COMPUTER SECURITY

Security policies

```
The company's computers should be used only for work.
```

```
xhost +maq1 -maq2
```

```
<Directory /usr/share/doc>
     Order deny,allow
     Deny from all
     Allow from fe.up.pt
</Directory>
```

```
Grant {
     permission java.net.SocketPermission
     "*:1024-65535", "connect,accept"; };
```

```
Only users of group ADMINISTRATOR can install system software.
```

Security policy: *who can do what, how and when*!

Security policies' basics

Definition of security policy

- specification of requirements for considering a system secure
- declaration of system's secure, authorized states[1]
  - opposed to insecure (or unauthorized) ones
- the set of all elementary security policies (as stated above)

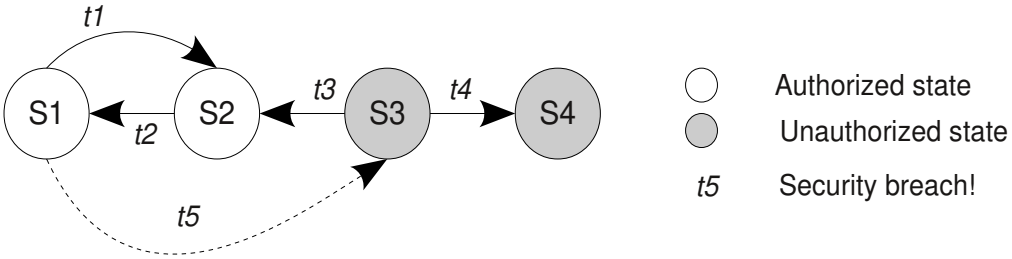


Fig. Example of a security breach.

1 a system state is the collection of values of all systems' components (memory, storage, registers, activities, users...).

Setting up security

- specify security policy
  - who can do what, how and when
  - *specification* should be *complete* and *correct*
- use security mechanisms
  - to enforce the defined policy
  - to prevent the system from entering an insecure state
  - *implementation* should be *complete* and *correct*

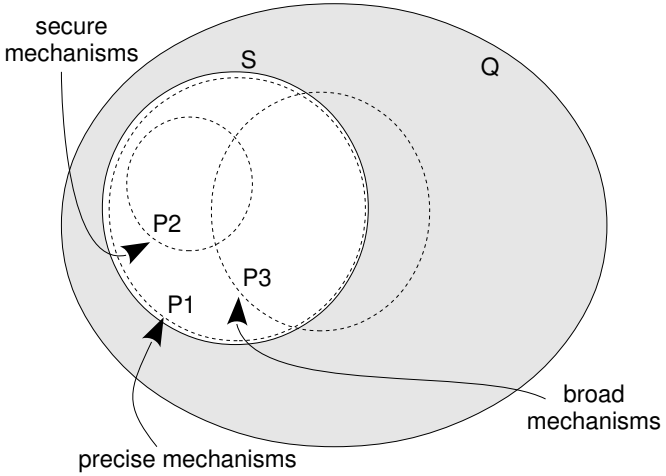


Fig. States of a system and types of security mechanisms.

Q – system's states
S – secure states (security policy)
P – allowed states (security mechanisms)

A classification for security mechanisms (and policies)

- mandatory[1] or rule-based[2]
  - usually, control is imposed by the system
  - (usual acronym: MAC, Mandatory Access Control)
- discretionary[3] or identity-based
  - control is determined by the owner of the info or resource
  - (usual acronym: DAC, Discretionary Access Control)
- combined!...
  - the system enforces certain controls and the owner is allowed to exercise others...

1 PT: geral
2 In some classifications (e.g. security Orange book) these mandatory policies are expected to be based on security marking of users and documents (see below Bell-laPadula's security model).
3 PT: individual

Languages for expressing policies

- normal *versus* technical
- high-level *versus* low-level
  - high-level: kind of abstract language (even mathematical)
  - low-level: language particular to the situations (and even associated to specific security mechanisms)

Examples of policies

```
/usr/bin/xhost +maq1 -maq2
```

```
<Directory /usr/share/doc>
     Order deny,allow
     Deny from all
     Allow from fe.up.pt
</Directory>
```

```
Grant {
     permission java.net.SocketPermission
     "*:1024-65535", "connect,accept"; };
```

```
The company's computers should be used only for work.
```

Definitions

- ***Subject***, *Entity* (or group of entities) - *S*
  - physical person, “active” system's agent[1]
- ***Object*** (or class of objects) - *O*[2]
  - resource that undergoes accesses (or actions) from *Subject*
- ***Information*** (data kept in object) - *I*
  - internal content or state of *Object* to be protected
- ***Action***
  - utilization of *Object* by *Subject* (e.g. "read" object, "delete" object...)

