SAPIENZA
UNIVERSITÀ DI ROMA

"Sapienza" University of Rome
Faculty of Information Engineering, Informatics, and Statistics
Departmet of Computer Science

Cybersecurity

Lecture notes integrataed with the book "Computer Security: Principles and Practice", W. Stallings, L. Brown

Author
Simone Bianco

6 gennaio 2024

Indice

Information and Contacts

Personal notes and summaries collected as part of the *Cybersecurity* course offered by the degree in Computer Science of the University of Rome "La Sapienza".

Further information and notes can be found at the following link:
https://github.com/Exyss/university-notes. Anyone can feel free to report inaccuracies, improvements or requests through the Issues system provided by GitHub itself or by contacting the author privately:

- Email: **bianco.simone@outlook.it**
- LinkedIn: **Simone Bianco**

The notes are constantly being updated, so please check if the changes have already been made in the most recent version.

Suggested prerequisites:

Preventive learning of material related to the *Computer networks* course is recommended

Licence:

1

Introduction to Cybersecurity

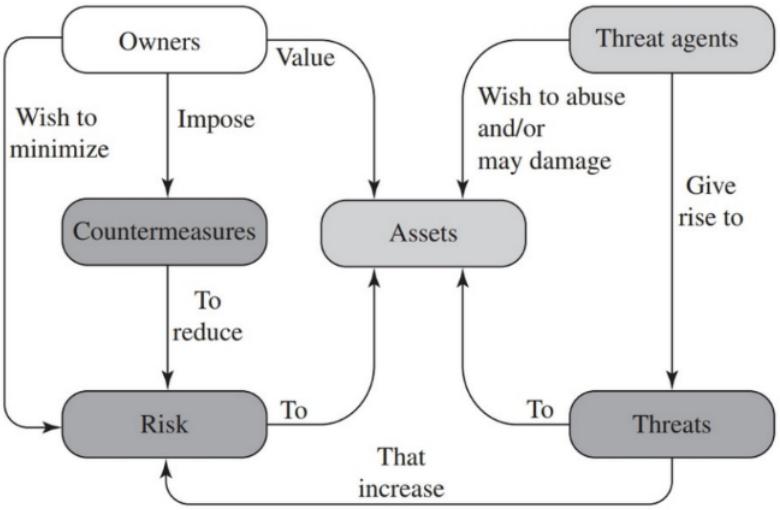
1.1 Fundamental concepts

The National Institute of Standards and Technology (NIST) defines **computer security** as the prevention of damage, protection and restoration of computers, electronic communications systems and services and any other type of digital structure.

In this course, we define **computer security** as measures and controls that ensure **confidentiality**, **integrity** and **availability** of information system assets including hardware, software and information being processed, stored, and communicate.

In order to talk about cybersecurity, first we have to give the following **essential definitions**:

- **Threat**: any circumstance or event with the potential to adversely impact organizational operations
- **Threat agent** (or *Adversary*): anyone who conducts or has the intent to conduct detrimental activities
- **Countermeasures**: a device or a technique that has the objective of impairing detrimental activities
- **Risk**: a measure of the extent to which an entity is exposed to a thread, such as the impact that would arise if an unaccounted event occurs and his likelihood of occurrences
- **Vulnerability**: weakness in an information system, internal controls, implementation, etc... that could be exploited or triggered by a thread source



Osservazione 1

The security of a system, application or protocol is always relative to the set of desired properties and the capabilities of the potential threat agent

Example:

- Standard file access permission in Linux or Windows systems are not effective against an adversary who can boot the system from a CD

Definition 1: Types of attacks

In order to distinguish between kinds of threats, we define the following **types of attack**:

- **Active attack**: an attempt to alter system resources or affect the operation.

