A Discretionary Access Control Model with Temporal Authorizations

Elisa Bertino Claudio Bettini Pierangela Samarati

Dipartimento di Scienze dell'Informazione
Università di Milano
Milano, Italy

Abstract

Conventional authorization models enforcing discretionary policies are based on authorizations which specify, for each user or group of users in the system, the accesses he is allowed to execute on objects. We propose a new authorization model which allows to associate with each authorization temporal constraints which restrict the validity of the authorization. Moreover, in our model we allow the specification of temporal dependencies among authorizations. Temporal dependencies allow the derivation of new authorizations on the basis of the presence or of the absence of other authorizations in given time intervals. In the paper we present the main characteristics of our model and illustrate some research issues we are currently investigating.

1 Introduction

The area of database security has been recently receiving much attention from both researchers and system developers. Indeed, from the industrial side, systems are now available (e.g., Trusted Oracle [13]) that are secure against attacks like Trojan Horses and some types of covert channels [1, 6, 5, 9, 15]. From the research point of view, several directions can be devised concerning respectively discretionary access control mechanisms and mandatory access control mechanisms. Along the first direction, current access control models are being extended to meet requirements deriving from more articulated security policies and from new data models, like object-oriented data models and deductive data models [14, 4, 10, 7, 17]. In particular, the increased awareness of data security in application environments has resulted in the need of flexible authorization models able to directly support application security policies. Examples of flexible authorization models include models with negative authorizations [3], with exceptions, with roles and tasks [8, 12, 18]. Along the second direction, mandatory access control techniques are being extended to deal with new-generation DBMS, like OODBMS [19, 2, 11] and active DBMS [16], as well as to develop techniques to deal with sophisticated attacks, like those through covert channels.

The work reported in this paper deals with the first of the above research directions. The goal of this work is to extend a conventional authorization model with temporal information. The new aspects of our model can be summarized as follows.

First, our model supports the notion of the temporal interval of validity of authorizations. In our model it is possible to specify that an authorization will expire after a specified point in time, or that an authorization is valid only in a specified temporal interval. Note that in many real-life situations authorizations are limited in time - consider for example a badge which is valid for only one day.

Second, our model provides the concept of temporal dependency among authorizations. A temporal dependency can be used, for example, to specify that a user has an authorization as long as another user has this authorization. This type of capabilities is very useful in many advanced applications, like for example CSCW applications.

Finally, besides proposing a basic set of operators to specify temporal dependencies, we introduce a formalism to concisely express many dependencies. For example, a single statement can specify that a user can read *all* the files that another user can read, relatively to an interval of time. To our knowledge, our work is the first to propose an authorization model with those temporal capabilities. Note that many new applications, such as office automation, CAD, CSCW, require such authorization functionalities.

The remainder of this paper is organized as follows. Section 2 illustrates the main characteristics of our proposal. Section 3 presents the formal definitions for our model. Section 4 discusses some problems concerning the management of authorizations in our model. Section 5 discusses some research issues which we are currently investigating. Finally, Section 6 presents the conclusions.

2 Overview of the authorization model

In this section we illustrate the characteristics of our model. The new features of our authorization model can be summarized as follows:

Temporal validity for authorizations Each authorization is associated with a time interval.

An authorization is valid only during the associated time interval and expires after the time interval elapses.

Temporal dependencies among authorizations A temporal dependency specifies that an authorization depends on another authorization under a *dependency mode*. Possible dependency modes are: WHENEVER, ASLONGAS, WHENEVERNOT, UNLESS. Intuitively, the semantics of the dependency modes is as follows.

- Dependency "A_1 WHENEVER A_2" states that authorization A_1 can be considered valid every time authorization A_2 is valid.
- Dependency "A_1 ASLONGAS A_2" states that if authorization A_2 is valid at the time at which the dependency is specified, then also authorization A_1 is valid and it will remain valid until authorization A_2 remains valid.

The ASLONGAS dependency mode, like the WHENEVER dependency mode, allows to derive authorization A_1 from authorization A_2.

The difference between the two modes is as follows. In order to derive authorization A_1 at a given time, the dependency mode WHENEVER requires authorization A_2 to be valid at that time. By contrast, the dependency mode ASLONGAS requires A_2 to have been valid since the time the dependency has been specified.

