

Lecture 10 - ACCESS CONTROL

Marvin TCHOULA NJIA - Endri VANGJEL

November 18, 2009

1 Introduction

To understand access control, we need to define some properties like identification, authentication, authorization and access control.

Identification is a process

Authentication is the process of verifying the validity of something claimed by a system entity. for example eyesrecognizing, fingerprint, digitcode. Eyesrecognizing and fingerprint is also identification's process.

Authorization is a right or permission that is granted to system entity to access a system resources. For example in ubuntu linux, we need system authorization by root password to be able to download synaptic packages. This kind of process use also authentication. Another example to explain more authorization is let a competition where everybody is accepted while the number of participant doesn't exceeded 500 persons. Everybody is authorized to register while we are less than 500 person.

Access Control is a Protection of system resources against unauthorized access; a process by which use of system resources is regulated according to a security policy and is permitted by only authorized entities (users, programs, processes, or other systems) according to that policy.

example : ubuntu linux.

2 ACCESS CONTROL

The goal of access control is to give the rights authorization to users who want to access to resources specifying what's allow or not to do. This is done by using matrices, lattices or other mathematical structures, which specifying which rights subjects have on objects. Access control is also concerned with enforcement mechanisms. There exist 2 main models of access control :

Mandatory Access Control (MAC) and **Discretionary Access Control (DAC)**

2.1 Discretionary Access Control (DAC).

DAC used the UNIX principle which means that the users own resources and control their access. In this model, the owner has total freedom. He can change object's permission and he can even be able to transfer ownership to other users. this allows direct or even transitive delegation of right. The advantages of DAC is that is flexible but there are some disadvantages like no control of information dissemination. there is no mechanism to show who have change information. In other word, authentication is not satisfied. A lot of trust is given to

the ability of the user the mechanism and respect the security policy.

example of DAC is UNIX SYSTEM

ACLs typically limited to 9 bits: rwx for user, group, and others.

-rw-r--r-- 1 plafourc users 4158 2007-11-09 12:37 Intro.tex

drwxr-xr-x 5 plafourc users 4096 2007-11-02 14:07 Tools/.

The file **Intro.tex** can be read by everyone but only the owner has the right to write on it.

2.2 Mandatory Access Control (MAC)

In this model of access control, the decision are formalized and control by comparing security level indicating sensitivity and criticality of objects with formal authorization of subjects. Different from DAC, in this model, the user doesn't have total freedom on the file where he is working. For example subjects are not able to change the level of security or transfer their access right, their abilities are specified before by an entity above him.

it is developed by military organisation and it's less flexible than DAC but more secure. We can defined the level of security of an object by listening it as unclassified, Confidential, Secret or Top secret.

We formalize this model by lattice of compartments and linearly ordered sentive levels. So we can defined class as a couple rank compartment: Class = (rank, compartment). There is dominance relation defined component-wise:

$$(r1, c1) \leq (r2, c2) \iff r1 \leq r2 \wedge c1 \subseteq c2.$$

For example: (secret, iraq) \leq (top secret, Middle east).

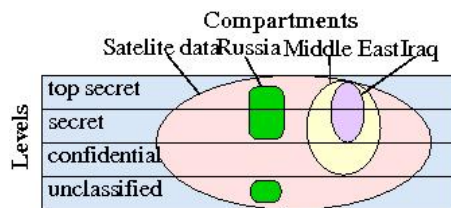


Figure 1: dominance relation

we can say the top secret in Middle East is more important than secret in iraq. Hence, someone who have access in middle East's file have access to iraq's file.

2.2.1 What's and why Lattice

a lattice (L, \leq) consists of a set of L and a partial ordering \leq so that for every 2 elements a,b in L, there exists a least upper bound u in L and a greatest lower bound I in L, i.e:

$$a \leq u, b \leq u \text{ and } (a \leq v \ \& \ b \leq v) \implies (u \leq v) \text{ for all } v \text{ in } L.$$

$$I \leq a, I \leq b \text{ and } (k \leq a \ \& \ k \leq b) \implies (k \leq I) \text{ for all } k \text{ in } L.$$

2.3 ANOTHER MODELS

2.3.1 Bell Lapadula Model (BLP) - 1975

The Bell-La Padula Model (abbreviated BLP and often misspelled Bell-LaPadula) is a state machine model used for enforcing access control in government and military applications. It was developed by David Elliott Bell and Leonard J. La Padula, subsequent to strong guidance from Roger R. Schell to formalize the U.S. Department of Defense (DoD) multilevel security (MLS) policy. The model is a formal state transition model of computer security policy that describes a set of access control rules which use security labels on objects and clearances

for subjects. Security labels range from the most sensitive (e.g. "Top Secret"), down to the least sensitive (e.g., "Unclassified" or "Public").