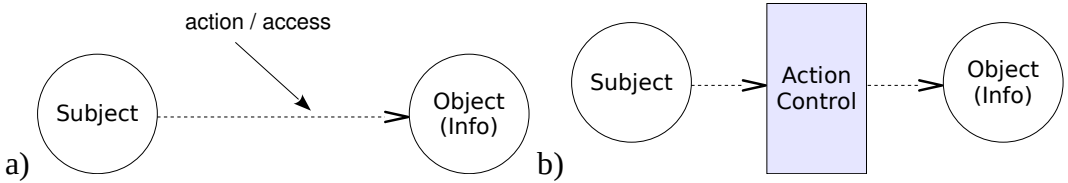


Fig. Acting on an a) unprotected, b) protected object (e.g. to get or change its Information).

1 such a operating system's *process*
2 Sometimes a *Subject* is an *Object* too, as some actions are performed on *Subjects*.

Sequence of access operations on protected object

- authentication of *Entity*
- retrieval of authorization for accessing the *Object*
- control of the *Subject*'s access to the *Object*
  - the *Object* can be accessed directly or through a resource's server

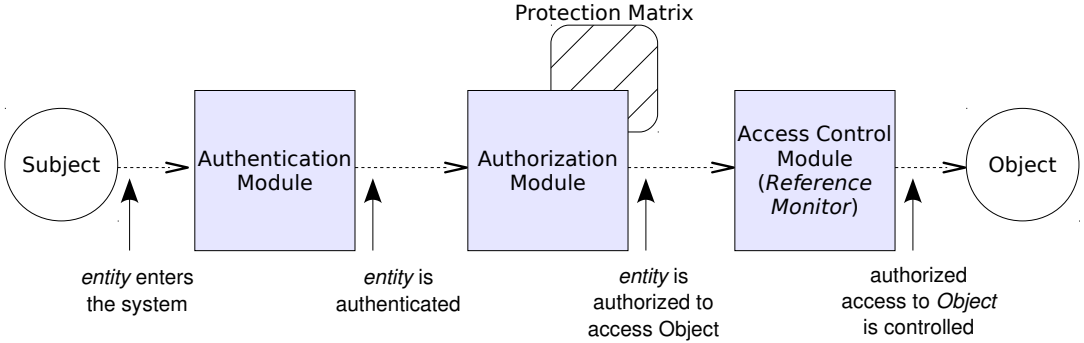


Fig. Recommended procedure for a *Subject* to access an *Object*.

Right, permission, attribute or access mode, ...

- confusing terminology that varies with computer system or theoretical model
  - Ex.: in Unix, (access) *permissions* to files – reading writing, etc. - are part of the files' *attributes*. Other attributes are: ownership, size, creation date…
- we will try to separate:
  - (access) right, permission
    - capacity of *Subject* to perform some action on resource (*Object*)
  - attribute, (access) mode
    - capacity of *Object* to sustain some action from *Subject*
    - type of access (from *Subject*) characteristic of an *Object*
- but sometimes we will intermix the use of the terms, if confusion is unlikely

So, for a Subject to be able to Act on an Object

- the *Subject* needs the appropriate *rights* or *permissions*
- the *Object* must have the corresponding *attributes* or *access modes* (to be able to sustain the action)

Examples of access rights[1] to an object

- observe, read (R): be able to know (the content of) the *Object*
  - so, acquiring its *Information*
- alter, write (W) [2]: be able to change (the content of) the *Object*
  - so, modifying its *Information*
- invoke, execute (X) [3]: be able to use (activate) the *Object*
  - potentially modifying its *Information* state
- append [4]: be able to add more data (*Information*) to (the content of) the *Object*
- delete: be able to remove from scope (or, perhaps, really eliminate) the *Object*

1 or actions it is able to sustain ; beware of variants dependent on security models!
2 in principle, just writing, no reading implied; sometimes, reading is implied...
3 executing, does not necessarily imply reading!
4 does not imply the reading right; important for auditing!

