International Journal of Security and Its Applications
Vol.8, No.3 (2014), pp. 407-424
http://dx.doi.org/10.14257/ijsia.2014.8.3.39

SWOT Analysis of Access Control Models

Ennahbaoui Mohammed[1] and El Hajji Said[2]

[1,2] *The laboratory of Mathematics, Computer science and Applications (LabMIA)*
Faculty of Science, University of Mohammed V-Agdal, Rabat, Morocco
[1] *ennahbaoui.mohamed@gmail.com and* [2] *elhajji@fsr.ac.ma*

Abstract

One of the most important components in information systems security is the Access Control policy. In order to ensure the best Access Control policy, it is mandatory to proceed to a modeling phase that respects a set of indications and criteria of a predefined model. There exists several Access Control models, each with a specific contribution. This paper exposes the results found through a SWOT analysis on the well-known models, and presents the advantages and drawbacks of each model. Then, a comparative table between these models is elaborated, in order to get an overview on the types of problems encountered in Access Control and discover the common vulnerabilities between its models. The discovering of the covert channels is among the main results of this study.

Keywords: *Security, access control, policy, SWOT analysis, model, information system*

1. Introduction

The security of an information system is based on the establishment of a security policy. This later is the set of rules and practices that ensure the way how sensitive information and other resources are managed, protected and distributed within a specific system. According to the definitions cited in [1] and [2], it appears that the main axis of a security policy is its Access Control policy. Hence, to ensure the establishment of a good security policy within an information system, the phases of conception and modeling have to be taken into consideration.

Indeed, a deep study of the different Access Control models is very requested. In this context, this paper presents a SWOT analysis of the most famous Access Control models, in order to extract the strengths and weaknesses of each one. In this way, we can compare and qualify them according to the security flaws of each, . In addition, this study also allows to discover a new type of vulnerabilities that affect all Access Control models.

This paper is divided into four parties. The three first concerns introducing the historical Access Control models (MAC, DAC and RBAC) with detection of advantages and drawbacks of each one. Then the last part, we have implanted a comparative table that summarizes the results given through comparing the previously cited models.

In this study we have discovered the flaw of covert channels. They are channels that exploit other means of communication not provided in the security policy. In the future, we will treat in details this vulnerability in order to propose the mechanisms and the solutions to by-pass this flaws.

2. The DAC Model (Discretionary Access Control)

2.1. Introduction

The access control policies are defined as high level directives (rules) [3] that specify who (subject) has permission to practice what (action) on which (object)

ISSN: 1738-9976 IJSIA

data. From this definition we identify three fundamental concepts of access control policy that are:

- **Subject**: active entity that accesses system data. The subject can be a user, an application, an IP address...
- **Object**: passive entity that represents the data to be protected. The object can be, for example, a file, a relational table, a class...
- **Action**: represents the action that subject performs on the object. The action can be to read, to write, to execute...

The Discretionary Access Control model (DAC) [4] allows a subject to assign permissions to other subjects. This access control is flexible, but it can cause errors. The agreement or revocation of privileges is regulated by an administrative policy.

The management access to the files of the operating system UNIX is a classic example of access control mechanism based on a discretionary policy.

We will present in the following the two well known discretionary models, that are the Lampson model and HRU Model (Harrison Ruzzo Ullman model).

2.2. The Lampson Model

The concept of access rights specified by a matrix of access control was introduced in 1971 by Lampson [2]. This model is represented by a triple (S, O, M), where S denotes the subjects, objects O and M = (M_{so}) the access control matrix that associates to each couple (subject s, object o) a set of access rights that are usually: read, write, run.

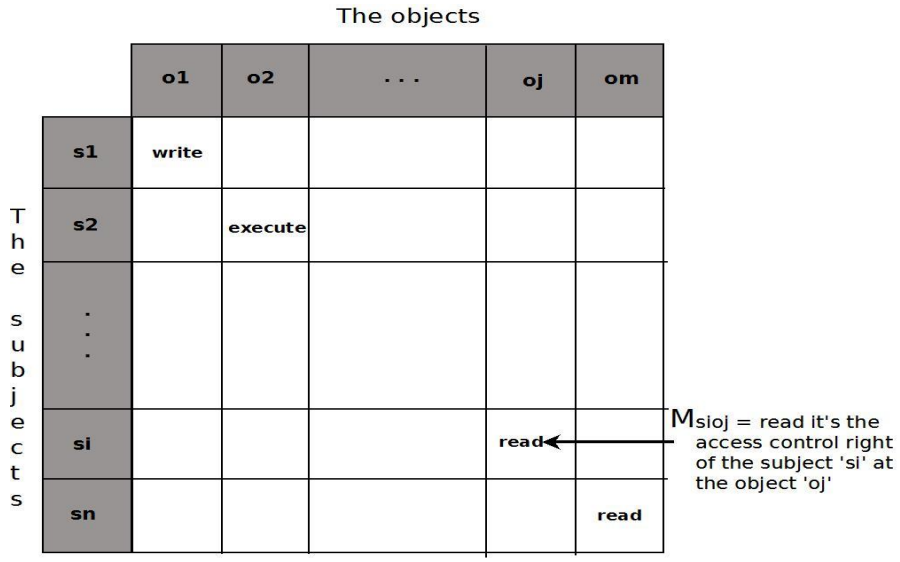


Figure 1. Matrix of Access Control

The matrix shown in Figure 1 shows that the right of access is associated with the subject s_i and the object o_j. While the matrix is not fixed yet, it can be updated by the creation of new objects or subjects, by the destruction of the latter as well as the addition or removal of access rights.