This model aimed to assure confidentiality of data with that simple rule : **no writing in a lower level, not playing a higher level**. But, on the other side, integrity is not guaranteed; an objet in a level A could be modified by all subject in this level and also by subject who are in level below. To correct this weakness, biba model was invented.

2.3.2 Biba model - 1977

BIBA MODEL was developed by Kenneth J. Biba in 1977. her goal is to correct the weakness of BELL LAPADULA. This correction is given by that simple rule : **no writing at a higher level, not reading a lower level**. Hence, this model assure that subject will not be able to modify an object to the level above on him and her objet will not be modified by an subject to the level below on him. Therefore, subject will only be able to modify data in her level or in level below. This guaranted integrity but not confidentiality of data/objet.

example : an manager can overwrite subordinate's data.

advise IF you want both integrity and confidentiality, Use BLP for classifying some data and biba for others.

3 ACCESS CONTROL MATRICE MODEL

ACCESS CONTROL MATRICE MODEL is a simple framework for describing a protection system by describing the privileges of subjets on objects. Remember that subject can be users, processes, agents, groups, ..., objects can be data, memory banks, other processes, ... and privileges(permissions, rights) can be read, write, modify, ...

A protection state is a triple $(\mathbb{S}, \mathbb{O}, \mathcal{M})$, where \mathbb{S} is a set of subjects, \mathbb{O} set of object and \mathcal{M} is a matrice defining the privileges/rights of a subject $s \in \mathbb{S}$ on an object $o \in \mathbb{O}$ (*figure 2*).

		Objects					
		File 1	File 2	File 3	File 4	Account 1	Account 2
Subjects	Alice	Own R W		W X		Inquiry Credit
	Bob	R	Own R W	W	R	Inquiry Debit	Inquiry Credit
	Charlie	R W	R		Own R X		Inquiry Debit

Figure 2: representation of access matric

\mathcal{M} provide a basis for different possible enorcement mechanism : **access control list** (*fig 3*) and **Capacities list**(*fig 4*).

Access Control list is usually used for DAC. it is compact and easy to review, deleting an object is simple but for subjects is more difficult. contrariwise, capacities list is easy for delegation and more difficult fr revocation.

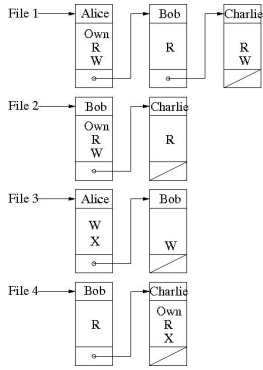


Figure 3: access matrix list

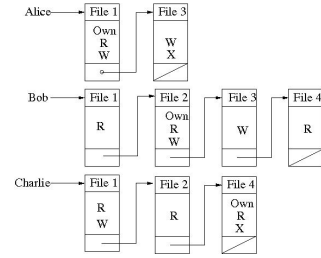


Figure 4: capacities list

4 OUTLINE

4.1 RBAC

Usually we deal with a structure which is composed of a great number of subject or objets. The two model seen above Acces Control Matrices and Access Control Lists are difficult to maintain with so many entries. To overcome this difficulty we can see that in a given structures many subjects have the same attributes so we can make a policy that subjects with specific attributes would have the same rights. This will help to make the structure more scalable. Another way is to determine the rights of the subject according to hierarchy. So for example a subject that is in high hierarchy have more rights than someone down; A director can have acces to some part where an simple employ cannot.To manage better the Acces Policy we introduce Role-Based Access Control which consist in decoupling users and permissions by introducing roles. We can formalized the Acces Control as : $AC := Roles \times Permissions \text{ rond } User \times Roles \text{ UA and } (r,p) \in PA \}$ Example The advantage of using RBAC is that roles are abstraction of jobs or function in an organisation and so it makes the policies more managable. Also less information is maintained when the number of roles is small. We can even optimized RBAC by introducing partial order $<$ on roles. Therefore larger roles inherit permissions from smaller roles. We can also apply hierarchies on users and permissions or introduce notion of session representing users' active roles.