information transferred to y.

Flows can be:

- Direct, eg y := x;
- Or indirect, eg y := (x==1 ? y+1:y);

A flow is permitted only if y dominates the least upper bound of the objects from which information is transferred.