GIS-07: Access Control.

Garantia de la Informació i Seguretat [102757]

Guillermo Navarro-Arribas

2020

Departament d'Enginyeria de la Informació i les Comunicacions
Universitat Autònoma de Barcelona

Contents

Access control is the traditional centre of gravity of computer security. It is where security engineering meets computer science.

– R. Anderson [1]

Introduction

Introduction

Objective

To control every access to a system assuring that only authorized accesses can take place.

And access control system:

- Regulates the operations that can be executed on data and resources to be protected.
- Controls operations executed by subjects in order to prevent actions that could damage data and resources.
- It is typically provided as part of the operating system and of the database management system (DBMS).

Access Control and Authorization

- **Access Control** is normally considered to be a two step process:
  1. Authentication: identify who is requesting an action.
  2. **Authorization**: determine if the requester can perform the action.
- Note that sometimes authorization is defined also as the "Access privileges granted to a user, program, or process or the act of granting those privileges" [3].

Access control mechanisms

policy vs. mechanism

Access control **mechanism**: system implementing the access control function.

- Usually part of other systems.
- Uses some access control **policy** to decide whether to grant or deny the subject's request.
- The access control system comprises access control mechanisms and all the information required to take access control decisions.

Entities: Objects and Subjects; and Actions

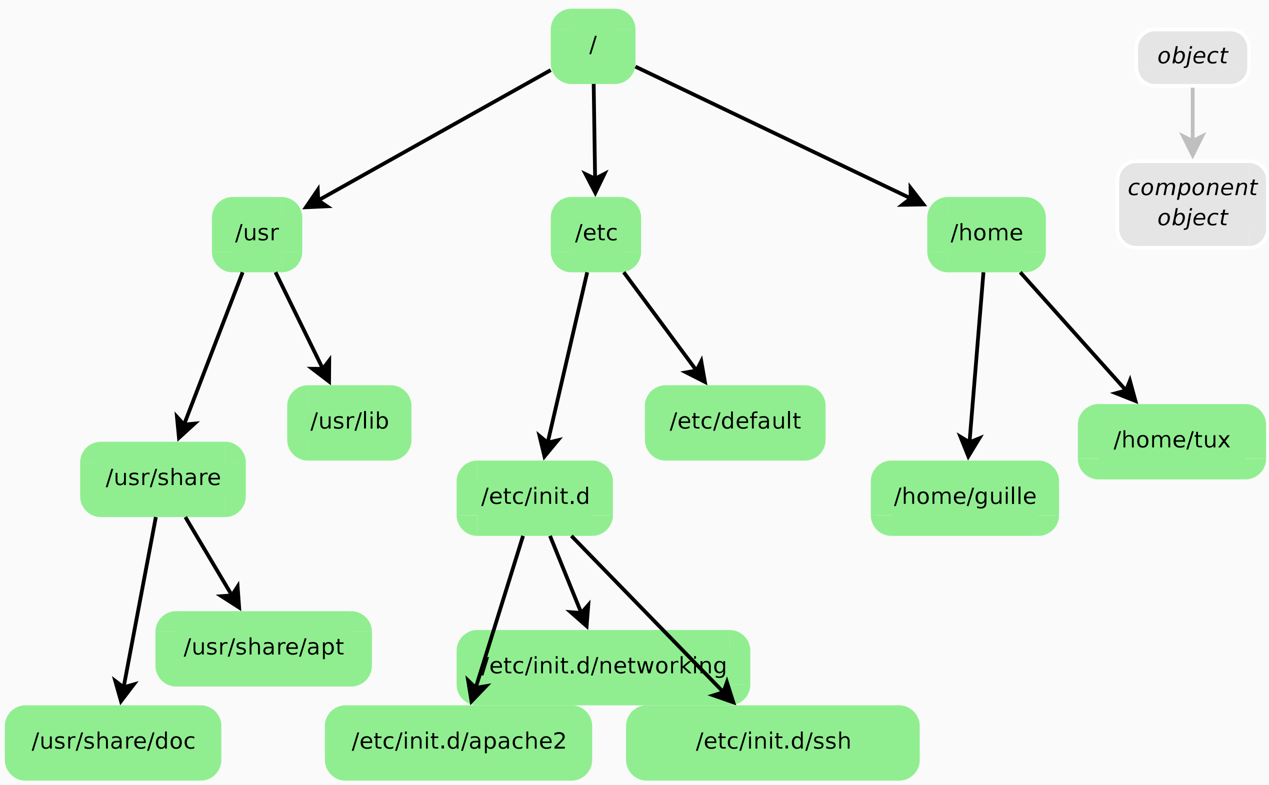
- **Object**
  - Anything that holds data or resources: file system, messages, network packets, I/O devices, physical media, . . .
  - Usually, not all the system's resources need to be protected.
- **Subject / Principal**
  - Abstraction of an active entity that performs computation in the system.
  - A possible classification:
    - users: single individuals.
    - processes: programs executing on behalf of users.
    - groups: sets of users.
    - roles: named collection of privileges / functional entities within the organization.
- **Actions**
  - Operations that a subject can exercise on the protected objects in the system.