  In particular, we establish four categories of active attack: **replay**, **masquerade**, **modification of messages** and **denial of service**

- **Passive attack**: an attempt to learn or make use of information from the system that does not effect the system resources

  In particular, we establish four categories of passive attack: **release of message contents** and **traffic analysis**

- **Inside attack**: initiated by an entity inside of the system's *security perimeter*, namely an **insider** who is authorized to access the system resources, using them in an unapproved way

- **Outside attack**: initiated by an entity outside of the system's *security perimeter* who is

1.2 Confidentiality, Integrity and Availability (CIA)

Definition 2: Confidentiality

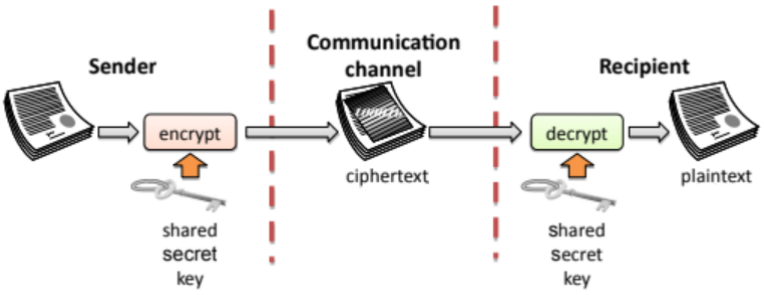
We define **confidentiality** as the avoidance of the unauthorized disclosure of information

Example:

- Confidentiality involves the protection of data, providing access for those who are allowed to see it while disallowing others from learning anything about its content

In order to **ensure** confidentiality is preserved, three main tools are used:

- **Encryption**: the transformation of information using a secret called *encryption key* in order to make the transformed information readable only by those who know another (or the same) secret, namely the *decryption key*



- **Access control**: rules and policies that limit access to confidential information to established people and/or systems
- **Authentication**: the determination of the identity or role that someone has, usually done through a number of different factors, such as something the person has, knows or is
- **Authorization**: the determination if a person or system is allowed to access resources based on an policy
- **Physical security**: the establishment of physical barriers to limit access to protected computational resources

Definition 3: Integrity

We define **integrity** has the property that something must not be altered in an unauthorized way

Examples:

- Integrity involves the use of backups, checksums, data correcting codes, etc...

Definition 4: Availability

We define **availability** as the property that something is accessible and modifiable in a timely fashion by those who are authorized to do so

Examples:

- Availability involves the use of physical protections and computational redundancies

The concepts of confidentiality, integrity and availability establish what is know as the **CIA security triad**. In order to be secure, a system should try to minimize the number of fallacies that conflict with the triad.

However, other concepts are used to describe the security of a system:

- **Authenticity**: the ability to determine that statements, policies and permission issued by a person are genuine.
- **Accountability**: the requirement for actions of an entity to be traced uniquely back to that same entity through the use of activity records
- **Anonymity**: the property that certain records or transactions are not to be attributable to any individual

1.3 Threat consequences and types

We can categorize events based on their ability to pose a threat on one or more concepts of the CIA triad or based on the type of attack implied by those events.

The first categorization can be reduces to the following types of events:

- **Unauthorized disclosure**: a circumstance or event whereby an entity gains access to data for which the entity is not authorized. This type of event is a threat to **confidentiality**
- **Deception**: a circumstance or event that may result in an authorized entity receiving false data and believing it to be true. This type of event is a thread to either **system integrity** or **data integrity**
- **Disruption**: a circumstance or event that interrupts or prevents the correct operation of system services and functions. This type of event is a threat to **availability** or **system integrity**
- **Usurpation**: a circumstance or event that results in control of system services or functions by an unauthorized entity. This type of event is a threat to **system integrity**

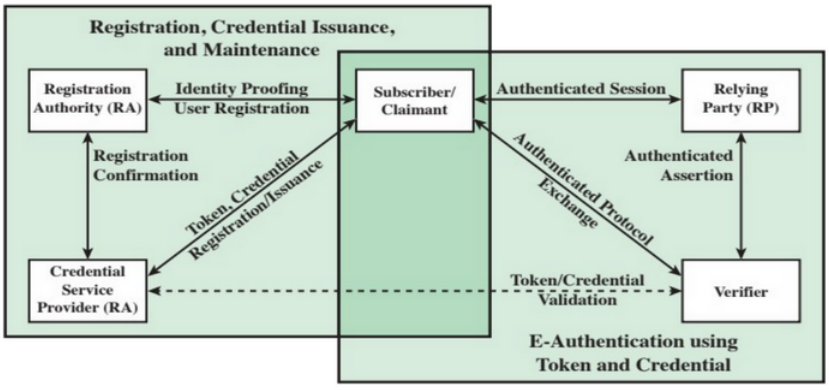
Instead, the second categorization can be reduces to the following types of attacks:

- **Interception**: the eavesdropping of information intended for someone else during its transmission over a communication channel
- **Falsification**: unauthorized modification of information, such as the *man-in-the-middle attack*, where a network stream is intercepted, modified and retransmitted to the original receiver
- **Denial of service (DoS)**: the obstruction or degradation of data service and/or information access
- **Masquerading**: the fabrication of information that is supposed to be from someone who is not actually the author
- **Repudiation**: the denial of commitment or data reception, such as the attempt to back out of a contract or protocol that requires the different partied to provide receipts acknowledging that data has been received
- **Inference** (or *correlation/traceback*): the integration of multiple data sources and information flows to determine the source of a particular data stream or piece of information

1.4 Authentication

As we already discussed, authentication can be described as the process of establishing confidence in the user identities that are presented electronically to an information system through the use of:

- Something the individual **knows**, such as a password or a PIN
- Something the individual **possesses**, such as a token or a key card
- Something the individual **is**, such as biometrics (fingerprints, iris, face, ...)
- Something the individual **does**, such as dynamic biometrics (handwriting, voice pattern, ...)



The use of more than one of these authentication means is called **multifactor authentication**, ensuring greater security as the number of methods used increases.

One of the most common means of authentication is the use of **passwords**. Usually, the user provides a name and a password, which then get compared by the system with the ones stored in their memory. The **user ID** determines that the user is authorized to access the system and the his privileges.

Definition 5: Hash function

An **hash function** is a one-way-function (meaning that it irreversible) capable of converting a string of plain text into an incomprehensible string of text of fixed length called **hash**

Since they are impossible to reverse, the best way to store passwords is through the use of **hash functions**. A good hash function must be capable of being efficient to compute while also being able to minimize the possibility of two string **colliding** into the same hash.

Definition 6: Salt

We define as **salt** a random string fed as an additional input to an hashing function by getting attached to the original input before being hashed.

The use of salts ensures that the input becomes sufficiently large, making the output more secure

Example:

- Modern UNIX systems store passwords by looping 1000 iterations of MD5 hash function with a salt of up to 48 bits, producing a 128 bit hash value

By **storing the hashed password**, one can check if the given password is correct simply by hashing it and then check if it matches the stored hash.

Many programs are used to **crack password** by exploiting the fact that people usually choose easily guessable password and/or short passwords, making it easy to **brute force** through the use of a cracking software:

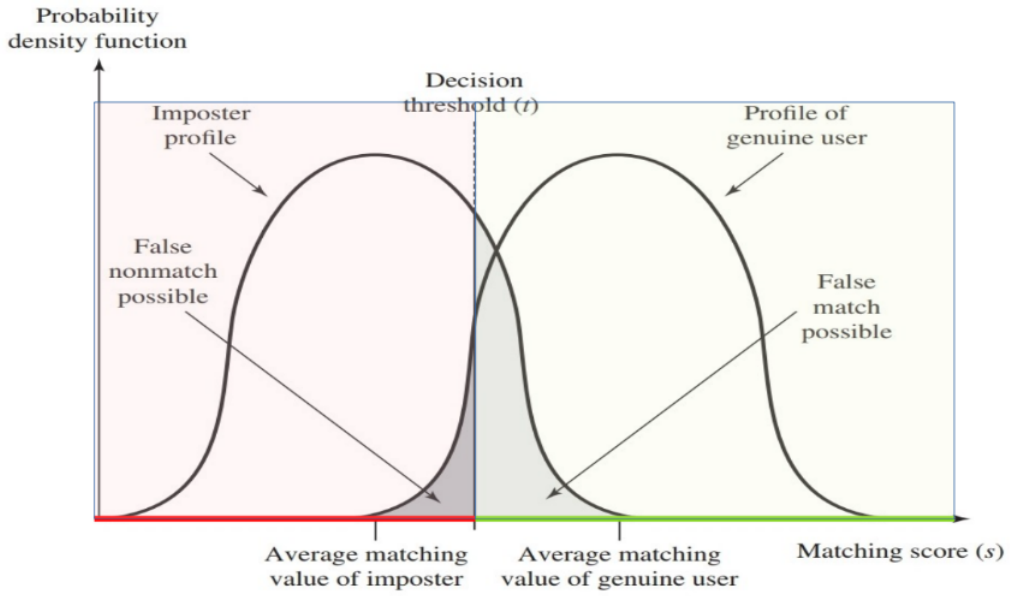
- **Dictionary attacks** are based on a large list of possible passwords, testing them one by one. Each password must be hashed using each different salt value and then compared to the stored hash values.
- **Rainbow table attacks** are based on pre-computed enormous tables of hash values fro all salts. This attack can be countered by using a sufficiently large salt value and a sufficiently large hash length