Although the matrix provides a good conceptual representation of permissions, it is inappropriate for implementation. Indeed, the direct storage of this matrix as a two-dimensional array can consume a lot of memory space. For this reason, in practice there are two main categories to implement the access control matrix:

- ACL (Access Control List): the matrix (actions) is stored in columns. Each object is associated with a list indicating for each user the actions it can perform on this subject.
- Capabilities List (CL): the matrix (actions) is stored in rows. Each user has a list, called a capability list, which indicates for each object, the set of the actions that the user is allowed to perform on that object.

Here are two examples of lists that can represent users rights on files 1, 2 and 3 for Alice and Bob.

Table 1. The Examples of ACL and CL

Type Access Control List (ACL)	Type Capabilities List (CL)
Grant read: ON: File A, File B, File C TO: Alice, Bob **Grant write:** ON: File A TO: Alice ON: File B TO: Bob	**Subject Alice:** IS Granted read ON: File A, File B, File C IS Granted write ON: File A **Subject Bob:** IS Granted read ON: File A, File B, File C IS Granted write ON: File B

There is an ambiguity in the adjective "discretionary", which may be understood :

- Either as the fact that rights are organized in a matrix of access control, but without specifying whether it is an authority that defines the rights, or if it is the users who can do.
- Either as is the fact that users themselves can define access rights on the resources they own (HRU model).

2.3. The HRU Model

The Harrison Ruzzo Ullman model (HRU), formalized in 1976 [5] represents an improvement of Lampson model. This model uses a classical access matrix like the Lampson model. The difference lies in that HRU specifies the commands (a set of primitive operations) to assign access rights (read, write, own, *etc.*), as well as create and delete subjects and objects.

In this model, if the right **"own"** is associated with a pair **(s, o)**, the subject s will be considered as the owner of the object o and it may assign its rights of access on the object o to other subjects. In other words, this action allows the subject to define the permissions on the entire column.

The possible primitive operations are:

- Enter: for adding rights.
- Delete: Delete rights.
- Create subject: to create new subjects.
- Create object: to create new objects.
- Destroy subject: the destruction of subjects.
- Destroy object: the destruction of objects.

The commands in the Harrison-Ruzzo-model Ullman are built from primitive operations above and take as argument subjects and objects. We can add a right R in an access matrix M_{so} if there is a command c that adds the right R in a cell of M_{so}:

$$\boldsymbol{c}: \boldsymbol{M_{so}} \longrightarrow \boldsymbol{M'_{so}}, \quad \exists s, o \; such \; that \; r \notin M_{so} \; and \; r \in M'_{so}$$

The HRU command has an optional conditional part as well a body, it has the following format:

$$
\begin{array}{l}
\textbf{\textit{command}}\ c(x_1, \ldots, x_k) \\
\quad \textit{if}\ a_1\ \textit{in}\ M_{s_1 o_1} \\
\quad\quad a_2\ \textit{in}\ M_{s_2 o_2} \\
\quad\quad \ldots \\
\quad\quad a_n\ \textit{in}\ M_{s_m o_m} \\
\quad \textit{then}\ op_1 \\
\quad\quad\quad op_2 \\
\quad\quad\quad \ldots \\
\quad\quad\quad op_n \\
\textbf{\textit{end}}
\end{array}
$$

With $n > 0, m \geq 0, a_1, \ldots, a_n$ are authorizations, $op_1, \ldots, op_n$ are primitive operations. HRU command may not have condition (i.e. $m = 0$).
For example:

1. The command for creating files **create_file** is composed of two primitive operations (without any condition):
   a) Create an object (file f) $\Longrightarrow$ Create f
   b) Declare the subject s as owner of f $\Longrightarrow$ Enter own into M_{so}
   So the command is:

$$
\begin{array}{l}
\textbf{create\_file}(\boldsymbol{s}, \boldsymbol{f}) \\
\quad \textit{Create}\ f \\
\quad \textit{Enter own into}\ M_{so} \\
\textbf{\textit{end}}
\end{array}
$$

2. The command **read_grant** in which the owner s of a file f grants permission to read a subject s' is composed of two primitive operations:
   c) On a condition: s must be the owner of the file f $\Longrightarrow$ if own in M_{so}
   d) On a operation: Enter the read right in the box is $M_{s'o}$ $\Longrightarrow$Enter read into $M_{s'o}$
   So the command is:

$$
\begin{array}{l}
\textbf{read\_grant}(\mathbf{s}, \mathbf{s}', \mathbf{f}) \\
\quad \textit{if own in}\ M_{so} \\
\quad \textit{then Enter read into}\ M_{s'o} \\
\textbf{\textit{end}}
\end{array}
$$

To study the security problem of a protection system, HRU focuses on the transfer of right, that occurs when a command inserts a specific right "r" in the access control matrix "M", in a cell where it was previously absent. Therefore, given an initial configuration of the security policy, a system is considered safe for a right "r" if no one of commands causes the transfer of the right "r".
The authors of the HRU model showed that in the case of a mono-operational protection system, *i.e.*, in which all commands contain only one primitive operation, the issue of whether the system is safe is decidable, but the verification algorithm is NP-complete. But the only command to create a file of an operating system like UNIX, already consists of two actions: Creating a new object, and positioning rights thereon.
So, the authors are interested to the problem in the general case, and prove that the security problem of a protection system is undecidable in the general case.
In addition, the authors mention that the problem size is reduced to a polynomial size when the actions of creating subject or object is removed from the protection system.

2.4. Problems Raised from DAC Model

The discretionary access control model limits access to objects, basing only on the user's identity. For this, the DAC model is also called IBAC (Identity Based Access Control). This basic principle makes the DAC model of the access control vulnerable to Trojan horses [3].