Hierarchies and groups

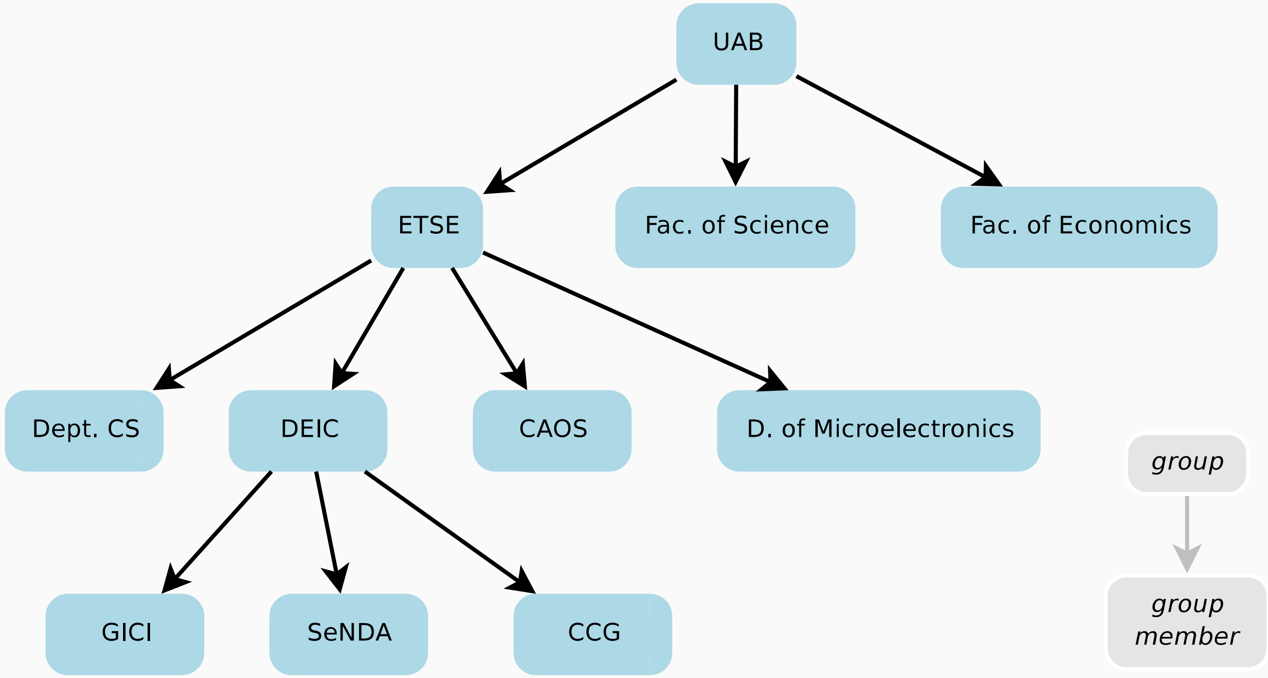
Subjects, objects, and actions can be organized into **groups** with **hierarchies**.

- Reduces the administration cost by reducing the number of permissions that the system has to manage.
- Support the specification of **exception** (by using negative authorizations).

Example of object hierarchy

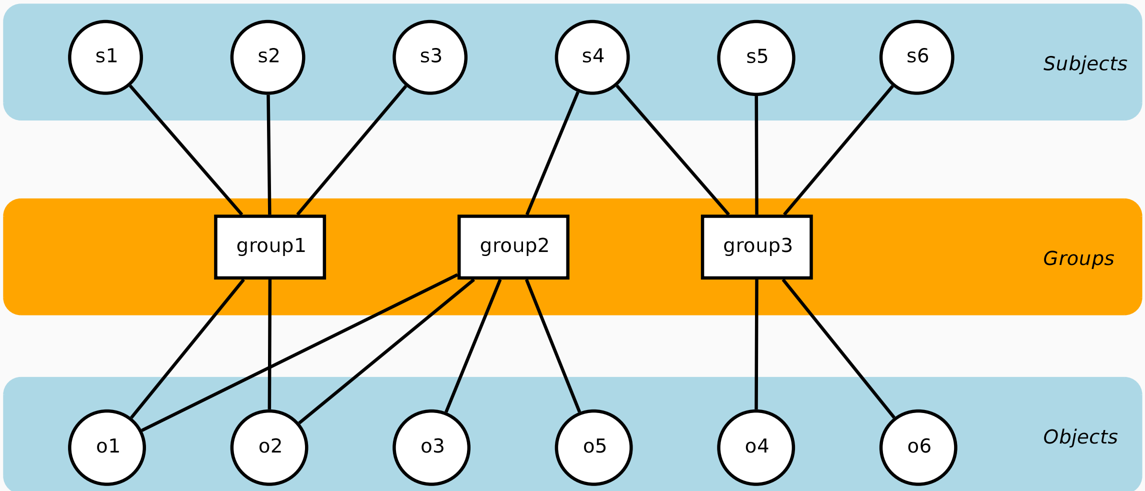


Example of group hierarchy (subjects)



Groups

Groups (without hierarchies) also easy the administration and can be seen as an intermediate level between users and objects:



A note on negative vs. positive permissions

Negative permission

specifies an operation that a subject is not allowed to perform.

