Formalism of the access control model based on the marked Petri Nets

Abderrahim Ghadi[1, 2], Driss Mammass[1], Maurice Mignotte[2], Alain Sartout[2]
[1] *IRMA*, 7 Réné Descarte 67084, Strasbourg-France
[2] IRFSIC, FSA université Ibn Zohr B.P 8106, Agadir Maroc
{ghadi, mignotte, sartout}@math.u-strasbg.fr, driss_mammass@yahoo.fr

Abstract

Access control system is a mandatory step in the security policy implementation. Accesses carried out according to certain modes (read, write ...) by subjects (users, process, programs) on objects (data, files, programs). To specify, implement and reason within a formal framework allow ensuring the correction of the programs that in turn guarantee the proper functioning of this policy.

This article deals, on the one hand, with designing a model within a formal framework in which both the comparison and the composition of access control policies can be expressed, and on the other hand, with elaborating new techniques ensuring the re-use of this model. Such formal models have already been developed but suffer from too many constraints, are not expressive enough and are specific to professional activities. They all deserve to be reconsidered and extended within a uniform framework in order to study their links and to come up with re-use techniques.

The main object of this work is the use of Petri Nets for system modeling. Our results enable us to combine formalism and new access control policies. Moreover, the access control policies are considered as the instances within a more general scheme.

Keywords: Computer security, Formal Models, Formalism, Privileges graph, Oriented Graph Petri Nets.

1. Introduction

The study of different systems is usually done by a model. To find a model system is to build an abstract object representing the structure and dynamism of observed system. Several access control models were proposed are DAC [7], MAC [8, 9], RBAC [10, 11, 12], TBAC [13] and TMAC [14]. None of these models is perfect, mainly because of the encountered difficulties to work in an organization.

This study is organized as follows: in section 2, we present a three principals family of formalisms and we give classification methodology In the next section we introduce the Petri nets concept and the applications of the marked Petri Nets in the access control model for the Unix/Linux systems.

2. Formalism

In the following, we give an abstract of different formalisms types and those classifications in order to find the adequate formalism for Unix/Linux systems. First we give some used terminologies:

- ➔ An entity is a discrete element of a system having a stat and behavior,
- ➔ A relation is a semantic link between many entities.

- ➔ Objects: the things to be protected, e.g., files,
- ➔ Subjects : users, groups, roles,
- ➔ Matrix entries: access rights, i.e., operations allowed by a subject on an object.

2.1 Formalism for structural modeling

Structural modeling is either the description of the entities forming the system and their comparison or the description of the variables characteristic of the system processes.
This kind of formalism describes the static relations between the system entities. The notion of semantic link cover mainly three aspects:

- ○ The interaction link : two knowing entities can exchange information,
- ○ Composition or aggregation link : an entity is an association of a several entities and an hierarchical relation is established,
- ○ A heritage link: an entity is a generalization of another entity. The entity properties are made by the union of the most abstracted entity properties and its own properties.

The modeling by objects and agents is an example of formalism of structural modeling.

2.2. Formalism of dynamical modeling

This kind of formalism describe the system entities behavior, means that the temporal evolution of the state of the system. For example, we can mention these formalisms :

- ➔ Differential equations Systems,
- ➔ Finite-states and finite-events automata,
- ➔ Dynamical diagrams UML : state charts

2.3. Hybrid formalisms

The last family of formalisms consists of formalisms where separation between structure and behavior is difficult. For examples:

- ➔ Petri Nets,
- ➔ Cellular Automata.

2.4. Formalisms classification

We have presented some examples of formalisms used in modeling of complex systems. Each formalism defines the aspect of a system: The structure or behavior description, space description... Thus, we can establish a classification of the formalisms according to the aspects to be modeled and of the properties of system or subsystem. The object of this classification is to prove that it is fundamental to choose the formalism according to the system.
In order to establish a classification, it is necessary to determine the discriminating properties of the system:

- ➔ Discrete or continuous system : this property is to consider according to several points of view, change of states, spaces, time, etc

- ➔ The modeled processes are deterministic or stochastic?
- ➔ The space is taken into account?
- ➔ Does time have an importance?