...Definitions: examples of access rights...

Other rights

- modify attributes: be able to change, including remove, the access modes of an *Object*[1]
- concession/exclusion of rights: give to (or remove from) a *Subject* a certain right (to an *Object*)[2]
- exclusion of subject: removal of a *Subject* from a group (with certain rights)[3]
- modify ownership: be able to change the attribute that identifies the *Subject* owner of the *Object*
- create: be able to make new *Objects* of a certain type
- copy: be able to create copies of an *Object* (also known as *grant right*)
- enter: be able to step in certain areas (or states) of the (complex) *Object* and use the associated rights

1 e.g. file F no longer can be read (sometimes, just by some, enumerated, subjects)
Note that the Principle of Attenuation of Privilege should apply: «*A subject cannot give to another subject rights it does not possess*.»
2 e.g. user U can create server (daemon type) processes
3 e,g, user V is removed from the group Z; so, his/hers access rights pertaining to that group will be lost

Examples of subjects to whom the access rights apply

- user: ordinary worker of system
- group: collective user, set of *Subjects* dealt in the same way
- owner: many times, creator (or caller of the creation) of an *Object*
  - sometimes, just the user that gets permission to access the *content* of object [1]
- administrator: controller or installer of the system
- auxiliary administrator: administrator with limited rights
- auditor: verifier or certifier of the system
- visitor: (temporary) anonymous user of the system
- other: user beyond the ones referred to in a specific context
  - Unix example: file `F`, with permissions: `r-- r-- ---`, can be read by the owner and by a member of the owner's group, but not by other users.

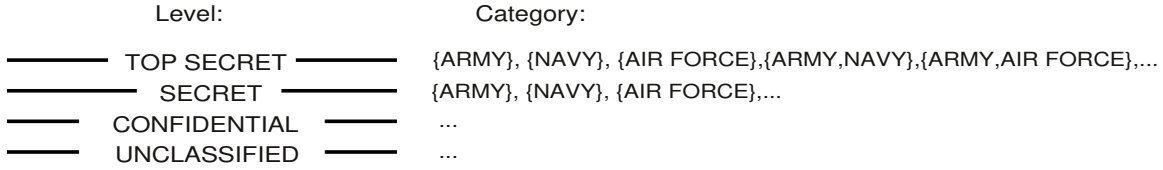
1 see, later, Originator's control policies

Types of security policies

- focus on
  - confidentiality
    - e.g. Bell-LaPadula
  - integrity
    - e.g. Biba
  - availability
    - service access control & response timing (quality of service)
  - other:
    - e.g. originator's control
- most commonly: mixed focus

Bell-LaPadula's confidentiality model[1]

- aims to prevent unauthorized disclosure of information (military interest!)
- is a reference in modeling of computer security (multi-level systems)
- defines:
  - ordered security levels
  - unordered categories[2]
  - membership of *Entities* and *Objects* to *levels* and *categories*
  - rules[3] for *Subjects* accessing *Objects*



1 also called Multi-Level Security (MLS) model
2 also called *compartments*
3 policies

Fig. Illustration of security levels and categories, according to Bell-LaPadula's model.

... Bell-LaPadula's confidentiality model (cont.): illustration of levels and categories