To explain the running of a Trojan horse in a security policy based on the DAC model, we will give the example following: Let's suppose an organization, the director Bob creates a file "Prod" containing very sensitive information about new products. This sensitive information, according to the policy of the organization, should be accessible only by Bob.

Now suppose that a malicious user Robert, an assistant to Bob wants to recover the sensitive information in order to sell them to a concurrent organization.

For this, Robert creates a file "Nprod and gives permission to write in this file to Bob. Robert then introduces two hidden operations in an application generally used by Bob. These operations are read from the file "Prod" and write in file "Nprod". Once Bob runs the application, read and write operations will be permitted. Since the malicious user Robert is the owner of the file "Nprod", it can access that file and retrieve the desired information.

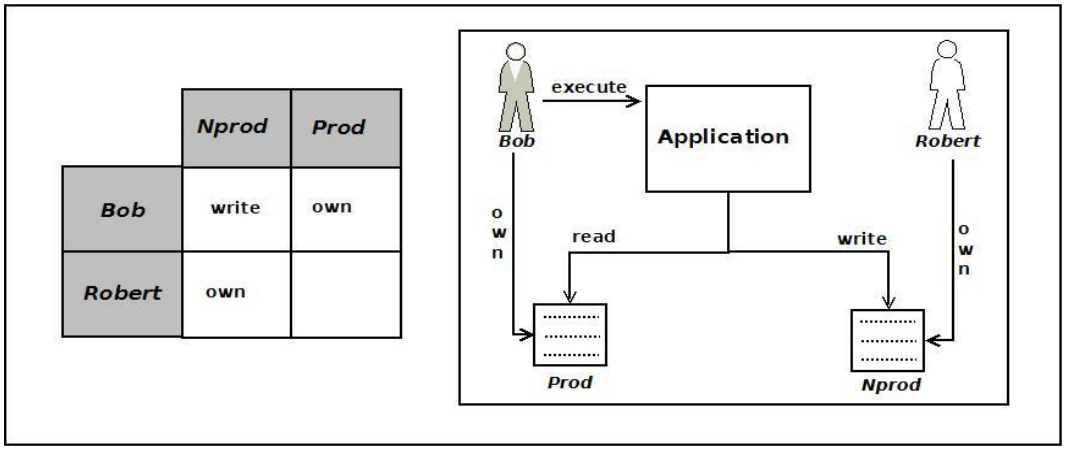


Figure 2. The Torjan Horse in DAC Model

We note that despite the fact that we trust users so they follow the policies of the organization, we cannot trust the processes running on their behalf, hence the need to distinguish between users and processes that are running for their accounts (subjects).

In the next section, we will show how the MAC models (specifically multilevel model) distinguish between subjects and objects to solve Trojan horses and the information leakage they cause.

3. The MAC model (Mandatory Access Control)

3.1. Introduction

The Mandatory Access Control model (MAC) [6] allows to create obligatory security policies that set essential rules to force compliance of access control requirements. Thus, unlike the DAC model, users can not define the rights of access control, because all objects are the exclusive property of the organization, which implies that in this model the access control policy is managed in a centralized manner.

The Mandatory Access Control [7] is based on the concept of security levels associated with each subject and object, from which are derived the permissions and actions.

A mandatory policy of security, is only a multi-level policy [8], this latter has the notion of access class. A partial order relation is defined on the set of access classes, it is the dominance relation symbolized by $\geq$.

Each access class has two component:

- **Security Level:** is an element of a totally ordered set, e.g. top secret (TS), secret (S), confidential (C) and unclassified (N) where TS $\geq$ S $\geq$ C $\geq$ N. For objects, security level is called the classification level and for subjects it is called clearance level.
- **A set of categories:** describes the various fields of system in study. For example, in military systems, the categories are nuclear and army, in commercial systems the categories are rather financial, administrative...

Let L be the set of security levels, equipped with the partial order relation $\geq$, and C is the set of categories, equipped with the partial order $\supseteq$. Let l_1, l_2, two levels and c_1, c_2 two categories such as: $l_1 \in L,\ l_2 \in L,\ c_1 \in C,\ c_2 \in C$.

Given two access classes: ac_1 and ac_2, the dominance relation $\geq$ is defined as:

$$\forall\, \boldsymbol{ac_1 = (l_1, c_1), ac_2 = (l_2, c_2)}: \boldsymbol{ac_1 \geq ac_2 \Leftrightarrow\ l_1 \geq l_1 \wedge c_1 \supseteq c_2}$$

The structure of all access classes forms a trellis that is why multi-level policies are also called by LBAC (Lattice based access control). The following figure shows an example of a trellis of security for a military system.

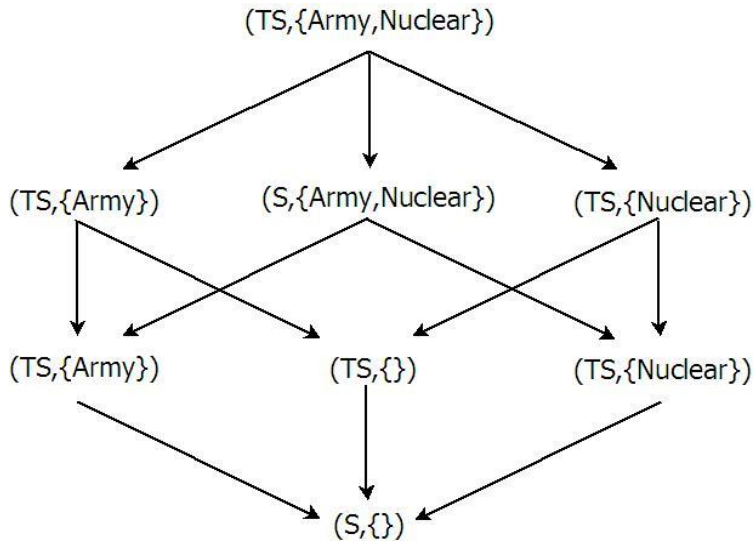


Figure 3. The Example of a Security Trellis

To summarize, the multilevel model with all its variations is based on the trellis concept, it also uses an access matrix identical to the HRU model in order to present authorizations on which are added security levels.

