Towards a task-based paradigm for flexible and adaptable access control in distributed applications

R. K. Thomas and R. S. Sandhu[*]

Center for Secure Information Systems
&
Department of Information and Software Systems Engineering
George Mason University
Fairfax, VA 22030-4444

Abstract

Historically, the access control problem has been couched within the framework of subjects, object, and rights. In this paper we argue for a newer paradigm for distributed and multi-system applications, that transcends the subject-object view of access control. This new paradigm views access control and authorization not in terms of individual subjects and object, but rather in terms of long-lived tasks that need to be authorized and managed in information systems.

1 Introduction

Historically, the access control problem has been couched within the framework of subjects, objects, and rights (access types). An access control request thus essentially seeks an answer to a question posed typically as: Is subject s allowed access a (or possess the right a) to object o? A tuple (s, o, a), which we define as an *authorization*, can be input to a function f, which returns true (or false), to indicate if the subject s has the right a (or not) to object o. We can visualize the implementation of such a function with an access control matrix. This subject-object view can be traced to the subject-object paradigm of access control that was formulated in the early era of the development of general multi-user computers and operating systems [7, 5].

Over the last two decades we have seen considerable advancements in the discipline of computer security. In particular, we have seen the evolution and development of many access control models. The initial proposals of Lampson [7] and Graham and Denning [5] led to formulation of the *HRU model* by Harrison, Ruzzo, and Ullman [6]. This was followed by the development of the *Take-Grant Model*. A good summary of these early efforts (in the first decade) can be found in [13]. More recent efforts have resulted in the *Schematic Protection Model* (SPM) by Sandhu [8], the *Extended Schematic Protection Model* (ESPM) by Amman and Sandhu [1], and the *Typed Access Matrix Model* (TAM) also by Sandhu [12].

In reviewing the above development in access control models, we note that the overriding concern was the fine-grained protection of individual objects and subjects in the system. However, with the advent of databases, networking, and distributed computing, we have witnessed a phenomenal increase in the automation of organizational tasks, as well as the computerization of information related services.

In light of this, is it not fitting that we shift our focus on security issues from the protection of individual objects and subjects in isolated computer systems, to the automation and provision of distributed tasks and services? Such tasks may involve groups of related activities that span multiple networks and databases. Authorization (access control) may be required for groups of related activities at several departments and may even organizational boundaries. Thus, we believe it is timely and necessary to transcend the above classical subject-object view of access control, and work towards newer paradigms.

[*]A preliminary version of this paper was presented at the sixteenth National Computer Security Conference, Baltimore, MD., and appears in the conference proceedings.

©1993 ACM 0-89791-635-2 $1.50

2 Task-based Authorization

In this section we elaborate on the central point of this paper. In a nutshell, authorizations in distributed applications should be seen in terms of tasks or activities rather than individual subjects and objects. We argue this, based on two emerging trends:

1. The integration of computing within organizations, and the subsequent increase in the automation of organizational functions and work-flows.
2. The shift from main-frame computer systems to workstations and client-server technologies.

With the first trend, we are witnessing an increased demand for the support of multisystem applications. Such applications may even cross departmental and organizational boundaries. A very good example of this in the telecommunications industry is that of service order provisioning [2]. This is the automated process of providing telephone services to customers. Upon receiving a service request, a service order is generated. The processing of the service order demands coordination and data exchange between several business units in the company, and eventually leads to the assignment of lines and equipments, as well as the update of billing information (among others). As another illustration, consider the automation of a paper-based sales order processing application (system). Sales order processing begins with the receipt of a customer purchase order. The subsequent processing steps may involve several documents such as sales orders, invoices, customer statements, and journal vouchers. These documents may propagate through several departments in the organization(s) such as SALES-ORDER, CREDIT, FINISHED-GOODS, SHIPPING, BILLING, and ACCOUNTS-RECEIVABLE, completing the many subtasks involved in processing the sales order request.

The above documents would have to undergo a sequence of authorization or approval phases. For instance, a sales order may be routed through the CREDIT department, and shipment authorized only after a credit check on the customer succeeds. An organization may also incorporate various controls and checks to minimize risks due to fraud. One way to achieve this is through separation of duties. Custodial functions performed by FINISHED-GOODS and SHIPPING departments are separated from the recording functions of BILLING and ACCOUNT-RECEIVABLE, and the authorization functions performed by the SALES-ORDER and CREDIT departments. Separation of duties among individuals can ensure that only goods intended for shipment to customers are removed from the FINISHED-GOODS storeroom, and that all such goods are shipped only to authorized customers and are billed correctly.

The need for task-based authorizations arises even within the world of a single user in an office, accomplishing a simple and routine task such as printing from a workstation. Resources such as printers, files, and applications, may be shared over a local area network. The printing of a multimedia document, for example, may require authorization and access to multiple servers, and data stored at several objects.