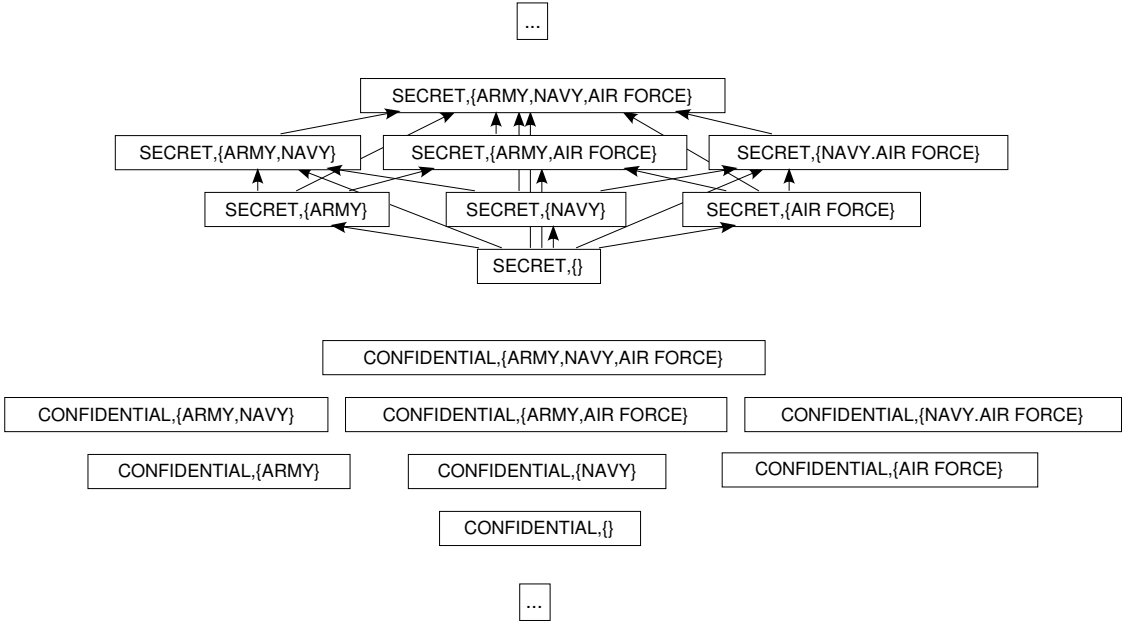


Fig. The arrows (shown only for level SECRET) represent the inclusion relations between categories (X is "included" in Y, X → Y, means information can flow from X to Y).

... Bell-LaPadula's confidentiality model: illustration of levels and categories (cont.)

Example:

- Subject "Adam" *is in* level[1] (SECRET, {AIRFORCE})
- Document "Report X" *is in* level[2] CONFIDENTIAL, {NAVY,AIRFORCE})
- Can "Adam" access "Report X"?

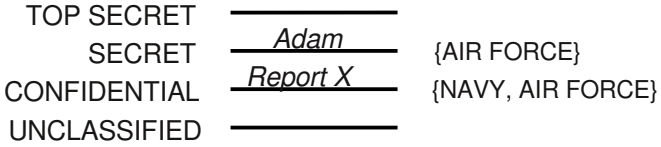


Fig. What type of access will "Adam" have relative to "Report X"? (See answer in later picture.)

1 or: *has clearance* at level
2 or: *is at* level

Bell-LaPadula's security policies

- Simple security condition (*ss - property*)[1]
  - a *Subject* can only observe *documents* in a security level equal or inferior to its own level and of equal or narrower scope (compartments)
- Star condition (* - *property)*[2]
  - a *Subject* can only alter *documents* in a security level equal or superior to its own level and of equal or broader scope
- Discretionary security condition (*ds – property)*
  - a *Subject* may only access *documents* for which it has the corresponding individual access permission

1 mandatory policy
2 mandatory policy

...Security policies of Bell-LaPadula's model (cont.)