- Mixing negative and positive permissions can be tricky.
- Usually policies assume a default, and specify permissions to 'bypass' the default.
  - **Open policy** (default **grant** access): access control rules determine negative permissions.
  - **Closed policy** (default **deny** access): access control rules determine positive permissions.
- If the system supports negative and positive permissions, it needs a **conflict resolution** mechanism.

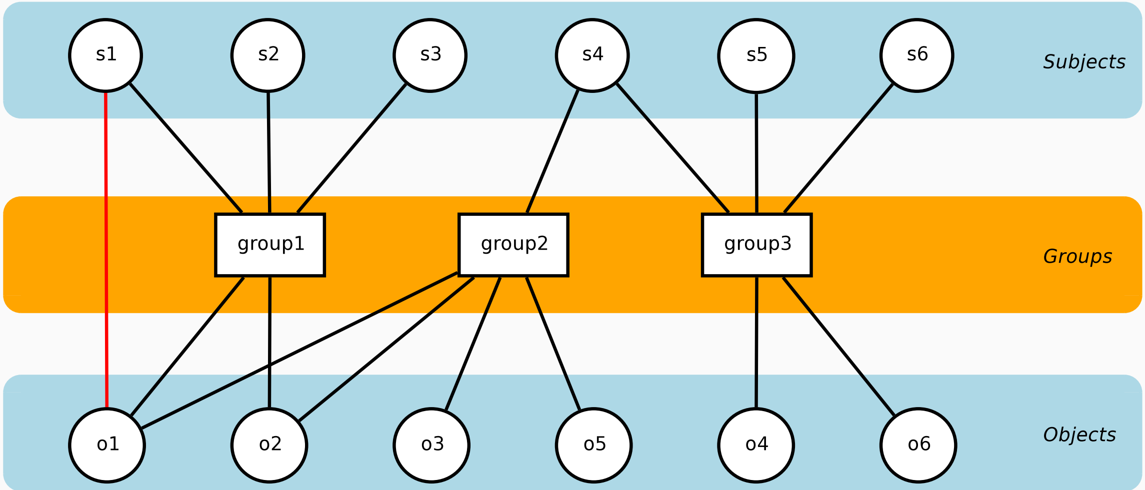
Groups and Negative permissions

Does it makes sense to use negative permissions in closed policies?

In real world situations there may be **exceptions** to a group authorisation management.

- A negative permission specifies an **exception**

Example of negative permissions and groups



Security Policies

We consider here a more specific notion of security policy:

Security Policy

statement that partitions the states of the system into a set of **authorized** (or secure) states and a set of **unauthorized** (or non-secure) states.

- A secure system, starts in an authorized state and cannot enter an unauthorized state.
- The policy defines the rules to change between secure states. That is, the rules determine what the subjects can or cannot do within the system.

Example: Policy context

Security policy normally assumes a non-formal context (laws, organisational polices, . . .)

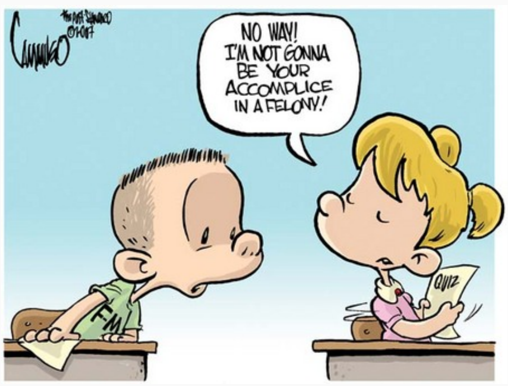
- Example:
  - Policy: disallows cheating (copying homework, with or without permission).
  - Mechanism: file system access permissions.

1. Students do homework on the computer.
2. Alice forgets to read-protect her homework file.
3. Bob copies it.

Example: Who cheated?

→ Who cheated? Alice, Bob, or both?

- Consider the differences between policy and mechanism.



Example: Bob cheated?

- Policy forbids copying homework assignment.
- Bob did it.
- System entered in an unauthorised state.
- If this is not explicit in computer security policy, it is certainly implicit.

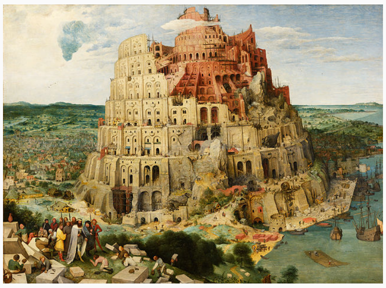
Example: What about Alice?

- Alice didn't protect her homework.
  - But that's not required by the security policy.
- She didn't breach security.
- If policy said students had to read-protect homework files, then Alice did breach security.