- Dependency "A_1 WHENEVERNOT A_2" states that A_1 is valid every time A_2 is not valid.
- Dependency "A_1 UNLESS A_2" states that if A_2 is not valid at the time at which the dependency is specified, then A_1 becomes valid and it will remain valid until the last instant before the one at which A_2 becomes valid.

The difference between the UNLESS and WHENEVERNOT modes is analogous to the difference between ASLONGAS and WHENEVER. In particular, in order to derive authorization A_1 at a given time, dependency mode WHENEVERNOT requires authorization A_2 to be not valid at that time. By contrast, the dependency mode UNLESS requires that A_2 has not been valid since the time the dependency has been specified.

A temporal dependency among two authorizations, together with the instant in which it is inserted, it is called *derivation rule.*

Parametric derivation rules These rules are used to concisely define derivation rules. Parametric derivation rules are rules which can be applied to different subjects, objects, or access modes.

Derivation rules allow the derivation of new authorizations on the basis of the presence or of the absence of other authorizations.

The following example illustrates derivation rules.

Example 2.1 Consider the authorizations and derivation rules illustrated in Figure 1. A_1, A_2, and A_3 are temporal authorization rules; their validity intervals are $[10, 20]$, $[30, 40]$, and $[15, 50]$, respectively. R_1, R_2, R_3, R_4, and R_5 are derivation rules expressing dependencies among authorizations. The time associated with each rule is the time at which the rule has been specified. Rule R_5 is parametric since it has a parameter, instead of a specific value, for the access mode of the authorizations. The use of the parameter allows to apply the rule for different values of the access mode field.

According to the semantics of the temporal dependency mode, the following authorizations can be derived from the rules above.

R_1: From rule R_1 and authorization A_1, authorizations ([5,9], (John,o_1,read)), ([21,29], (John,o_1,read)), and ([41,∞], (John,o_1,read)) can be derived. That is, the authorization (John,o_1,read) can be derived for every time instant in which authorization (Alice,o_1,read) is not valid.

R_2: From rule R_2 and authorization A_1, authorization ([6,9], (Bob,o_1,read)) can be derived, where 6 is the instant at which R_1 is inserted, and 9 is the instant preceding the first instant at which (Alice,o_1,read) becomes valid. Note the difference between this rule and the analogous rule R_1 with the WHENEVERNOT dependency mode.

R_3: From rule R_3 and authorizations A_1 and A_2, authorizations ([13,20], (Sam,o_1,read)), and ([30,40], (Sam,o_1,read)) can be derived.

R_4: From rule R_4 and authorization A_1, authorization ([14,20], (Matt,o_1,read)) can be derived, where 14 is the instant at which R_4 is inserted, and 20 is the instant preceding the first instant at which (Alice,o_1,read) becomes not valid. Again note the difference between this rule and the analogous rule with the WHENEVER dependency mode (R_3).

R_5: From rule R_5 and authorizations A_1, A_2, and A_3, authorizations ([10,20], (Ann,o_1,read)) ([30,40], (Ann,o_1,read)), and ([10,50], (Ann,o_1,write)) can be derived. Note that, being the considered rule parametric with respect to the access mode, it has allowed the derivation of authorizations with different access modes.

□

The authorizations considered valid at a given time are, beside the authorization explicitly specified, the authorizations derivable through the application of derivation rules. Then, every time an access request is submitted to the system, the access control is executed to determine whether the access is authorized. The access is authorized if either an explicit authorization exists for it or an authorization for it can be derived by the application of the rules. Note however, that

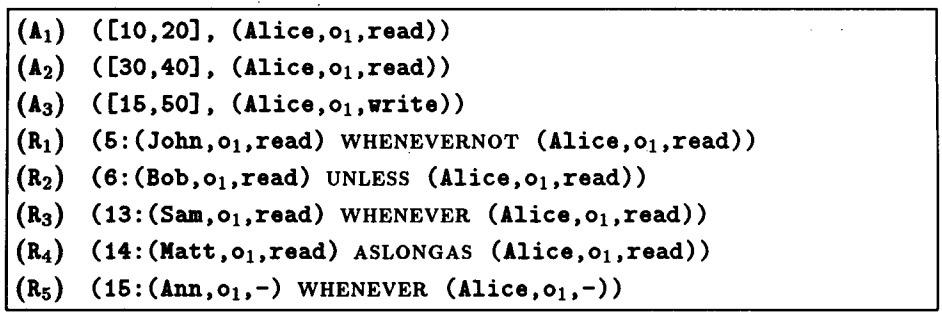

Figure 1: An example of authorizations and derivation rules

authorizations derived by the rules are not explicitly stored in the authorization base, rather, the existence of a derived authorization for a given time instant is determined at the time the access is requested. Indeed, explicitly storing derived authorizations would require continuous modification of the derived authorizations upon modification, addition, or removal of other authorizations.