We will then speak of the two most famous models of Mandatory Access Control, that are the Bell-LaPadula model (BLP) which has the purpose to ensure confidentiality, and the Biba model which is interested to integrity.

3.2. The Bell-LaPadula Model (BLP)

The Bell-LaPadula model (BLP), developed in 1975 [4, 9] seeks to preserve the confidentiality of the data, that is to say that these latter are only accessible by authorized users.

In this model, access rights depend classifications assigned to objects and authorizations granted to subjects, basing on two laws:

1. **The simple property (No read up):** simply do not read up. In effect, this law prohibits a subject to have a read access to an object that has a higher classification than the habilitation of the same subject.
2. **Star property (No write down):** simply do not write down (write is used to mean the only writing or addition). In effect, this law prohibits a subject to have a write access to an object that has a classification lower than the habilitation of the same subject.

To formalize the two laws of Bel-LaPadula, we need a function $f: S \cup O \longrightarrow L$ applied to subjects and objects, and returns the level of classification and habilitation of these latter.

The formal description of the two principles "no read up" and "no write down" is reflected by the following axioms:

1. **Simple property:** A subject s can read an object o only if its habilitation dominates the classification of the object:

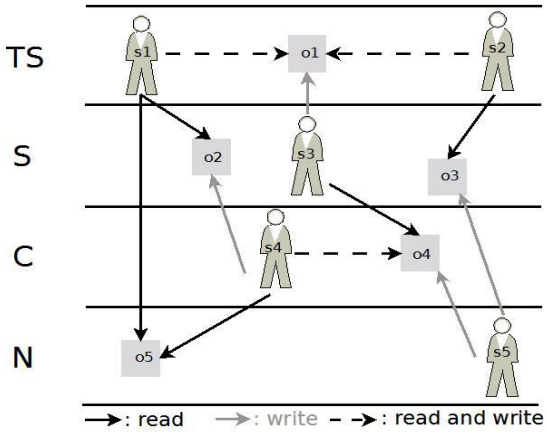
$$read \in M_{so} \Longrightarrow f(s) \geq f(o)$$

2. **Star Property:** A subject s can write in an object o only if the classification of the object dominates the habilitation of the subject:

$$write \in M_{so} \Longrightarrow f(o) \geq f(s)$$

As a consequence of these two laws, there are a property that describes if a subject s can write in and read from an object o. So the habilitation of s is equal to the classification of o. The next axiom formulates this property:

$$write \in M_{so} \wedge read \in M_{so} \Longrightarrow f(o) = f(s)$$



	o1	o2	o3	o4	o5
s1	read write	read			read
s2	read write		read		
s3	write			read	
s4		write		read write	read
s5			write	write	

Figure 4. Two Laws of BLP Model

3.3. The Biba Model

The Biba model developed in 1977 [4, 10] focuses on data integrity, what is missing in the BLP mode. Ensuring data integrity means that they can only be changed by authorized users. As in BLP, the Biba model is based on two laws:

1. **The Simple Property (No Read Down):** simply do not read down. In effect, this law prohibits a subject to have a read access to an object that has a classification lower than the habilitation of the same subject.
2. **Star Property (No Write Up):** simply do not write up (write is used to mean the only writing or addition). In effect, this law prohibits a subject to have a write access to an object that has a higher classification than the habilitation of the same subject.

The formal description of the two principles "No read down" and "No write up" is translated in the following axioms:

1. **Simple Property:** A subject s can read object o if the habilitation is dominated by the classification of the object:

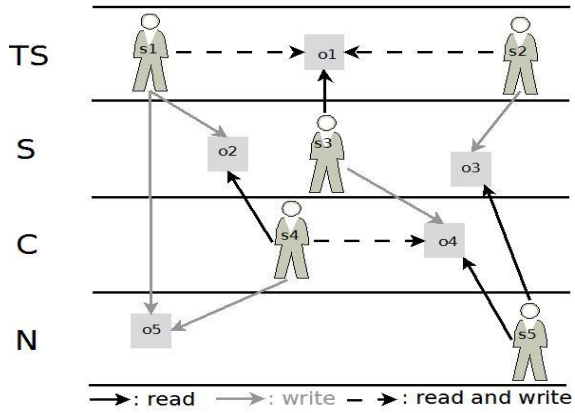
$$read \in M_{so} \Longrightarrow f(o) \geq f(s)$$

2. **Star Property:** A subject s can write in an object o if the classification of the object is dominated by the habilitation subject:

$$write \in M_{so} \Longrightarrow f(s) \geq f(o)$$

As a consequence of these two laws, there are a property that describes if a subject s can write and read an object o. So the habilitation of s is equal to the classification of o. The next axiom formulates this property:

$$write \in M_{so} \wedge read \in M_{so} \Longrightarrow f(o) = f(s)$$



	o1	o2	o3	o4	o5
s1	read write	write			write
s2	read write		write		
s3	read			write	
s4		read		read write	write
s5			read	read	

Figure 5. Two Laws of Biba Model

The satisfaction of both Biba laws, prevents the information flow from low integrity level (less reliable) to a higher integrity level. These two laws are too restrictive, which led the Biba model to propose two other principles that are more flexible to preserve the integrity:

- **Low water mark for the subjects:** A subject s can read an object o at any integrity level. But after this operation, the integrity level of the subject decreases towards its the lower level and that of the object:
  $f'(s) = inf(f(s), f(o))$ (where f' is the new integrity level of the subject).
- **Low water mark for objects:** A subject s can write in an object o at any integrity level, but after this operation, the level of integrity of the modified object decreases towards its lower level and that of the subject:
  $f'(o) = inf(f(s), f(o))$ (where f' is the new integrity level of the object).