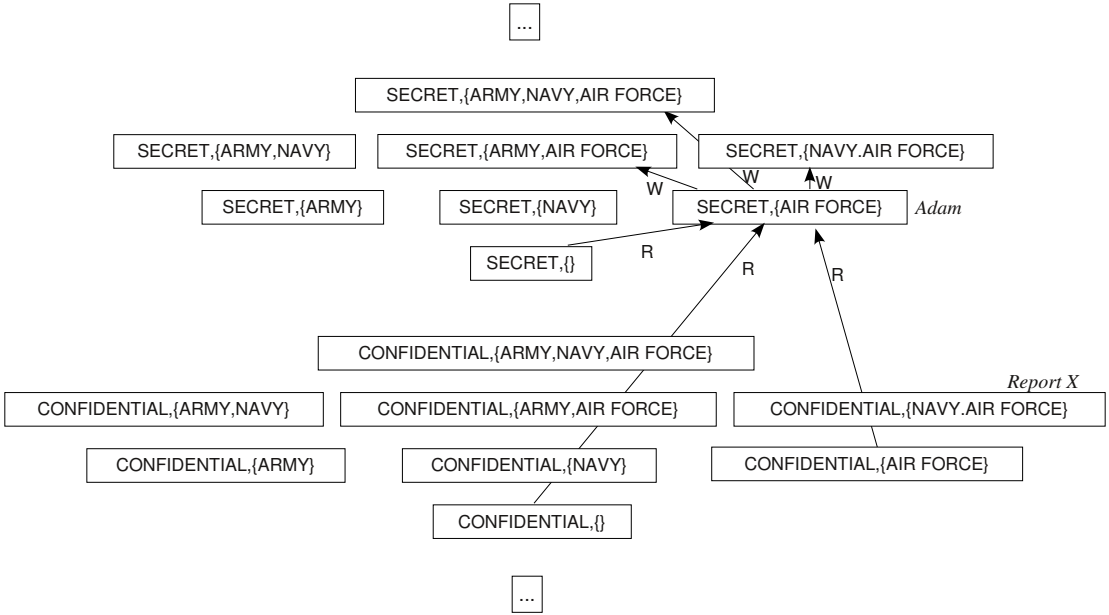


Fig. Overall picture of *Subject* "Adam" and *Object* "Report X" in the security level's tree.
It is seen what "Adam" may only read (R) and write (W) in levels SECRET and CONFIDENTIAL. The arrows show the direction of the information's flow.

...Security policies of Bell-LaPadula's model (cont.)

Comments:

- model's main concern, prevention of unauthorized knowledge,
  - explains the specific prevention of (unauthorized) insertion of information
  - there is no other worrying about the integrity control of objects!
- alteration of security levels
  - necessary for producing documents that are to be “disclosed”
  - appropriate mechanisms are needed
    - for controlling and monitoring of (current and maximum) security levels
    - trusted subjects (allowed to violate some security properties)[1] can exist
- objections and controversy over the model
  - there were many along the years (that is the cost of pioneering);
  - as a result
    - the model suffered adjustments and the concept of "secure state" was refined
    - the meaning of "security" depended on the each system's needs

1 such as Unix's *root*

Biba's integrity model

- aims to ensure confidence in sensitive information (commercial interest!)
  - takes into account the execution of potentially dangerous programs
- defines:
  - ordered security (integrity) levels
    - the higher the level, the greater is the trust on an *Object* and on the actions of a *Subject*
  - unordered categories[1]
  - membership of entities and objects to levels and categories
  - rules for *subjects* accessing *Objects*

Biba's policies:

- of strict integrity (Biba's model)
- ... (*not covered here*)

1 also called *compartments*

Strict integrity policy (Biba's model)

- a *Subject* may only read from *Objects* rated at a superior or equal level
- a *Subject* may only write to *Objects* rated at an inferior or equal level
- a *Subject* may only execute operations on *Objects* rated at an inferior or equal level
- -> dual of LaPadula's policy

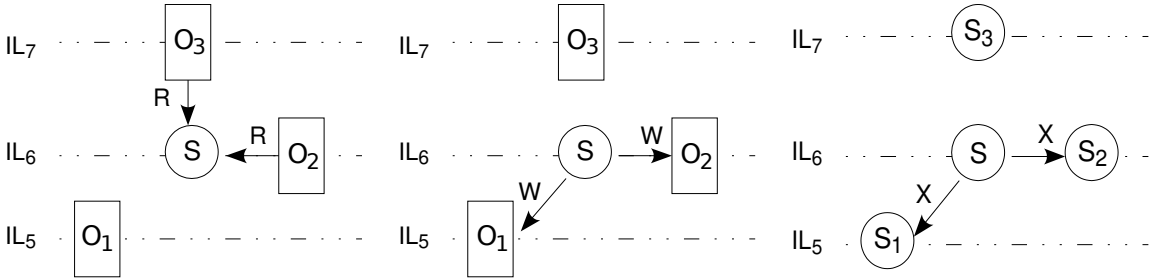


Fig. The different parts of the **strict integrity policy** (the so called **Biba's model**).
(The only difference to other policies, also related to the model, lies in the left side picture!

Lipner 's integrity view

- aims to ensure a sound and secure structural and functional organization of enterprises
- he recommends:
  - separation of duty[1]
    - e.g. production and development tasks should be performed by different subjects
  - separation of function[2]
    - e.g. production and development should have different working environments and data
  - auditing
    - for certification and control, performed by “external” *Subjects*
- he formulates:
  - Lipner's requirements - set of specific integrity requirements for the protection of a system

1 PT: separação de funções - de responsabilidades ou de pessoas
2 PT: separação de ambientes

Lipner's requirements

- R1 – users don't write programs (programmers do)
- R2 – programmers develop and test programs in a separate area from the production environment (using real data, obtained in a controlled way – *sanitized data*)
- R3 - the passage of a program to the production environment (installation, deployment) follows specific rules
- R4 – the process of deploying a program is controlled and audited
- R5 – managers and auditors have access to the system status and all records produced