A task, as identified in the scenarios above, may characterized as one that:

- is long-lived;
- may involve multiple subtasks, where a subtask may need to be individually or collectively authorized;
- may involve multiple and often distinct principals to carry out individual subtasks;
- is distributed in space and time.

We believe, that the authorization of tasks that span multiple systems over departmental and organizational boundaries, as well as those that involve individual workstations and servers, are conceptually similar and can thus be addressed in a unified manner.

As an initial attempt, we introduce the abstraction of an *authorization-task* as a unit for managing the authorizations in distributed applications. A *task* is a logical unit of work in such applications and may consist of several individual subtasks. In the earlier mentioned sales order processing system, when an order is taken, a corresponding authorization-task is begun. Individual authorization actions, such as the credit approval for a customer, can be done by finer units of authorization-tasks called *authorization-subtasks.*

3 Flexible and Adaptable Access Control

In the subject-object view of access control, every authorization tuple represents a primitive unit of access control information. Collectively, these tuples are unrelated to each other. Contrast this with the requirements of our application above, where the sales order processing task involves several related individual subtasks that need approvals (authorizations). Such a requirement calls for a higher level control structuring facility. An analogy to the above predicament can be seen in the realm of transactions and databases. Classical transactions with the ACID (Atomicity, Consistency, Isolation, and Durability) properties represent concurrent but unrelated units of work. Consider a requirement (restated from an example in [14]) for the sequencing of three transactions such as:

Execute T_1, followed by T_2 and T_3 in parallel; If T_2 fails, then abort T_3 as well.

With the transaction as the main control abstraction, it is impossible to implement the above without ad-hoc application programming. This has led researchers to propose other abstractions, such as the so-called ConTracts [14]. Authorizations in distributed applications similarly call for abstractions beyond individual subject-object authorizations.

We list some of the obvious questions that need to be answered on the road that could lead to a task-based authorization approach.

1. Abstraction and Modeling:
   What are the proper abstractions to express and manage the required authorizations for tasks?

2. Grouping Authorizations:
   How can groups of related activities be collectively authorized?

3. Flow Control and Dependencies:
   How can we describe and manage the control flow and dependencies between the authorizations of the various steps in a task?

4. Incorporation of Integrity Mechanisms:
   How can we incorporate controls such as those based on separation of duties and multiple approvals?

5. Failures, Exceptions, and Recovery:
   How can we handle failures in the authorization of individual steps of tasks? If a certain authorization/approval for a certain step in a task is not forthcoming, we may wish to specify alternate paths to be taken. For example, if the credit worthiness of a customer cannot be immediately established, the organization may have a policy that allows the sales order to go through, so long as the value of the items ordered does not exceed a certain amount. Or perhaps, in another scenario, we may wish to express that a certain authorization/approval step may be selectively ignored, under some conditions.

4 An Illustrative Example

To illustrate the intuition, flexibility, as well as generality, of task-based authorizations, let us take a concrete example that requires separation of duties. Suppose there already exists some predefined mechanisms and formalisms for expressing separation of duties. For example, Sandhu in [10, 11] has proposed *transaction control expressions* as an approach to implement separation of duties in computerized systems. It is based on a database activity model that utilizes the notions of *transient* and *persistent* objects. Transient objects include documents such as vouchers, purchase orders, sales slips, to name a few. These objects are transient in nature in the sense that they issue a finite set of operations and then leave the system (in a paper world this happens when a form is archived). These operations eventually affect persistent objects such as inventory databases, and bank accounts. The fundamental idea is to enforce controls primarily on the transient objects, and for transactions to be executed on persistent objects only as a side effect of executing transactions on transient objects.

Consider a check processing application where a clerk has to prepare a check and assign an account, followed by three (separate) supervisors who have to approve the check and account, and finally the check to be issued by a different clerk (in the paper world, this would be accomplished trough a voucher). This can be represented by the following transaction control expressions:

prepare • clerk;
3: approve • supervisor;
issue • clerk;

The colon is a voting constraint specifying 3 votes from 3 different supervisors. Each expression consists of a *transaction* and a *role*. Separation of duties is achieved by requiring the users who execute different transactions in the transaction control expression be all distinct.