According to these two principles, an extension of the Biba model is proposed with two different ways:

1. The first way using the laws "No write up" and "Low water mark" for the subjects.
2. The second way using the laws "No read down" and "Low water mark" for objects.

This approach of "Low water mark" is also applied at the BLP model in order to have an extension of this latter, on condition to consider that this model is the dual case of Biba model.

To ensure the confidentiality and integrity in a single model, there are suggestions to assemble two contradictory models BLP and Biba in a unique design that will give to each subject and object two levels, the first one is the confidentiality level and the second is integrity level.

3.4. Problems Raised from MAC Model

The MAC model has a fixed security policy managed by an authority, and cannot be changed by users, which excludes problems related to information leakage (using Trojan horses) that exist in the DAC model. This is mainly due to the fact of not allowing users to interfere with the access control policy.

The highlight of the MAC model is that is rigid and this same point makes the system unreliable because: The use of this model with all its variations is complicated, by the fact that it provides a too narrow policy and therefore difficult to deploy in practice. However, for large organizations, it is difficult to classify

objects and subjects in a predefined number of security levels. The users do not have sufficient privileges to manage their own security needs, especially those regarding privacy. And therefore the flow of information between users is reduced.

We can add to the problem of rigidity of MAC model, another problem met in the majority of access control models, which is the problem of covert channels.

A covert channel is a communication path that it was not originally intended or is not allowed to transfer information, which violates security policy of implementation. Note that a covert channel requires at least two contributors: one who gives the information (writing data) and another who receives it (reading information).

Generally, there are two types of covert channels [11]: memory channels and temporal channels.

Channels of memory or storage (storage channels) allow communication of information by altering the objects stored and shared between subjects of different habilitations.

If the system obeys a mandatory policy, it prohibits the use of direct means such as shared memory, sending messages, sharing files, the reuse buffer or temporary files, *etc*...

A covert channel of storage is the use (indirect) of a mechanism that allows a process to write an information which can be read by another process.

The timing channels allow the communication of information by altering the temporal sequence of events. In fact, when a hardware or software resource is shared between different habilitation users and there is a common clock which is accurate enough, it is possible to create a temporal channel by modulating the use of the resource.

A classic example are the shared disks: the privileged user can move the disk arm on the outer cylinder (to transmit a "1") or the inner cylinder (to transmit a "0"), the non-privileged user can then move the arm on the same disc to the inner cylinder: as the response time, it will know if it is the case of a "1" or "0".

Models of mandatory security policies MAC have historically been implemented in many operating systems such as some versions of Solaris, HPUX or other UNIX systems. And today, there is a new generation of smart cards called "multi-application cards" that are based on these models to implement a security policy which is rigid and closed.

4. The RBAC model (Role-Based Access Control)

4.1. Introduction

The access control model based on RBAC role is considered as an alternative approach to mandatory access control (MAC) and discretionary access control (DAC). RBAC was proposed for the first time in 1992 by David Ferrailo and Richard Kuhn [12], it aims to facilitate the administration of the access control policy.

The core of a RBAC model is the **role**, that represents in an abstract way a function or a trade within an organization that combines the authority and responsibility assigned to a person who plays this role (*e.g.*, Professor, Director, Engineer, Technician ...).

Each role is associated with permissions (or privileges) that constitute a set of rights corresponding to the tasks that can be performed by each role. Finally and contrary to the models that preceded RBAC, permissions are associated in direct way to the subjects, but through roles.

The two relations of the figure below "Detain (Role, Permission" and "Play (Subject, Role)" define precisely the permissions granted to each subject.

A role can have many permissions, and permission may be associated with multiple roles. Similarly, a subject may be a member of multiple roles and vice versa, a role can be performed by several subjects.

Thus, if Dr. Dupont is both surgeon and hospital director, as a surgeon ,he has the access right to medical records, while as a director, he will have access to administrative information.