Design principles

Saltzer an Schroeder design principles [6]. i

1. **Economy of mechanism**
   - or keep the design simple
   - Sometimes referred as the KISS principle:
     → Keep it simple, stupid!
   - Complexity is one of the largest enemies of security.



2. **Fail-safe defaults**.
   - The default action of the system should be to deny access to someone or something until it has been explicitly granted the necessary privileges
   - Some sensible exceptions apply: life-critical systems, etc.

Saltzer an Schroeder design principles [6]. ii

3. **Complete mediation**.
    - or every object access needs to be authorized.
4. **Open design**.
    - The security of a particular component should not rely on the secrecy of its design.
5. **Separation of privilege**.
    - No individual acting alone can compromise the security of the system.
    - To achieve it, the responsibility for specific tasks is normally divided between several subjects.

Saltzer an Schroeder design principles [6]. iii

6. **Least privilege**.
    - every program and every user of the system should operate using the least set of privileges necessary to complete the job
7. **Least-common mechanism**.
    - minimize the sharing of tools, resources, and systems mechanisms between processes and users.
8. **Psychological acceptability**.
    - create user interfaces that allow users to generate appropriate mental models of the system.

History

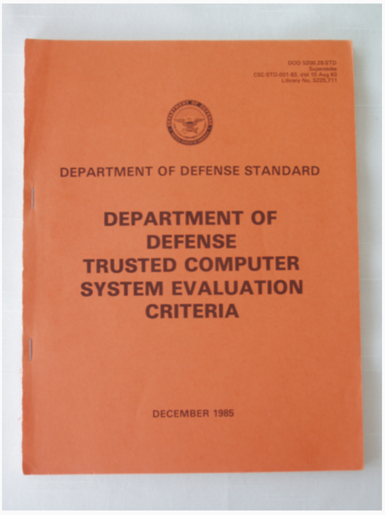
Security System Certification

- Attempt to certify the **security level** of a system.
- It has historical relevance.

The Orange Book

Trusted Computing System Evaluation Criteria (TCSEC), 1983

- By USA DoD (NSA)
- Became very important but now should be considered obsolete.



Orange Book

- Divisions: (lowest) D, C, B, A (highest).
  - **D. Minimal protection**: fail to meet requirements for a higher division.
  - **C. Discretionary protection**:
    - *C1. Discretionary security protection*: enforce access on an individual basis.
    - *C2. Controlled access protection*: more fine grained and includes audit trails.
  - **B. Mandatory protection**
    - *B1. Labeled Security Protection*: data carries a label which determines its authorization.
    - *B2. Structured Protection*: includes covert channel protection.
    - *B3. Security domains*: security code (reference monitor) must be tamper-proof and small enough to be subject to analysis and test.
  - **A. Verified protection**: B3 with formal methods to verify the system functionality.

Common Criteria

Common Criteria, CC

Common Criteria for Information Technology Security Evaluation (ISO/IEC 15408)

- CC appeared by unifying several existing standards (including the Orange Book, with European, and Canadian ones).
- Developed by Canada, France, Germany, Netherlands, UK, and USA.
- Used nowadays to certify security products (mainly intended for government defense and intelligence use).
- Defines 7 Evaluation Assurance Levels (EAL)

CC EALs

- EAL1. Functionally Tested
- EAL2. Structurally Tested
- EAL3. Methodically Tested and Checked
- EAL4. Methodically Designed, Tested, and Reviewed.
- EAL5. Semi-formally Designed and Tested.
- EAL6. Semi-formally Verified Design and Tested.
- EAL7. Formally Verified Design and Tested.

- Government approved laboratories can perform the evaluation: in Spain the CCN (CNI) acredites (`https://oc.ccn.cni.es/`): Applus (EAL5+), Inta (EAL4+), Dekra (EAL4+), Clover (EAL1), .

General information and product catalog:
`http://www.commoncriteriaportal.org/products/`

Access control models

Access control model

- A security model explains what needs to be done, not how to do it.
- A high-level description.

There is a traditional classification of access control models (mainly) derived from the Orange Book.

- They have historical interest and the main concepts are still used by some security people/products/vendors/. . . (although they are currently of dubious utility).
- Three conventional categories:
  - **Discretionary**
  - **Mandatory**
  - **Role-based**

Access control models i

Discretionary Access Control (DAC)

A means of restricting access to objects (e.g., files, data entities) based on the identity and need-to-know of subjects (e.g., users, processes) and/or groups to which the object belongs. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control). [3]

- **Allows access rights to be propagated at subject's discretion**.
- Normally has the notion of **owner** of an object.

Access control models ii

Mandatory Access Control (MAC)

A means of restricting access to objects based on the sensitivity (as represented by a security label) of the information contained in the objects and the formal authorization (i.e., clearance, formal access approvals, and need-to-know) of subjects to access information of such sensitivity. [3]

- Normally implemented with **multi-level security** (MLS) policies, or information flow policies.