Definition 7: Token

We define as **token** a small string of text able to identify an user or an entity

Common examples of tokens include barcodes, magnetic stripe cards, smart tokens and smart cards realized through the use of RFID technology.

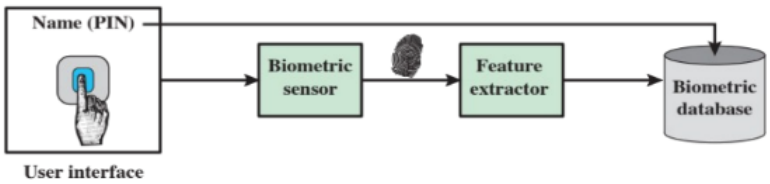
Less common examples of tokens include biometrics: the data gets read and then converted to a *reference vector*, which then gets compared to the stored one through the use of matching techniques based on *similarity* (since a perfect copy is never possible to replicate).



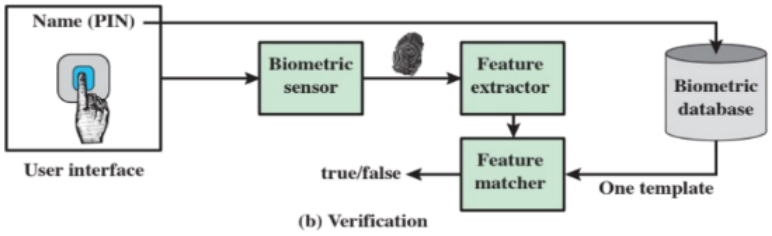
Biometrics such as voice and face recognition are usually low cost with low accuracy, while biometrics such as iris and fingerprint scanning are medium to high cost while also being pretty accurate.

Biometric authentication systems usually involve one or more of the following three types of operations:

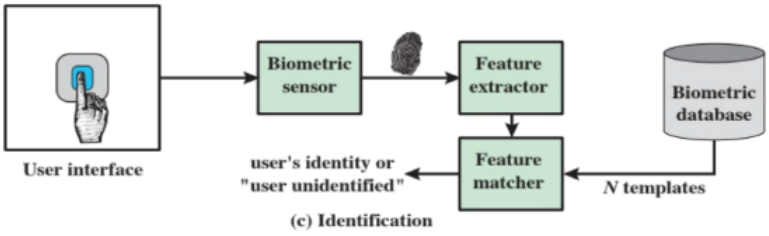
- **Enrollment**: the user registers his biometric data through the use of a PIN and a scanner



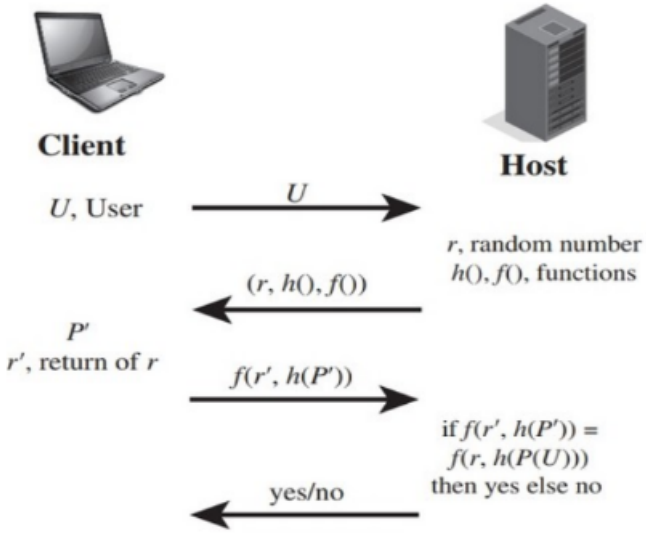
- **Verification**: the user gets recognized by giving the registered PIN and his biometric data by matching. Requires a previous registration and one sample of the user's biometric data