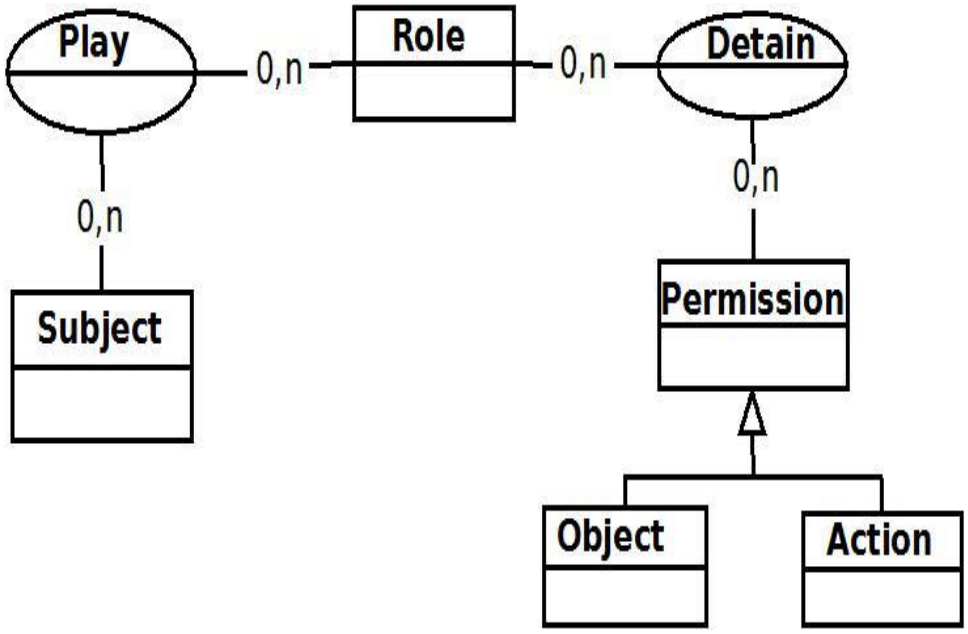


Figure 6. The Two Relations of RBAC

One way to define roles within an organization is to group in each role the tasks that perform the same operations, then it is the matter to identify the objects used by these tasks and define the rights of access to these objects, *i.e.*, the pairs (set of rights, objects) and finally combine these rights to roles. The assignment of subjects to roles is a task to be done separately, and probably by other administrators.

As several roles models have been proposed, we will deal with a RBAC family of four models [13]:

- The $RBAC_0$ model, which presents the basic concepts and relations is the core of the model.
- The $RBAC_1$ model, incorporates the $RBAC_0$ model and introduces the notion of hierarchy
- The $RBAC_2$ model, incorporates the $RBAC_0$ model and introduces the notion of constraint.
- The $RBAC_3$ model, incorporates $RBAC_1$ and $RBAC_2$ models and takes into account the interactions between constraints and hierarchy.

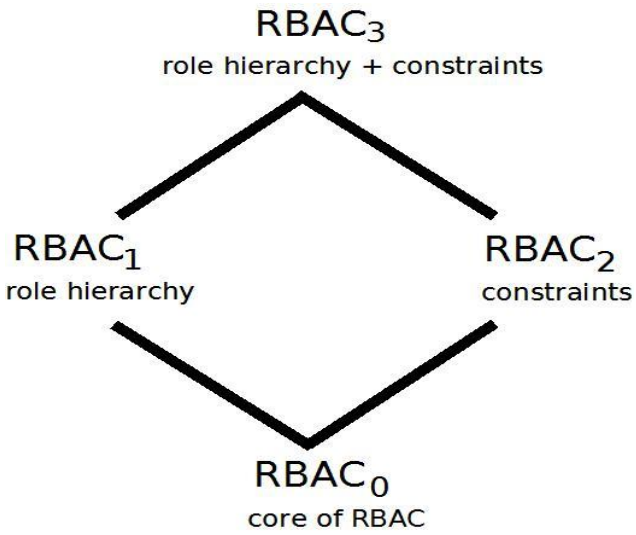


Figure 7. The Four Models to the RBAC Family

4.2. The $RBAC_0$ Model (Core RBAC)

It covers the basic criteria included in all RBAC systems. It recognizes seven administrative elements:

- **User:** refers to the person who interferes with the computer system. In many conceptions, a user can have multiple sessions that are simultaneously active.
- **Subject:** refers to the process playing on behalf of a user. Two relations are defined: the first one connects a user to a subject and another one connects a subject to the set of active roles.
- **Session:** it is the entity that represents an active user in the system. A given session is assigned to one and a single user. In a one session, a user chooses to assume some or all of the roles assigned to him.
- **Role:** the functions or responsibilities of employees in an organization.
- **Operation:** it is an active process invoked by the subject (read, write, *etc*).
- **Object:** any resource available in a computer system (files, devices such as printers, *etc*).
- **Permission** (privilege): it is an authorization to perform an operation on an object.

This model has two laws to follow:

- **Authorization of a role:** a subject can never have an active role not allowed to him.
- **Authorization to access an object: a** subject s can perform an operation op on an object o only if there is:
  1. a role r belonging to the set of active roles on subject s.
  2. a permission granted to role r that allows it to perform the operation op on the object o.

4.3. The $RBAC_1$ model (The Hierarchy Role)

The motivation to introduce this aspect in RBAC is that within an organization many roles may have several common permissions. As examples to general permissions there are an internal Web site access, the ability to upload documents, *etc*. These permissions can be granted to all employees or most of them.

The inheritance relation of a role creates an authorization form. If the role A inherits from role B, this means that all permissions of role B are allowed by the role A. In

other words, the permissions of B constitute a subset of the set of permissions of A, and all users playing the role A can also play the role B. Two types of hierarchy roles have been defined:

1. **General:** allows multiple inheritance of permissions. This means that a role can simultaneously have one or more parents (inherits permissions from multiple sources).
   Formally, we define a partial order noted $\geq$ on $RH \subseteq ROLES \times ROLES$ (with RH is the set of inherited roles and ROLES is the set of roles) such that $r_1, r_2 \in RH$:
   $r_1 \geq r_2 \Longrightarrow$authorized-permissions$(r_2) \subseteq$ authorized-permissions$(r_1) \wedge$ authorized-users$(r_1) \subseteq$ authorized-users(r_2) .
   with:
   authorized-users(r)$= \{u \in USERS \mid r_1 \geq r_2 \wedge (u, r_1) \in UA\}$ (USERS: all users).
   authorized-permissions(r)$= \{p \in PRMS \mid r_1 \geq r_2 \wedge (p, r_1) \in PA\}$ (PRMS: set of permissions).
   where UA is the set of pairs (user, role), i.e authorized users by role and PA is the set of pairs (permission, role), i.e the permissions granted to a role.
2. **Limited:** is defined as the general hierarchy but it does not allow multiple inheritance.
   Formally, $r_1, r_2, r_3 \in RH: r_1 \geq r_2 \wedge r_1 \geq r_3 \Longrightarrow r_2 = r_3$