Access control models iii

Role-based Access Control (RBAC)

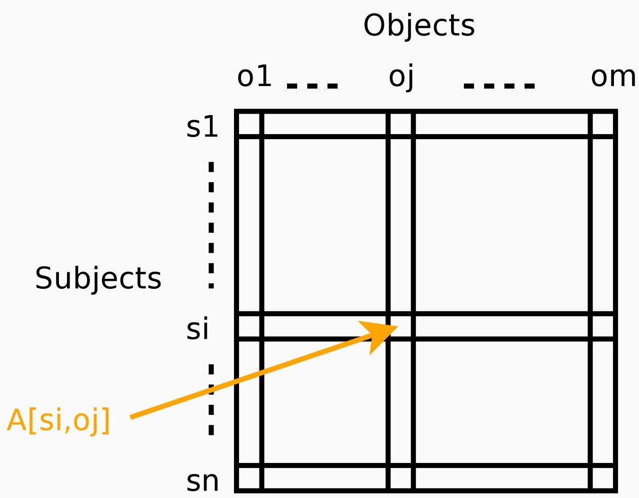
Access control based on user roles (i.e., a collection of access authorizations a user receives based on an explicit or implicit assumption of a given role). Role permissions may be inherited through a role hierarchy and typically reflect the permissions needed to perform defined functions within an organization. A given role may apply to a single individual or to several individuals. [3]

Access Control Matrix

Access Control Matrix

- The **access control matrix** is the most precise model of a protection state.
  - Transitions ⇒ change elements of the matrix.

- Subjects, $S = \{s_1, \dots, s_n\}$
- Objects, $O = \{o_1, \dots, o_m\}$
- Rights, $R = \{r_1, \dots, r_k\}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{r_x, \dots, r_y\}$: subject s_i has rights $r_x, \dots, r_y$ over object o_j.



Access Control Matrix Example

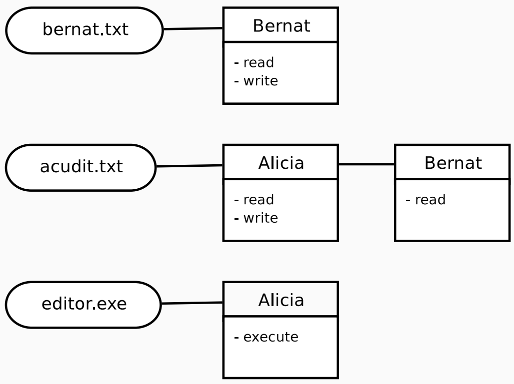
File system access:

	bernat.txt	**acudit.txt**	**editor.exe**
Alicia	-	{read, write, own}	{execute}
Bernat	{read, write, own}	{read}	-
Carolina	-	{read}	-

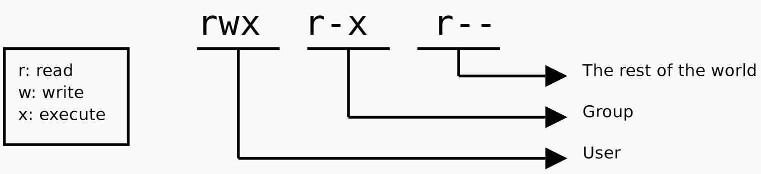
Implementation of the access control matrix

- The access control matrix is an abstract model.
- Two common implementations of the matrix:
  - **Access control lists**: list of users with actions or permissions for each object.
  - **Capabilities**: List of objects with actions or permissions for each user.

Access Control List



- Simple example: permissions in UNIX file system.



POSIX Extended Access Control List i

```
$getfacl myfile
# file: myfile
# owner: prince
# group: admin
user::rwx
group::r-x
other::r-x
```

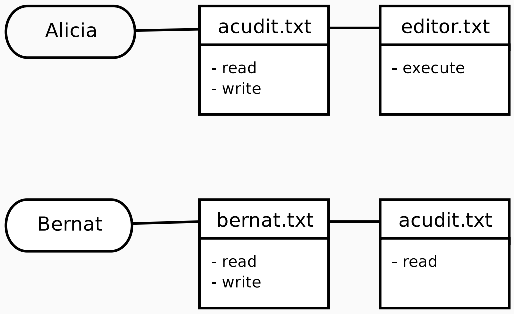
POSIX Extended Access Control List ii

```
$setfacl -m user:sara:rwx myfile
$getfacl myfile
# file: myfile
# owner: prince
# group: admin
user::rwx
user:sara:rwx
group::r-x
other::r-x
$ls -l myfile
-rw-rwxr--+ 1 daniel admin 2 Mar 19 15:53 myfile
```


Advantages of Access Control Lists

- Preferable when users manage their own files.
- Easy to change rights to a particular object.
- Relatively easier to implement (are more often used in practice than capabilities).