- **Identification**: the user gets identified by giving only his biometric data. Requires a previous registration and a chosen amount of samples of the user's biometric data



Modern systems are also able to do **remote user authentication** over a network, the Internet or more complex communication links. While being convenient, this types of authentication include **additional security threats** such as eavesdropping, password capturing and repli attacks. To avoid this threats, they generally rely on some form of a challenge-response protocol.



1.5 Access Control

Definition 8: Access Control

We define **access control** as the process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities

One of the main access control models is the **Discretional Access Control (DAC)**, which controls access based on the identity of the requestor and on access rules stating what requestors are allowed and not allowed to do. This is achieved through the use of a scheme in which an entity may be granted access rights that permit the entity, by its own volition, to enable another entity to access some resource.

Common ways to implement the DAC model include:

- **Access Control Matrix**: one dimension identifies subjects asking data access to the resources (the users), while the other dimension identifies the objects that may be accesses. Each entry of the matrix indicates the access rights of the associated subject to the associated object. An empty entry defaults to no access right granted

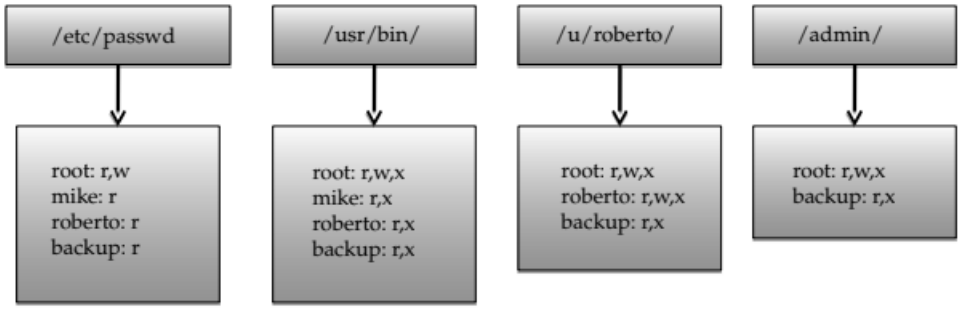
	/etc/passwd	/usr/bin/	/u/roberto/	/admin/
root	read, write	read, write, exec	read, write, exec	read, write, exec
mike	read	read, exec		
roberto	read	read, exec	read, write, exec	
backup	read	read, exec	read, exec	read, exec
...	...	...	...	...

- **Extended Access Control Matrix**: considers the ability of a subject to create another subject and to have "owner" access rights to that subject. Can be used to define a *hierarchy of subjects*

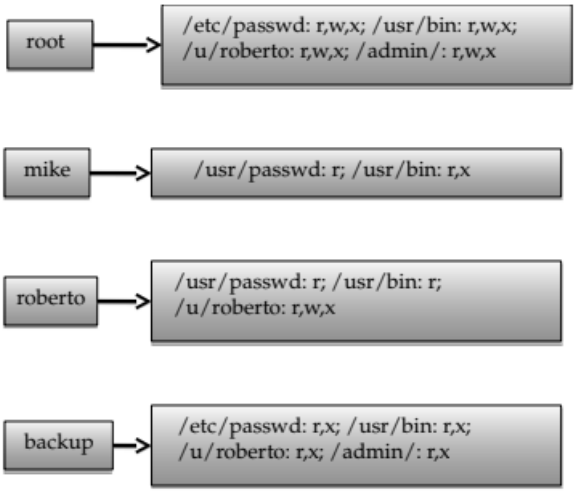
		OBJECTS								
		subjects			files		processes		disk drives	
		S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
SUBJECTS	S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	S_2		control		write *	execute			owner	seek *
	S_3			control		write	stop			