The following figure, proposed in [5], draws up a formalisms classification dedicated to the specification of the dynamics of the systems according to three criteria : The handling of continuous or discrete variables, the representation of continuous or discrete space and the presence of continuous or discrete time.

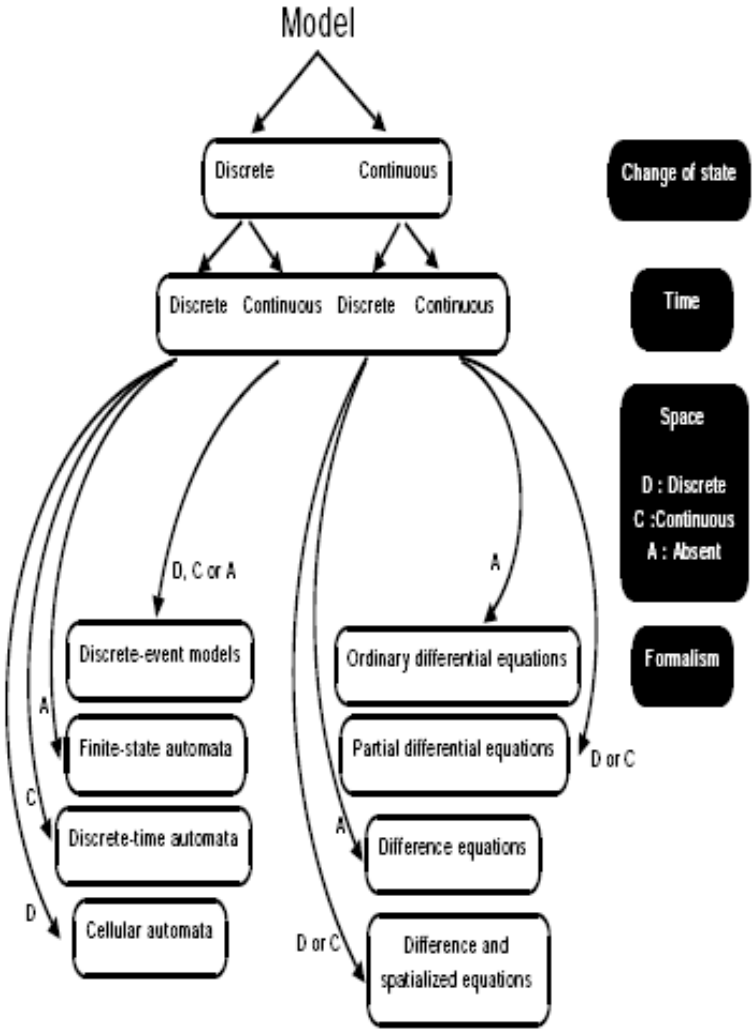

Figure 1. Formalisms classification

In our case, we want to formalize the access control of Unix/Linux system. Since the time and change of state in our system are two discrete variables, and that the space aspect is absent, then we chose the Petri Nets formalism, which is an extension of the finite-states automata.

3. Petri Nets formalism

Access Control Matrix or Access Matrix is an abstract formal computer protection and security model used in computer systems, which characterizes the rights of each subject with respect to every object in the system:

A Petri net is one of several mathematical representations of discrete systems. As a modeling language, it graphically depicts the structure of a distributed system as a bipartite oriented graph with annotations. Such as a Petri Net have place nodes, transition nodes, and oriented arcs connecting places with transitions.

A Petri net consists of places, transitions, and graph. Arcs run between places and transitions, not between places and places or transitions and transitions. The places from which an arc runs to a transition are called the input places of the transition, the places to which arcs run from a transition are called the output of the transition. In our case places contain a number of tokens. A distribution of tokens over the places of a net is called a marking. Transitions act on input tokens by a process known as firing. A transition is enabled if it can fire, i.e., there are tokens in every input place. When a transition fires, it consumes the tokens from its input places, performs some processing task and places a specified number of tokens into each of its output places.