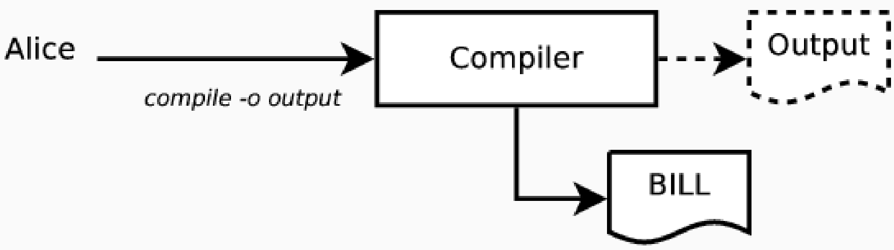
Capabilities



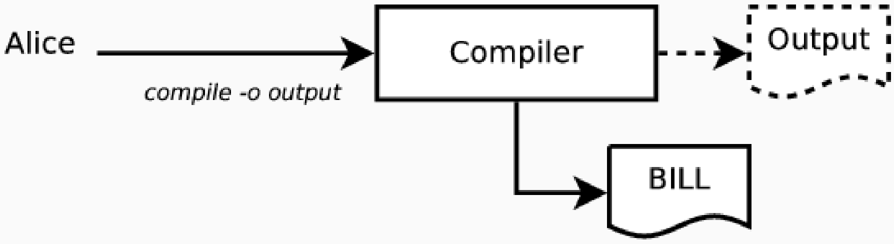
- A capability can be seen as a token associated to the user/process.

The Confused Deputy Problem i

- Pay-by-use service: compiler.
- Billing file: BILL.
- User: Alice.
- Compiler service is called with the output file as parameter.



The Confused Deputy Problem ii



	Compiler	BILL
Alice	x	–

Compiler should be able to w BILL
but Alice shouldn't

The Confused Deputy Problem iii

- What if ... Alice calls the compiler as `compile -o BILL`?
- What privileges uses the Compiler when it is executed by Alice?

→ the compiler (deputy) is confused! (has two masters)

- E.g. consider the `passwd` command in UNIX-like systems. It is executed by a user but needs to write to `/etc/shadow`. How is this solved?
- Capabilities (easily) solve this problem by associating the proper capability to each operation.

Advantages of Capabilities

- Solve the confused deputy problem.
- Easy to implement least privilege.
- Easier to delegate.
- Easier to add/delete users in the system.

Common DAC-like models

DAC Models i

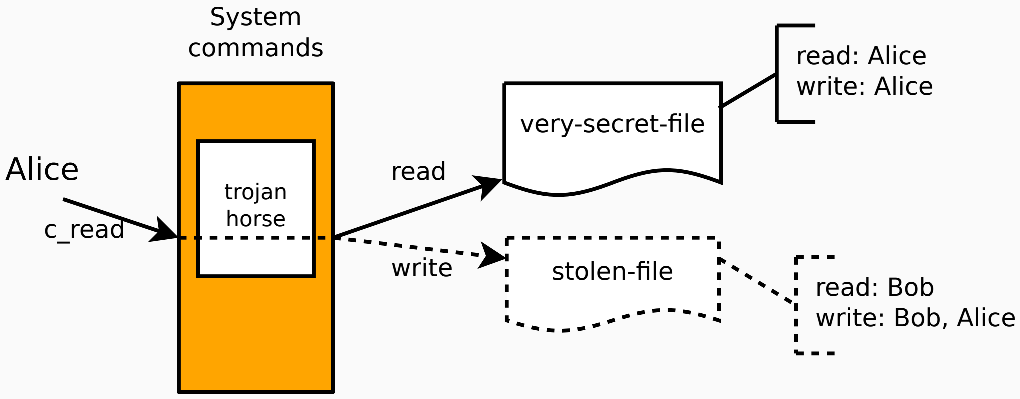
- Govern the access of subjects to objects on the basis of subjects' identity, objects' identity, and permissions.
- When an access request is submitted to the system, the access control mechanism verifies whether there is a permission authorizing the access.
- Such mechanisms are discretionary in that they **allow subjects to grant other subjects authorization to access their objects at their discretion**.
- Advantages
  - Flexibility in terms of policy specification.
  - Supported by all OS and DBMS.
- Drawback
  - No information flow control (Trojan horse attacks).

DAC Models ii

- Normally a relatively straight forward implementation of the access matrix as ACL.
- First well known DAC model: HRU model (Harrison, Ruzzo, Ullman)
  - provided 6 **primitive operations on the access control matrix**:
    - add object
    - add subject
    - add permission
    - remove object
    - remove subject
    - remove permission

Classical DAC problem: Trojan horse

- DAC models are unable to protect data against Trojan Horses embedded in application programs.



- MAC models were developed to prevent this type of illegal access.

Multilevel Security Models

Mandatory access control

- MAC specifies the access that subjects have to objects based on subjects and objects classification.
- Nowadays better known as **multilevel security (MLS)**, or **information flow policies**.
- Many of the MLS have been designed based on the Bell and LaPadula (BLP) model [2].

The Bell-LaPadula Model

- Elements of the model:
  - **objects**: passive entities containing information to be protected.
  - **subjects**: active entities requiring accesses to objects (users, processes).
  - **access modes**: types of operations performed by subjects on objects (we only consider read/write for simplicity)
    - **read**
    - **write**

Levels

- **Subjects** are assigned **clearance** levels and they can operate at a level up to and including their clearance levels.
- **Objects** are assigned **sensitivity** levels.
- The clearance levels as well as the sensitivity levels are called **access classes**.