* - copy flag set

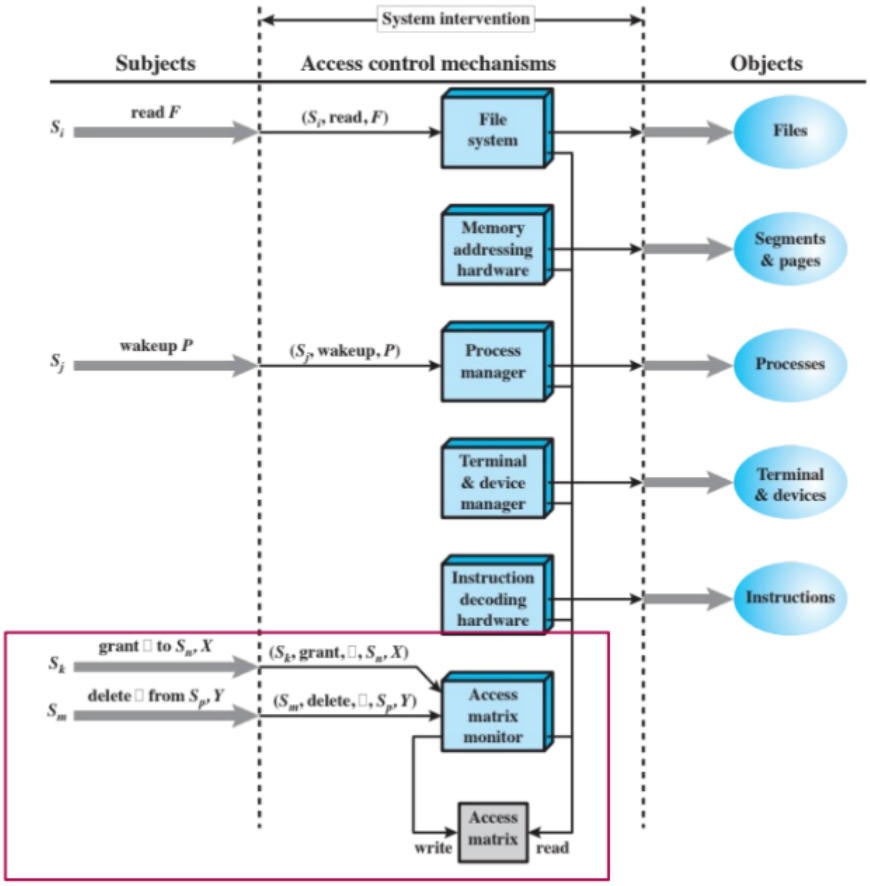
- **Access Control List**: each object has an associated list which enumerates all the subjects that have access rights for that object, specifying the granted rights



- **Capability List**: each subject has an associated list which enumerates all the objects and the access rights granted for each one of them to that subject (same as ACL but objects and subjects are swapped)



- **UNIX File Access Control**: a minimal ACL version, where each objects is identified by the owner (User ID), the primary group (Group ID) and 12 protection bits (Read, Write and Execute bits for object owner, group members and all other users)
- **Access Control Function**: every access by a subject to an object is mediated by the controller for that object. The controller's decision is based on the current contents of the matrix. Certain subjects have to authority to make specific changes to the access matrix



Another way to manage access control is through the **Mandatory Access Control (MAC)** model, where each subject and each object gets assigned a security class, forming a strict hierarchy and being referred to as **security levels**. A subject is said to have a **security clearance** of a given level, while an object is said to have a **security classification** of a given level.

Through **Multilevel Security (MLS)**, the MAC model defines four access modes:

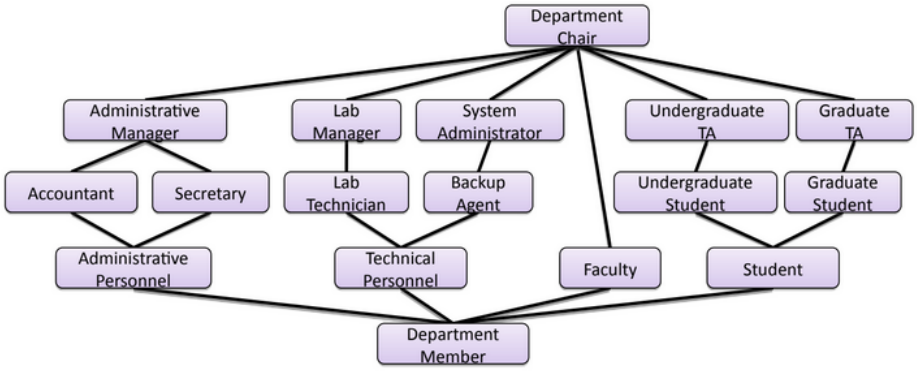
- **read**: the subject is granted read access to the object
- **append**: the subject is granted write access to the object
- **write**: the subject is granted read and write access to the object
- **execute**: the subject is granted the ability to execute the object