3 Formal semantics

In this sections we formalize the concepts introduced earlier.

Definition 3.1 (Authorization) *An authorization is a triple* (s,o,m) *where*

$s \in S$ *is the subject (user) to whom the authorization is granted*

$o \in O$ *is the object on which the authorization is granted*

$m \in M$ *is the access mode, or privilege, for which the authorization is granted.*

Triple (s,o,m) states that user s is authorized to execute access mode m on object o.

Definition 3.2 (Temporal authorization) *A temporal authorization is a pair* (time, auth), *where* time *is a time interval* $[t_i, t_j]$, *with* $t_i \in \mathbb{N}$, $t_j \in \mathbb{N} \cup \infty$, $t_i \leq t_j$, *and* auth *is an authorization.*

Temporal authorization $([t_1,t_2],(s,o,m))$ states that subject s is allowed to exercise access mode m on object o in the interval $[t_1,t_2]$ including time instants t_1 and t_2.

A simple authorization, i.e., an authorization without any temporal constraint, can be represented as a temporal authorization whose validity spans from the time to which the authorization is granted to infinity.[1]

[1] This corresponds to having an implicit ALLTIME operator, often used in temporal logics [20].

As already mentioned, starting from the specified temporal authorizations, new authorizations can be derived through rules. Derivation rules are defined as follows.

Definition 3.3 (Derivation rule) *A derivation rule is defined as* $(t_r : A_1 \langle \text{dep-mode} \rangle A_2)$, *where* t_r *is the time at which the rule has been specified,* A_1 *and* A_2 *are authorizations, and* ⟨dep-mode⟩ *is one of the following dependency modes:* WHENEVER, ASLONGAS, WHENEVERNOT, UNLESS.

The semantics of a derivation rule depends on the dependency mode used in the rule as illustrated in Section 2.

Unlike authorizations, derivation rules do not have associated time intervals. A rule is considered valid from the time t_r of its insertion until the time it is deleted or infinity.

Definition 3.4 (Parametric derivation rule) *A parametric derivation rule is a derivation rule where symbol "–" appears for subjects, objects, or access modes in the authorizations. If symbol "–" appears in an authorization of the rule, it must appear, in the same position, also in the other authorization.*

Symbol "–" is a parameter which denotes any subject, object, or access mode depending on its position in the authorization.

A *Temporal Authorization Base (TAB)* thus consists of the union of the temporal authorizations and the derivation rules.

The following section illustrates the management of the authorization base and discusses some issues which must be considered in the administration of authorizations in our model.

4 Administration of authorizations

In our model, we allow the administrator to modify the TAB by adding, removing, or modifying temporal authorizations and derivation rules. The administrative operations which the administrator can execute are as follows.

GRANT To grant a privilege on an object to a subject. The grant operation results in the addition of a new temporal authorization.

REVOKE To revoke a privilege on an object from a subject. The revoke operation results in the deletion of all the temporal authorizations of the subject for the privilege on the object. The revoke operation can be used also to remove a specific authorization of the subject.

MODIFY To modify the temporal constraint of an authorization previously granted.

ADDRULE To add a new derivation rule.

DROPRULE To drop a derivation rule previously specified.

Note that execution of administrative operations requires particular care in our model. In particular, also if the administrator revokes an authorization or drops a rule, the considered authorization/rule may not be deleted by the TAB. The same applies when an authorization expires. This is due to the fact that evaluation of temporal dependency modes such as UNLESS and ASLONGAS requires evaluating the validity of some authorizations in a past time interval. Therefore, uncontrolled deletion of authorizations or rules, even if revoked or expired, may bring to an incorrect evaluation of other authorizations later on.

To illustrate, consider the TAB of Figure 1. At time 20 authorization A_1 expires. However, it cannot be deleted from the authorization base because it is needed for the application of rule R_4.

For this reason, we consider that every time a REVOKE/DROPRULE operation is required, the corresponding authorization/rule is not removed, rather is tagged as unusable with the time t_d at which the administrator asked for its deletion.