The Chinese Wall model

- hybrid security policy model
- very close to the enterprise world & current legislation rules
- the "**acquired knowledge**"[1] is very important

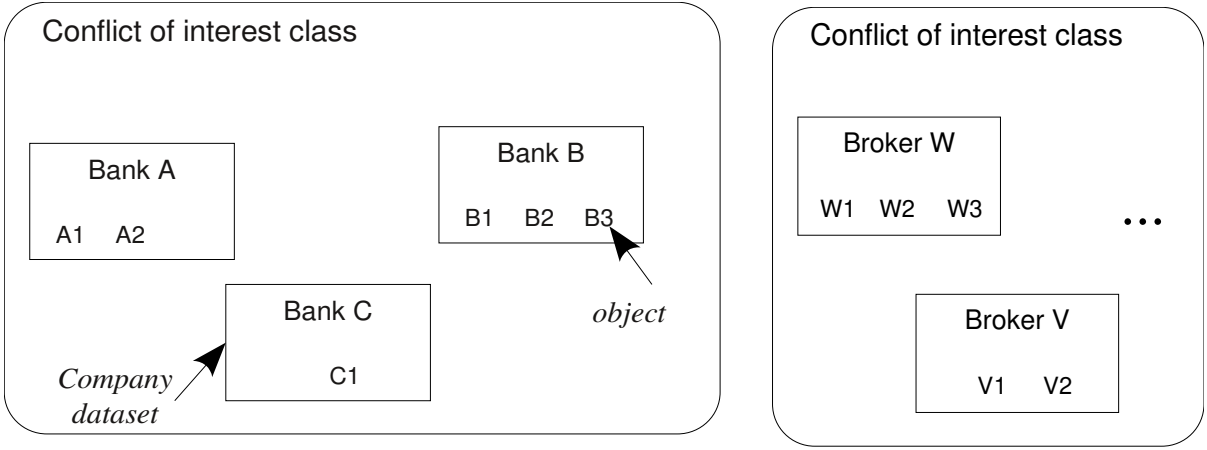


Fig. The Chinese Wall model: *objects*, *datasets* and *conflict of interest classes*.

1 or access history

Policies of the Chinese Wall model

- Simple security condition (ss - property)
  - *Subject S* may read *Object*, if and only if satisfies any of the following:
    - *S* has never read *Objects* of other datasets in the same *conflict of interest class*
    - *S* has only read *Objects* of other *conflict of interest classes*
    - *Object* is sanitized (i.e. purged of sensitive data)
- Star condition (* - *property*)
  - *Subject S* may write to *Object O*, if and only if satisfies **all** of the following:
    - *S* satisfies the "simple security condition"
    - *S* cannot read sensitive *objects* of other datasets, beyond those in the dataset in where s/he intends to write on
      - [*S* cannot read any sensitive O' such that $CD(O') \neq CD(O)$]
      - Note: this condition just says that a subject wanting to write sensitive data, is confined to a single dataset (for writing and reading).

...The Chinese Wall model (cont.): policies

Model's difficulties

- Does a company have so many personnel to abide to correct allocation of tasks, given their past history?
- Subjects are humans, right?...

Example/exercise:

- Suppose that S1 works with Bank A and that S2 works with Bank B. Both S1 and S2 work with Insurance Company C of a different class of conflict of interest than the bank's. Without peeking at the picture below:
  - explain how there could be a security breach, by information leakage, if the only policy taken into account was the "*simple* security condition" of the Chinese wall model.
  - Show how that leak could be prevented, by taking into account (and enforcing) the "*star condition*" policy.
- (Look at the picture below, only after considering your own answer to this problem.)