Confidentiality is achieved if a subject at high level may not convey information to a subject at lower level, unless that flow accurately reflects the will of an authorized user as revealed by an authorized declassification:

- **No read up**: a subject can only read an object of less or equal security level
- **No write down**: a subject can only write into an object of greater or equal security level

1.5.1 Role-based Access Control

A more advanced model of access control is **Role-based Access Control (RBAC)**, where access rights are defined on roles instead of directly on subjects, allowing to descrive organizational access control *policies* based on job functions.



An user's permissions are determined by its roles rather by identity or clearance, increasing flexibility and scalability in policy administration. Each role is assigned to users through the use of an **User Assignment table**, while each access right gets assigned to roles through the use of a **Permission Assignment table**.

Example:

- Consider the following user and permission assignments:

User	Role
Alice	Radiologist
Alice	GP
Bob	GP
Charlie	Radiologist
David	Nurse

Role	Permission
Nurse	(read, prescription)
GP	(read, prescription)
GP	(write, prescription)
GP	(read, history)
Radiologist	(read, history)
Radiologist	(insert, image scane)

- The corresponding access matrix is defined as:

	Prescription	History	Image scan
Alice	read, write	read	insert
Bob	read, write	read	insert
Charlie		read	insert
David	read		

Through the years, the standard RBAC model, namely the **RBAC0** model, has evolved into four sub-models. The first sub-model is **RBAC1**, where roles are structured in an hierarchy, making lower level roles inherit access rights of their related superior level, reflecting an organization's role structure. Formally, we say that $x \leq y$ if and only if x is a specialization of y. If $x \leq y$, then the role x inherits permissions of role y. The $\leq$ relationship forms a *partial order* on the defined roles.

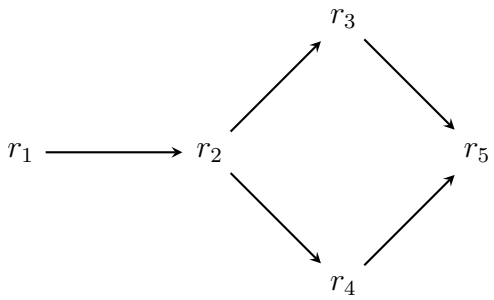
Example:

- Consider the following user and permission assignments:

User	Role
u_1	r_2
u_2	r_3
u_3	r_4
u_4	r_5

Role	Permission
r_1	p_1
r_2	p_2
r_3	p_3
r_4	p_4
r_5	p_5

- Consider now the following hierarchy of roles:



- The corresponding access matrix is defines as:

	$\mathbf{p_1}$	$\mathbf{p_2}$	$\mathbf{p_3}$	$\mathbf{p_4}$	$\mathbf{p_5}$
$\mathbf{u_1}$	×	×			
$\mathbf{u_2}$	×	×	×		
$\mathbf{u_3}$	×	×		×	
$\mathbf{u_4}$	×	×	×	×	×

The second sub-model is **RBAC2**, where role hierarchy is replaced with the definition of **constraints**, providing means of adapting RBAC to the specifics of administrative and security policies of an organization. Constraints are defined through **relationships** among roles or a **condition** related to roles:

- **Mutually exclusive roles**: an user or permission can only be assigned to one role of the defined mutually exclusive set
- **Cardinality**: setting a maximum number of assignable roles
- **Prerequisite roles**: dictates that an user can only be assigned to a particular role only if it is already assigned to some other specified role