We are investigating an algorithm for the maintenance of the TAB. The algorithm is based on the classification of temporal authorizations and derivation rules as follows:

Withdrawn Temporal authorizations and derivation rules whose removal has been explicitly required (with a REVOKE/DROPRULE command) by the administrator.

Expired Temporal authorizations and derivation rules which have not been withdrawn but which are not applicable anymore. They are authorizations whose end-time has passed and UNLESS/ASLONGAS rules from which no authorizations can be derived anymore.[2]

Active Temporal authorizations and derivation rules not withdrawn nor expired.

[2]The semantics of WHENEVER and WHENEVERNOT rules implies that they cannot expire.

The algorithm we are investigating periodically examines the TAB and eliminates the expired and withdrawn authorizations which are not needed (and will never be needed again in the future). Developing the algorithm requires to devise a mechanism which determines whether a given authorization/rule may need to be evaluated in the future because of the presence of an ASLONGAS or UNLESS rule. The algorithm for maintaining the TAB must be proved to be correct, i.e., not to remove authorization which will be needed again. To formally prove the correctness of the algorithm we are formalizing a set of properties which characterize whether a given authorization/rule may still have effect in the evaluation of the TAB or is, and will always be, ineffective. Obviously, ineffective authorizations/rules will be the only rules which the algorithm can remove.

Another issue regarding the administration of authorizations concerns the consistency of the authorization base. We proved that we cannot have sets of rules leading to an inconsistent authorization base. However, in our model authorizations can be derived, through derivation rules, on the basis of the presence or the absence of other authorizations. Then, the interaction of different types of rules, those operating on the presence of authorizations (WHENEVER and ASLONGAS) and those operating on the absence of authorizations (WHENEVERNOT and UNLESS) may have undesirable effects. We have characterized such situations on the basis of the definition of *critical sets* of rules. A set of rules is said to be critical if, starting from the assumption that an authorization is not valid, it allows to derive the authorization. A simple critical set is the rule expressing the dependency "A_1 WHENEVERNOT A_1". This rule states that every time authorization A_1 is not valid, it is valid. Indeed, if A_1 is not in the TAB nor can be derived by other rules, then through this rule, it can be derived. Critical sets may result from the combination of several rules. Although only the administrator can modify the TAB, it can happen that he inserts a rule which triggers a critical set formed by a long rule chain. We do not allow critical sets of rules in our model. We are developing an algorithm for the identification of critical sets of rules.

5 Research Issues

The work reported in this paper is currently being carried out by the authors and several issues are being investigated.

A first issue concerns the development of algorithms and tools for the representation and maintenance of the authorization base. The main complexity of our authorization model derives from the need to use an inference mechanism to derive authorizations from the authorizations stored into the authorization base. Therefore, this inference process must be enhanced by using techniques similar to those proposed for view materializations in relational databases and deductive databases. Administration tools are particularly crucial when dealing with sophisticated authorization models. In our model, for example, it is im-

portant to develop a tool providing information about derivation rules involved in critical sets.

A second issue we are investigating concerns the administration of authorizations. In the paper we have made the hypothesis that the administration is centralized, i.e., only the administrator is allowed to modify the authorization base. An alternative decentralized policy can be devised. However, the application of such a policy would require particular consideration of the temporal constraints associated with the authorizations and their modification. For example, a user should be authorized to grant an authorization for a time interval if it does not have that authorization in that specific interval. Then upon granting of an authorization, the temporal constraints associated with the authorizations of the grantor must be propagated to the authorization being granted. Moreover, possible modifications to the temporal constraints associated with the authorizations of a user must be propagated to the authorizations the user has granted.

A third issue we are investigating concerns the association, with each rule, of a time interval of validity as done for the authorizations. In the paper we have considered a rule is valid from the time it is inserted until the time it is deleted. The model can be easily extended to the consideration of time interval associated with rules.

A further direction we plan to investigate concerns the introduction of negative authorizations. Negative authorizations are authorizations which specify the denial for a user to execute a particular access. Rules can be devised which allow the derivation of either positive or negative authorizations on the basis of the presence or absence of a denial. The consideration of both positive and negative authorizations may lead to cases of inconsistencies which therefore will need to be investigated.