...The Chinese Wall model (cont.): policies

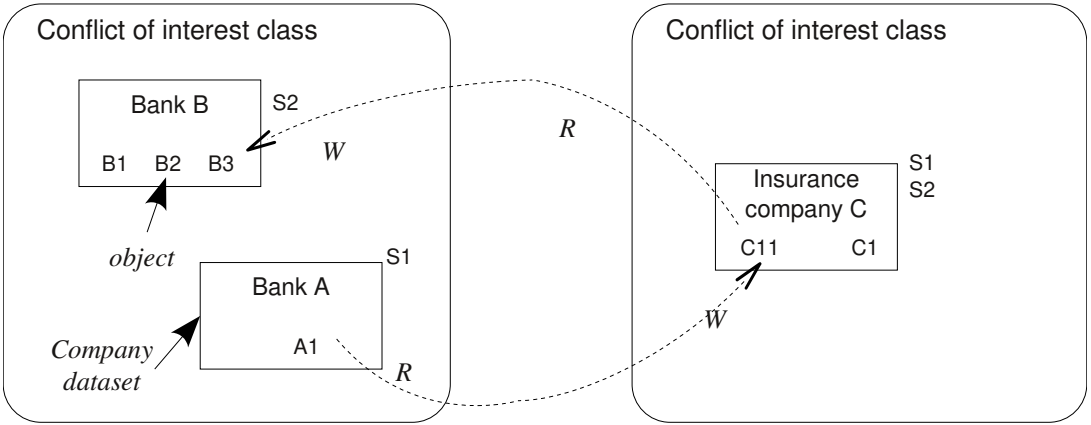


Fig. Illustration of the importance of the "star condition" in the Chinese wall model. (Follows example exercise.)

Other types of policies

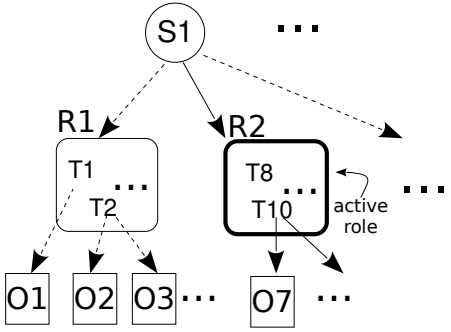
Originator's control policies[1] (or “copyrights' policies“)

- distinguishes the following *Subject*s:
  - who created an *Object*[2] - originator (or author or creator)
  - who currently possesses it - holder (or owner)
  - who can use the *Object*[3] - user
- rules (policies)[4]
  - the current holder of the *Object* cannot alter its access attributes
  - copies of the *Object* keep its original access constraints
  - the *originator* of the *Object* can alter its access attributes (in terms of use and of rights of access (!...)

1 acronym: ORCON, ORiginator CONtrolled...
2 or, somehow, detains special rights over it
3 in predefined ways
4 how to enforce them?...

Role-based model[1]

- access (reading, writing…) to *Object* depends on the current **Role**[2] of the *Subject* (its current duties)
- so, what matters is the *Role*, not the *Subject*
- defines:
  - transaction, task - activity that corresponds to the execution of a certain procedure or set of procedures
  - Role – duties given to a subject so that, by execution of a certain set of tasks, a goal can be achieved
  - active Role – *Subject*'s current *Role*
  - authorized Roles – set of roles that an entity might be invested with



1 corresponding mechanism: RBAC - Role Based Access Control
2 PT: função

Rules (policies) of the role-based model

- Execution of Tasks – a task may only be carried out by a *Subject* with an appropriate (active) Role. (Inversely:)
- Assignment of Roles – a *Subject* may execute a task only when given the appropriate Role
- Authorization of Roles – an *Entity* may only be given an (active) Role from the set of authorized roles
- Incompatibility of Roles[1] – a *Subject* may only assume roles that are not mutually exclusive; the set of mutually exclusive roles will have to be stated and is system-dependent

Model's Difficulties

- What about when a *Subject* changes *Role*?
  - Will there not be a security breach, due to the previously acquired knowledge?...
  - Subjects are humans, right?...

1 principle of separation of duties

Attribute based model[1]

- currently popular in Cloud and Multi-Tenant computing environments
  - allows providers to flexibily enforce strict isolation between different customers sharing the same hardware
  - model is capable of enforcing both Discretionary Access Control (DAC) and Mandatory Access Control (MAC) concepts
- emphasis is on:
  - generalization of *attributes* use
  - relevance of *context*[2] on decisions
  - *dynamism* of attributes
  - set of policies, specified in terms of attributes and conditions

1 corresponding mechanism: ABAC - Attribute Based Access Control
2 or *environment*

...Other types of policies (cont): attribute-based model

Attributes apply ((dynamically) to

- objects (e.g. type, owner, sensitivity...)
- subjects (e.g. job, department, clearence...)
- actions[1] (e.g. read, delete, download...)
- environment (e.g. time, location, situational_status...)