All the Unix/Linux files are stored in repertories which can also contain other repertories. The file system structure is comparable with a tree of which the root is a repertory and whose sheets are the files. Each object (file or repertory) belongs to a user and a group. Any user can manage three types of rights on the objects of which he is the owner: read, write, execute (in addition to the rights SUID, SGID and sticky bit). It can grant or refuse each one of three distinct users sets: owner, members of object owner group and all other users.

The graph nodes can be gathered in two principal types: the first represents the set of all privileges which a given user has in the initial protection matrix, and thus a node for each user. The second node type represents a subset of common privileges to a user group.
The arcs are also gathered in two types. An arc from Node N1 towards a N2 node can mean that: first, the set of privileges into N2 is a subset of that defined into N1, second it exists a method which allows a user having the only privileges defined into N1 to acquire those defined in N2
This work shows how to define a formal framework ready to represent the access control systems in the UNIX /Linux system. We use the marked Petri Nets modeling to define a formal mathematical framework.

4. Conclusions

By dint of privilege graph and Petri nets, it is possible to obtain an image as precise as possible of the global safety of system. Thus we can improve the security policy and we identify three different phases in this process: first of all,
The graph construction, then it should be proven that privileges transfers are licit. Finally it is necessary to compare the obtained graph with objective of safety systems.

5. References

[1] A. Chander, J.C. Mitchell, and D. Dean. A state transition model of trust management and access control. In Proceedings of the 14th IEEE Computer Security Foundations Workshop CSFW, pages 2743. IEEE Computer Society Press, 2001.
[2] M. Jaume and C. Morisset. A formal approach to implement access control. Journal of Information Assurance and Security, 2:137148, 2006.
[3] M. Jaume and C. Morisset. Towards a formal specification of access control. In P. Degano, R. Kusters, L. Vigano, and S. Zdancewic, editors, Proceedings of the Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis, FCS-ARSPA06, pages 213232, 2006.
[4] McLean. The algebra of security. In Proc. IEEE Symposium on Security and Privacy, pages 27. IEEE Computer Society Press, 1988.
[5] R AMAT, (2003). Contributions à la modélisation et à la simulation des systèmes complexes. Habilitation à diriger des recherches.
[6] M.V. Tripunitara and N. Li. Comparing the expressive power of access control models. In SIGSAC:
11th ACM Conference on Computer and Communications Security. ACM SIGSAC, 2004.
[7] M.C. Tschantz and S. Krishnamurthi. Towards reasonability properties for access- control policy languages. In D.F. Ferraiolo and I. Ray, editors, SACMAT 2006,11th ACM Symposium on Access Control Models and Technologies, Proceedings, pages 160 –169. ACM, 2006.
[8] K. J. Biba. Integrity consideration for secure computer systems. Technical Report MTR-3153, The MITRE Corporation, Juin 1975.
[9] D. E. Bell et L. J. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Technical Report ESD-TR-73-306, The MITRE Corporation, Mars 1976.
[10] R. Sandhu, E. J. Coyne, H. L. Feinstein et C.E. Youman. Role-based access control models. IEEE Computer, 29(2):38-47, 1996.
[11] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn et R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and System Security, 4(3):222-274, Rencontres Aot 2001.
[12] S. I. Gavrila et J. F. Barkley. Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management. Third ACM Workshop on Role-Based Access Control, pages 81-90, 22-23 October 1996.
[13] R. Thomas et R. Sandhu. Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management. 11th IFIP Working Conference on Database Security, Lake Tahoe, California, USA, 1997.
[14] Roshan K. Thomas. TMAC: A primitive for Applying RBAC in collaborative environment. 2nd ACM, Workshop on RBAC, pages 13-19, Fairfax, Virginia, USA, 6-7 November 1997.
[15] B. Lampson. Protection. 5th Princeton Symposium on Information Sciences and Systems, pages 437-443, Mars 1971.