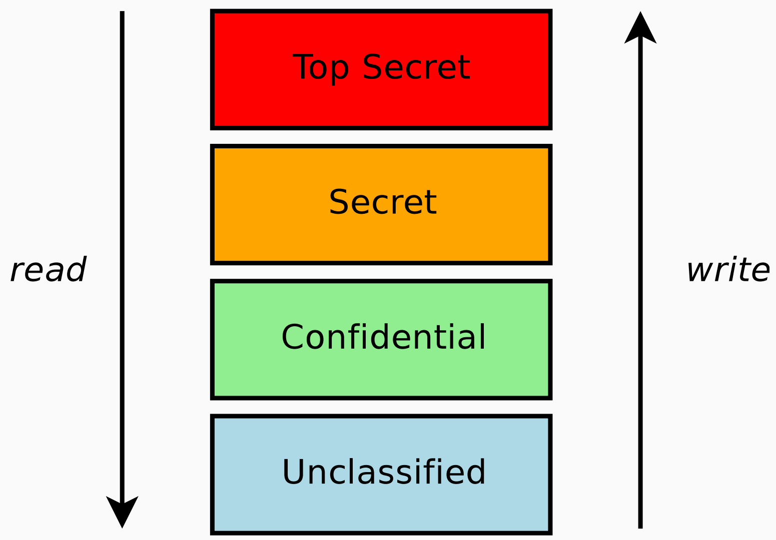
Access Classes

- An access class consists of two components:
    - A **Security level** (L): element from a totally ordered set: $L =\{$ Top Secret $>$ Secret $>$ Confidential $>$ Unclassified $\}$
    - A **category set** (SC): set of elements, dependent from the application area in which data are to be used. Also known as compartments: $SC =\{$Army, Navy, Air Force, Nuclear$\}$
- For simplicity we will consider only security levels here.
    - $L(s) = \textit{secret}$: security level of subject s is secret.
    - $L(o) = \textit{confidential}$: security level of object o is confidential.

BLP axioms

- **Simple security property $\rightarrow$ no-read-up**
  - **Subjects cannot read data to upper levels.**
  - s can read o if and only if $L(o) \leq L(s)$.
- ***-property $\rightarrow$ no-write-down**
  - **Subjects cannot write data to lower levels.**
  - s can write o if and only if $L(o) \geq L(s)$.

Access rule simplification example



Covert channels in Bell-LaPadula

A very simplistic and naive example:

- General Patton with "Secret" *clearance* attempts to write a document named *new-plan-to-send-Patton-To-kurdistan.txt*.
- The document exists but has level "Top Secret"
- The write (or creation) fails (file already exists) $\Longrightarrow$ Now, Patton knows that there is a document named *new-plan-to-send-Patton-To-kurdistan.txt* at the "Top Secret" level.

Consideration on Bell-LaPadula

- BLP is the base for most MLS nowadays.
- In general, the model is considered too rigid for generic corporate environments.
- Mostly used in military-like environment (easy to establish authority, high-security systems, . . .).
  - But also in highly secret corporate environment documentation management, network firewalls, medical information, . . .

RBAC

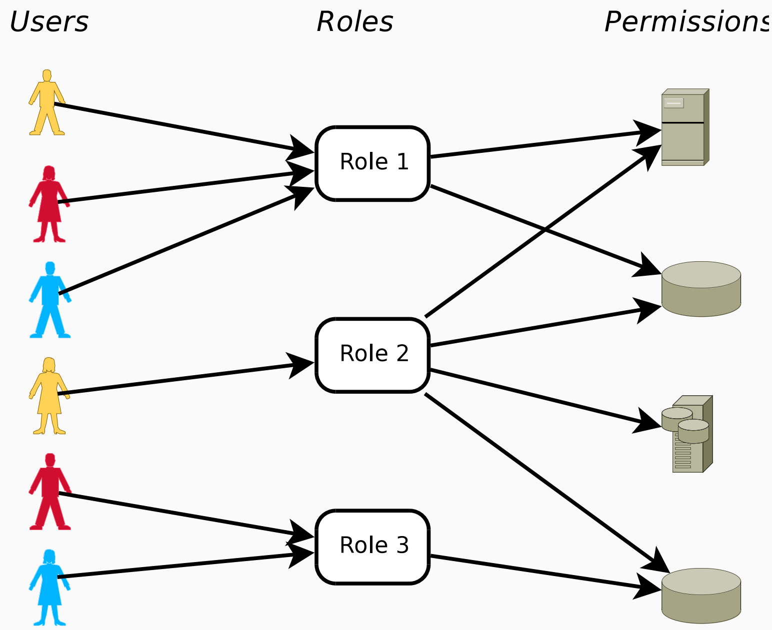
RBAC: Basic concepts [4]

- **Role**: a function within the context of an organization with an associated semantics regarding its authority and responsibility.
- **User**: a human being, a machine, a process, or an intelligent autonomous agent, etc.
- **Session**: a particular instance of a connection of a user to the system and defines the subset of activated roles.