4.4. The $RBAC_2$ model (The Constraints)

The concept of separation of duties (SOD or Separation Of Duty) was added later in the RBAC model. This concept ensures that no person has the ability to control all the steps involved in a high-risk operation, and no user has enough rights to abuse the system alone. This helps prevent from frauds and major errors. There are several varieties of SOD where two broad categories have been described by [14]:

1. **Static Separation of Duties:** no user can have two roles designated as mutually exclusive (conflicting).
2. **Dynamic Separation of Duties:** no user can have during a same session, two roles called mutually exclusive.

We can add to the constraint of separating tasks, another type called the Temporal Constraints. These latter have a purpose to introduce the concept of time in the specification of access control requirements [15]. For example, a user can have the role of cashier that during the opening hours of the bank (*e.g.*, from 9:00 to 2:00 Monday to Friday and 9:00 to 12:00 on Saturdays).

4.5. The $RBAC_3$ Model

The $RBAC_3$ model assembles $\boldsymbol{RBAC_1}$ and $\boldsymbol{RBAC_2}$ models respecting the properties and criteria of each one in order to have a stable security policy, that is complete and easy to administer. The figure below includes the principle operating of $RBAC_3$:

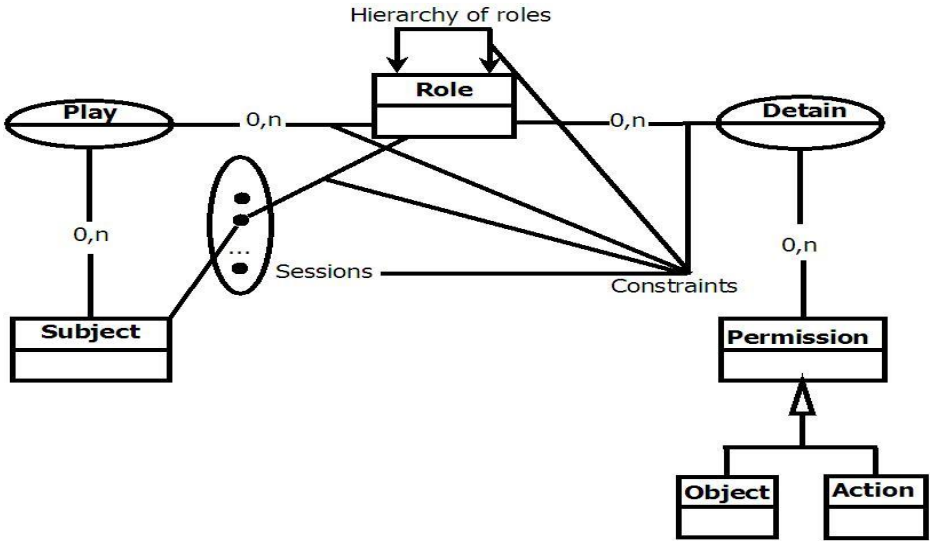


Figure 8. The $RBAC_3$ Model

4.6. Problems Raised from RBAC Model

The analysis of role-based policies allows us to conclude that they are relatively easy to administer and flexible enough to adapt to any organization. Indeed, the definition of roles may reflect the structure of the organization. Roles can be structured hierarchically to simplify the assignment of permissions and reduce the number of required associations as they do not directly link individuals with these permissions. With RBAC model, it is easy to add a user: just assign it the roles it should play in the organization. Similarly, it is relatively easy to evolve tasks due to the creation or modification of an object: just update the privileges of roles involved.

The main drawback of RBAC model is the difficulty to manage and implement rules depending on context, such as " only doctors can read the medical file of a patient" or "students have the right to access only to their personal data". To solve this problem with RBAC, we must create as many roles "doctor treating patient X" as patients.

For the problem of student information, one proposed solution is to create, for example, for each student a private role. It is theoretically a solution, but in practice this is not feasible because there are a large number of students, which leads RBAC to lose its simplicity of administration.

We can add other drawback to RBAC as:

- The permission concept is primitive. There is no generic structure of permissions. These latter depend on the concrete application of the model. We believe on the contrary that it would be better to add to the model a generic permission structure.
- The hierarchy concept of role is somewhat ambiguous. It is generally incorrect to consider that the role hierarchy corresponds to the organizational hierarchy. For example, the director of a software development company has an administrative role higher than the role of a developer. However, the director is not necessarily a developer.
- The distinction between the role concept and that of a group is unclear. The group is a concept that was introduced before the definition of RBAC model.
- As in other models (DAC and MAC), it is not possible to define the obligations and recommendations.
- The application of RBAC model in the definition of system security policy that contains several organizations reveals other limitations of this model.

5. SWOT Analysis of Access Control Models

We classified the historical models of access control according to the advantages and drawbacks of each one.

We tried to take the common criteria between these models such as flexibility, security flaws, modification, the ability to express rules of authorizations, prohibitions, obligations and recommendations. And from these criteria, we drew a comparison table that explains the influence of each criterion on each model. For instance, we can observe that both security issues: the covert channel flaw and the possibility to only express the authorization rules, are existing in the three models: DAC, MAC and RBAC. However, the Trojan horses flaw is only existing in the DAC model. The RBAC is the only one of the three models that is simple in updates. If we talk about the flexibility, we can say that the MAC model is rigid while DAC and RBAC models are flexible. Table 2 discusses in detail the results found during our SWOT analysis between these access control models.