6 Conclusions

In this paper, we have presented a survey of a temporal authorization model. This model has several innovative features, such as temporal dependencies among authorization rules. We believe that such features are crucial for the support of many advanced applications. The main qualifying points of this work are three. First, our work proposes a model that directly supports the requirements of many application environments. Indeed, in many real-life situations, authorizations have a limited validity and they may depend from each other. Second, our authorization model is independent from any specific data model. It can be applied to relational database systems, as well as object-oriented database systems and deductive database systems. Third, it opens several new research issues that need both theoretical investigations as well as experimental work.

References

[1] V. Atluri, E. Bertino, and S. Jajodia. Achieving stricter correctness requirements in multilevel secure databases. In *Proc. IEEE Symposium on Security and Privacy*, pages 135–147, Oakland, California, May 1993.

[2] E. Bertino, L. Mancini, and S. Jajodia. Collecting garbage in multilevel secure object stores. In *Proc. IEEE Symposium on Security and Privacy, Oakland, California*, May 1994.

[3] E. Bertino, P. Samarati, and S. Jajodia. Authorizations in relational database management systems. In *Proc. First ACM Conference on Computer and Communications Security*, Fairfax, Virginia, November 1993.

[4] E. Bertino and H. Weigand. An approach to authorization modeling in object-oriented database systems. *Data and Knowledge Engineering*, 12(1), 1994.

[5] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security policies. In *Proc. IEEE Symposium on Security and Privacy*, pages 184–194, Oakland, California, April 1987.

[6] O. Costich. Transaction processing using an untrusted scheduler in a multilevel secure database with replicated architecture. In C.E. Landwehr, editor, *Database Security, V: Status and Prospects*, pages 173–189. North-Holland, Amsterdam, 1992.

[7] K. Dittrich, M. Hartig, and H. Pfefferle. Discretionary access control in structurally object-oriented database systems. In C.E. Landwehr, editor, *Database Security, II: Status and Prospects*, pages 105–121. North-Holland, Amsterdam, 1989.

[8] J.E. Dobson and J.A. McDermit. Security models and enterprise models. In C.E. Landwehr, editor, *Database Security, II: Status and Prospects*, pages 1–39. North-Holland, Amsterdam, 1989.

[9] V. Doshi and S. Jajodia. Referential integrity in multilevel secure database management systems. In G.G. Gable and W.J.Caelli, editors, *IT Security: The Need for International Cooperation*, pages 359–371. North-Holland, 1992.

[10] E. B. Fernandez, E. Gudes, and H. Song. A security model for object-oriented databases. In *Proc. IEEE Symposium on Security and Privacy*, pages 110–115, Oakland, California, May 1989.

[11] S. Jajodia and B. Kogan. Integrating an object-oriented data model with multilevel security. *Proc. IEEE Symposium on Security and Privacy, Oakland, California*, pages 76–85, May 1990.

[12] D. Jonscher, J.Moffett, and K. Dittrich. Complex subjects or: the striving for complexity is ruling our world. In *Proc. of the 7th IFIP WG 11.3 Workshop on Database Security*, pages 18–36, Huntsville, Alabama, September 1993.

[13] W. T. Maimone and I. B. Greenberg. Single-level multiversion schedulers for multilevel secure database systems. In *Proc. 6th Annual Computer Security Applications Conf.*, pages 137–147, Tucson, Arizona, December 1990.

[14] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A model of authorization for next-generation database systems. *ACM Trans. on Database Systems*, 16(1):88–131, March 1991.

[15] W.R. Shockley, M. Heckman, R.R. Schell, D.E. Denning, and T.F. Lunt. The SeaView security model. *IEEE Transactions on Software Engineering*, 16(6):593–607, June 1990.

[16] K.P. Smith. *Managing rules in active databases.* PhD Thesis, December 1992.

[17] D. L. Spooner. The impact of inheritance on security in object-oriented database systems. In C.E. Landwehr, editor, *Database Security, II: Status and Prospects*, pages 141–160. North-Holland, Amsterdam, 1989.

[18] G. Steinke and M. Jarke. Support for security modeling in information systems. In B.M. Thuraisingham and C.E. Landwehr, editors, *Database Security, VI: Status and Prospects*, pages 125–141. North-Holland, Amsterdam, 1993.

[19] M. B. Thuraisingham. Mandatory security in object-oriented database system. In *Proc. Conf. on Object-Oriented Programming: Systems, Languages,and Applications*, pages 203–210, October 1989.

[20] Johan van Benthem. Temporal logic. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of logic in artificial intelligence and logic programming*, volume 3. Oxford University Press, 1991.