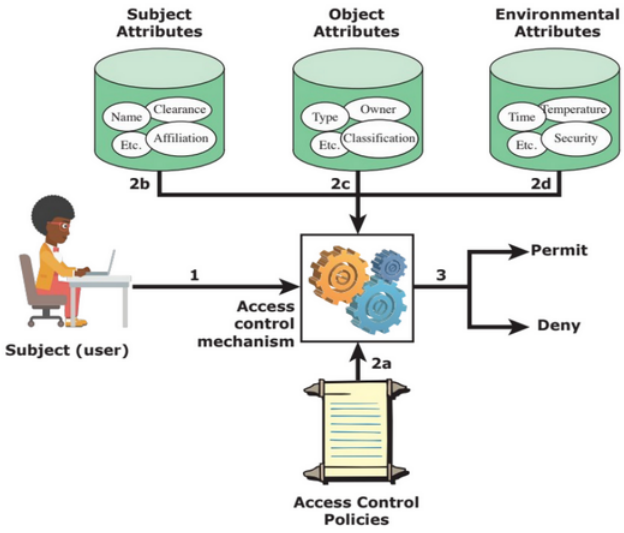
The last sub-model is **RBAC3**, a combination of RBAC1 and RBAC2 (role hierarchy and constraints).

1.5.2 Attribute-based Access Control

Another type of advanced access control model is **Attribute-based Access Control (ABAC)**, which uses attributes to define authorizations that express conditions on properties of both the resource and the subject. The main obstacle to the adoption of this model in real systems has been a concern about the performance impact of evaluating predicates on both resource and user properties for each access.

There are three types of usable attributes:

- **Subject attributes**: a subject is an active entity that causes information to flow among objects or changes the system state. Attributes define the identity and characteristics of the subject
- **Object attributes**: an object is a passive information system-related entity containing or receiving information. Objects have attributes that can be leverages to make access control decisions
- **Environmental attributes**: describe the operational technical and even situational environment or context in which the information access occurs. These attributes have so far been largely ignored in most access control policies



Definition 9: Policy

A **policy** is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions

Example:

- Consider the following policy:
  - Movies rated R can only be accessed by users of age 17+

  - Movies rated PG13 can only be accessed by users of age 13+
  - Movies rated G can only be accessed by everyone
- The following function for the role $R1$ determines if an user u can access the movie m with the given environment values e:

$$\begin{aligned} \text{R1: can\_access}(u, m, e) \leftarrow\ & (\texttt{Age}(u) \geq 17 \wedge \texttt{Rating}(m) \in \{\text{R}, \text{PG13}, \text{G}\}) \vee \\ & (\texttt{Age}(u) \geq 13 \wedge \texttt{Rating}(m) \in \{\text{PG13}, \text{G}\}) \vee \\ & (\texttt{Age}(u) < 13 \wedge \texttt{Rating}(m) \in \{\text{G}\}) \end{aligned}$$

In the RBAC model, as the number of attributes increases to accomodate finer-grained policies, the number of roles and permissions grows exponentially. The ABAC model, instead, deals with additional attributes in an efficient way.

Example:

- Suppose that:
  - Movies are classified as either New Release or Old Release, based on release date compared to the current date
  - Users are classified as Premium User and Regular User, based the fee they pay
  - The policy states that only premium users can view new movies
- In the RBAC model, we have to double the number of roles and the number of separate permissions in order to distinguish each user by age and fee
- In the ABAC model, we can simply define the following functions for the roles $R1, R2$ and $R3$:

$$\begin{aligned} \text{R1:can\_access}(u, m, e) \leftarrow\ & (\texttt{Age}(u) \geq 17 \wedge \texttt{Rating}(m) \in \{\text{R}, \text{PG13}, \text{G}\}) \vee \\ & (\texttt{Age}(u) \geq 13 \wedge \texttt{Rating}(m) \in \{\text{PG13}, \text{G}\}) \vee \\ & (\texttt{Age}(u) < 13 \wedge \texttt{Rating}(m) \in \{\text{G}\}) \end{aligned}$$

$$\begin{aligned} \text{R2:can\_access}(u, m, e) \leftarrow\ & (\texttt{MembershipType}(u) = \text{Premium}) \vee \\ & (\texttt{MembershipType}(u) = \text{Regular} \wedge \\ & \texttt{MovieType}(m) = \text{OldRelease}) \end{aligned}$$

$$\text{R3:can\_access}(u, m, e) \leftarrow \text{R1:can\_access}(u, m, e) \wedge \text{R2:can\_access}(u, m, e)$$

2

Types of Attack