Table 2. The Comparative Table of the Historical Access Control Models

DAC	
Advantages	**Disadvantages**
• Flexible. • Use in environments where the sharing of information is more important than protection	• Updating the security policy • is costly. • Vulnerable to Trojans and to covert channels. • Does not distinguish between users of the subjects. • Does not permit to express prohibitions, recommendations or obligations
MAC	
Advantages	**Disadvantages**
• Rigid. • Distinguishes between users and subjects. • Conceived in the environments where the hierarchy of users is more important than sharing information.	• Updating the security policy is costly. • Vulnerable to covert channels. • Does not allow the flow of information between the different levels (problem related to the rigidity). • Does not permit to express prohibitions, recommendations or obligations.
RBAC	
Advantages	**Disadvantages**
• Updating the security policy is simple which explains the ease of administration policies based on this model. • Encompasses the advantages of the traditional models DAC and MAC. • Applied in complex and distributed areas the fact that this model is based on the concepts of constraints and inheritance.	• Impossible to express the rules depend on the context. • The distinction between the role concept and the group concept is vague. • The absence of generic structure of permissions. • Vulnerable to covert channels. • Does not permit to express prohibitions, recommendations or obligations.

6. Conclusion

In this paper, we have exposed a detailed study on the Access Control models. We have started with discretionary, mandatory models and we finished with RBAC Model. We have explained the principle of each model, then we have described their advantages and drawbacks, which allowed us to detect the vulnerability of DAC model to Trojan horses and the ability of MAC model to resolve this vulnerability. Although, we have noticed that MAC model falls on another type of security vulnerabilities called covert channels, due to the fact that it is rigid. Our study shows that the RBAC model comes to solve the security issues of the both previous models (MAC and DAC), but the major concern with this model is its inability to express the contexts.

After elaborating a comparison between these three historical models, we were able to extract common flaws between them: covert channels and the absence of a way to express prohibitions, obligations and recommendations in the security policy.

Even with the disadvantages of these models, we cannot eliminate them or favor one over the other, because each one has its specifications and its application domains. This is what motivated us to do this study as an attempt to reuse them in new areas such as embedded systems, smart cards, *etc*.

The main result of this study was the discovering of the covert channels flaw, and as perspective we will address in detail this type of vulnerability and proceed to a thorough research, in order to find a way to treat and by-pass this attack that affects the information systems.

References

[1] K. Rihaczek, “The harmonized ITSEC evaluation criteria”, Computers & Security, vol. 10, no. 2, **(1991)**, pp. 101-110.
[2] B. W. Lampson, “Protection”, Operating Systems Review, vol. 8, no. 1, **(1974)**, pp. 18-24.
[3] P. Samarati and S. de Vimercati, “Access Control: Policies, Models, and Mechanisms”, Foundations of Security Analysis and Design, **(2001)**, pp. 137-196.
[4] C. E. Landwehr, “Formal models for computer security”, ACM Computing Surveys, vol. 13, **(1981)**, pp. 247-278.
[5] A. M. Harrison, L. W. Ruzzo and D. J. Ullman, “Protection in Operating Systems”, Communications of ACM, vol. 19 , no. 8, **(1976)**, pp. 461-471.
[6] E. D. Bell and J. L. La Padula, “Secure computer system: Unified exposition and Multics interpretation”, Mitre Corporation, Bedford, MA, **(1976)**.
[7] R. S. Sandhu, “Lattice-Based Access Control Models”, IEEE Computer, vol. 26, no. 11, **(1993)**, pp. 9-19.
[8] D. E. Denning, “A Lattice Model of Secure Information Flow”, Commun. ACM, vol. 19, no. 5, **(1976)**, pp. 236-243.
[9] D. Elliott Bell and L. J. LaPadula, “Secure computer systems: Mathematical foundations”, MITRE Technical Report 2547, vol. I, **(1973)**, pp. 74-244.
[10] J. K. Biba, “Integrity Considerations for Secure Computer Systems”, U.S. Air Force Electronic Systems Division , Bedford, MA, **(1977)**.
[11] A. Abou El Kalam, “Modèles et politiques de sécurité pour les domaines de la santé et des affaires sociales”, Ph.D. dissertation, Institut National Polytechnique de Toulouse (INPT), French, **(2003)**.
[12] F. D. Ferraiolo and R. D. Kuhn, “Role Based Access Control”, Proceedings of 15th National Computer Security Conference”, Baltimore, **(1992)**, pp. 554-563.
[13] R. Sandhu, E. Coyne, H. Feinstein and C. Youman, “Role-based access control models”, IEEE Computer, vol. 29 , no. 2, **(1996)**, pp. 38-47.
[14] R. T. Simon and M. E. Zurko, “Separation of Duty in Role-based Environments”, CSFW, IEEE Computer Society, **(1997)**, pp. 183-194.
[15] G.-J. Ahn and R. S. Sandhu, “Role-based authorization constraints specification”, ACM Trans. Inf. Syst. Secur., vol. 3, no. 4, **(2000)**, pp. 207-226.

Authors

Ennahbaoui Mohammed, PhD, Student in the laboratory of Mathematics, Computer science and Applications (LabMIA), Faculty of Science-Rabat, University of Mohammed V-Agdal, Rabat, Morocco. Email: ennahbaoui.mohamed@gmail.com

El Hajji Said, Full Professor in Faculty of Science-Rabat, University of Mohammed V-Agdal, Rabat, Morocco. Head of the laboratory of Mathematics, Computer science and Applications (LabMIA), Email: elhajji@fsr.ac.ma