⇒ Users are thus simply authorized to "play" the appropriate roles in a given session, thereby acquiring the roles' authorizations.

role ≠ group

RBAC access control



RBAC Benefits

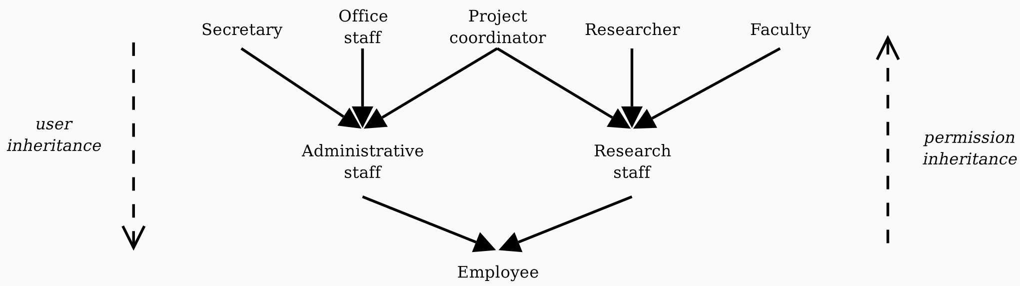
- Because roles represent organizational functions, an RBAC model can directly support security policies of the organization
- Granting and revoking of user authorizations is greatly simplified
- There is some consensus on a standard RBAC model
    - Most popular standard for RBAC: NIST RBAC model: `http://csrc.nist.gov/groups/SNS/rbac/`

RBAC NIST Model

- Three main levels of increasing functional capabilities:
  - **Core RBAC** (also called Flat RBAC): simple model, with roles users and permissions.
  - **Hierarchical RBAC**: adds support for role hierarchies.
  - **Constrained RBAC**: adds support for constraints.

Hierarchical RBAC

- Role hierarchies are a natural means for structuring roles to reflect an organization's line of authority and responsibility.



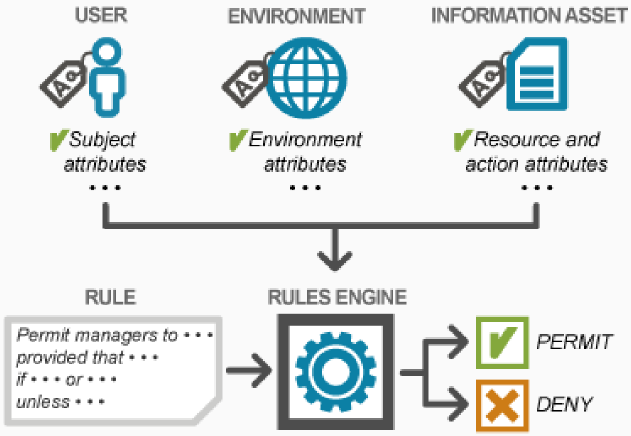
Constrained RBAC

- Constrained RBAC is an RBAC model with the capability of supporting **Separation of Duties** (SoD) policies
- Defines sets of mutually exclusive roles (a user cannot be assigned or activate more than one role in the set).

Attribute based access control

Attribute based access control i

- Attribute based access control (ABAC): determines access based on attributes of the subject, object and environment.



Source: Axiomatics (`https://www.axiomatics.com/`)

Attribute based access control ii

- Example: XACML (eXtensible Access Control Markup Language): XML-based standard policy language for ABAC.
- Example of common ABAC rules:
  - Any user with an e-mail name in the "med.example.com" namespace is allowed to perform any action on any resource between 8:00 and 22:00.
- Can be seen as a generic model
- NIST provides a guide for ABAC [5].

References

References i

[1] Ross Anderson.
Security Engineering: A Guide to Building Dependable Distributed Systems.
Wiley, 2nd edition edition, 2008.
`http://www.cl.cam.ac.uk/~rja14/book.html`.

[2] D. E. Bell and L. J. LaPadula.
Secure computer system: Mathematical foundations.
Technical Report ESD-TR-278, vol.1, The Mitre Corp., 1973.

[3] Committee on National Security Systems.
National Information Assurance (IA) Glossary.
CNSS Instruction No. 4009, `http://www.ncix.gov/publications/policy/docs/CNSSI_4009.pdf`, April 2010.

[4] D. Ferraiolo, D. Kuhn, and R. Chandramouli.
Role-Based Access Control.
Artech. 2nd ed., 2007.

[5] Vincent C. Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone.
Guide to attribute based access control (ABAC) definition and considerations.
NIST Special Publication 800-162,
`http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf`, January 2014.

[6] J.H. Saltzer and M.D. Schroeder.
The protection of information in computer systems.
Proceedings of the IEEE, 63(9):1278–1308, September 1975.

C-x C-c

C-x C-c

Guillermo Navarro-Arribas