Rules (policies)

- diverse, grouped according to type of computing environments
- mainly logical evaluation of attributes
  - e.g. "Allow *Senior Managers* to *Edit Financial Reports* only if they are on the *Corporate VPN* during *Business Hours*"
  - e.g. "Allow access if { `user.department = resource.department AND user.clearance ≥ resource.sensitivity AND time < 18:00` }"

1 or *operations*

Availability Policy Models

- rules for ensuring a system remains operational and accessible to authorized users
- focus: Resilience, Uptime
  - *Redundancy Models (N+1, 2N)*
    - named after number of "extra" components needed for system to survive a specific number of failures[1]
  - *Service Level Agreements (SLAs)*
    - in business contexts, define expected "uptime" (e.g., 99.9%) and penalties if availability targets are not met
  - *Business Continuity & Disaster Recovery Models*
    - categorize systems based on:
      - Recovery Time Objective - how fast you need to be back up
      - Recovery Point Objective - how much data loss is acceptable

1 e.g. *N+1* model ensures that if one unit fails, the remaining *N* units can still handle the full workload

...Availability Policy Models...

- Examples of model/standards
  - ISO 27001 (Annex.17): *Information Security Aspects of Business Continuity Management*
    - A.17.1: information security continuity
      - Planning, Implementing, Verify, Review & Evaluate
    - A.17.2: redundancies
      - ensure availability of information processing facilities
  - NIST SP 800-34: *Contingency Planning Guide for Federal Information Systems*
    - enabling system recovery after disruptions
      - covers strategies for servers, networks and mainframes to ensure organizational resilience

Availability Mechanisms

- technical and administrative tools used to enforce availability policies

Category	*Mechanisms*	*Purpose*
Technical	Load Balancing	distribution of traffic across multiple servers (DoS prevention)
	Fault Tolerance	hardware redundancy (disk mirroring, UPS...) to keep system running on part's failures
	DDoS Mitigation	tools to filter out malicious, generalized traffic meant to overwhelm and crash services
Physical	Hot/Cold Sites	alternative data centers: *Hot Site* - fully mirrored, ready for instant failover; *Cold Site* - just space with power
	Environmental Controls	fire suppression, cooling systems, backup power generators (physical outages' prevention)
Administrative	Backups	regular, off-site or cloud-based data copies (attack or failure's recovery)
	Patch Management	regularly updating of software (known, crash-capable, bugs prevention)

Policy composition and interference

Real systems' issues

- need for global composite policies
  - large systems have several sub-systems each with own specific policies
- "interference" (or communication) between subjects
  - security policy & mechanisms are sometimes bypassed with special tricks through shared paths: *covert channels*[1]

Base principles of composite systems

- autonomy: a permission granted by one of the system's components, should be honored by the composite system's policy
- security: a denial of access declared by one of the components, should also be honored by the composite system's policy
- safe defaults: initially, no permission is granted; they should be explicitly granted when necessary

1 PT: canais camuflados

Pointers...

- The "**Bell-La Padulla model paper**", 1976 – D. E. Bell and L. J. La Padula
  - csrc.nist.gov/publications/history/bell76.pdf
- The "**Biba model paper**", 1975 – K. J. Biba
  - seclab.cs.ucdavis.edu/projects/history/papers/biba75.pdf
- The "**Lipner integrity model paper**", 1982 – Steven B. Lipner
  - www.cs.washington.edu/research/projects/poirot3/Oakland/sp/PAPERS/00044637.PDF
- The "**Chinese wall paper**", 1989 - David F. C. Brewer and Michael J. Nash
  - www.cs.purdue.edu/homes/ninghui/readings/AccessControl/brewer_nash_89.pdf
- The "**ORCON paper**", 1989 – Richard Graubart
  - www.dtic.mil/dtic/tr/fulltext/u2/a219102.pdf
- The "**RBAC paper**", 1992 – D.F. Ferraiolo and D. R. Kuhn
  - arxiv.org/pdf/0903.2171v2.pdf
- The "**Security Orange book**" (TCSEC), 1985 – DoD 5200.28-STD
  - csrc.nist.gov/publications/history/dod85.pdf
- The "**Guide to Attribute Based Access Control (ABAC)**", 2014 – NIST 800-162
  - nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-162.pdf
- The "**Contingency Planning Guide for Federal Inf. Syst.**", 2010 – NIST 800-34
  - nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-34r1.pdf