Now consider a certain application that requires the use of two vouchers (transient objects). Now suppose the first voucher needs to be completely processed before the second one can be started. Further, we require separation of duties across these vouchers. We would proceed by defining an authorization-task that consists of two authorization-subtasks. Each subtask is assigned to a single transient object (a voucher in this example) and executes the transaction control expressions of the transient object. A subtask may specify the failure semantics within a transient object. If for

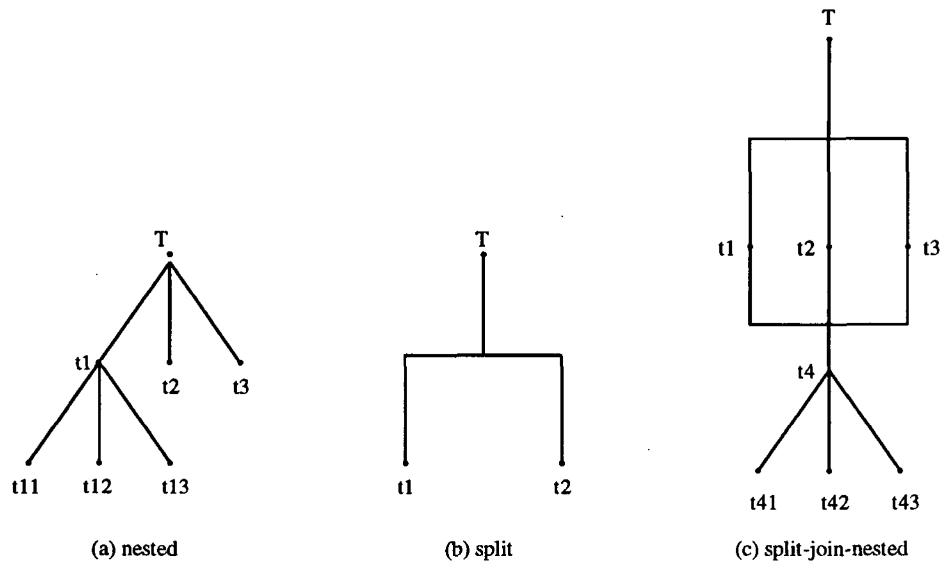

Figure 1: Models of authorization-tasks

example, the same clerk attempts to issue the check after preparing the voucher, the separation of duties requirement is violated. The authorization-subtask may then pursue some alternate action. A violation in the separation of duties across the two vouchers will be detected by the parent authorization-task. The authorization task may have to maintain global history and context information for this purpose.

5 Models of Authorization-tasks

In the last section, we discussed an example of an authorization-task that consisted of two authorization-subtasks. In general, authorization-tasks may form much more complicated structures due to the application semantics and dependencies between individual subtasks. It may be possible to analyze these dependencies and classify them as belonging to one or predefined categories. For each category we may generate a model of authorization-tasks to manage such dependencies.

Figure 1 illustrates some basic models that could be enumerated. In 1(a), we have an authorization-task T that is modeled as subtasks t_1, t_2, and t_3. The subtask t_1 in turn consists of subtasks $t_{1,1}$, $t_{1,2}$, and $t_{1,3}$, and hence the nested tree-like structure. A depth-first traversal of the tree would give the order in which the subtasks need to be completed. In addition to the relative order between the subtasks, we may also associate deadlines for the completion of individual subtasks. Figure 1(b) shows a split-model for an authorization task. This could be used to model the scenario where an individual requisition form has generated (split into) multiple forms, and where authorizations (signatures) can be granted independently on each of these forms. Finally, figure 1(c) illustrates a split-join-nested composite structure. In this case, after a split, the multiple requisitions are approved independently and later consolidated into one requisition. The consolidated task (requisition form) then has a nested structure of subtransactions.

6 Conclusions

In this paper, we have argued for a new paradigm for flexible and adaptable access control in distributed applications. We have motivated a task-based approach that represents a departure (and a paradigm shift) from the subject-object view of access control. Unlike the subject-object paradigm where the focus is on the doer, our paradigm focuses on what needs to be done. Our approach naturally leads to a transaction-based view of modeling and managing authorizations. We have presented authorization-tasks as a central abstraction for this purpose.

We believe that many of the ideas in the recent advances of transaction models are useful [4]. For example, one may specify dependencies and failure semantics between transactions, in a flexible way and often user-defined fashion. The insights here could be used to give more internal structure and semantics to authorization tasks. A long-term vision should be the development of a comprehensive framework (such as ACTA, for database transactions [3]) for specifying and reasoning about authorizations in distributed applications.

Acknowledgements

The work of both authors is partially supported by a grant from the National Security Agency, contract No: MDA904-92-C-5140. We are grateful to Pete Sell, Howard Stainer, and Mike Ware for their support and encouragement.

References

[1] Ammann, P.E. and Sandhu, R.S. "The Extended Schematic Protection Model." *Journal of Computer Security*, Volume 1, Numbers 3 and 4, 1992, pages 335-383.

[2] M. Ansari, L. Ness, M. Rusinkiewicz, and A. Sheth. Using flexible transactions to support Multi-system telecommunications applications. *Proc. of the 18th VLDB Conference*, British Columbia, Canada, 1992.

[3] P. K. Chrysanthis and K. Ramamritham. ACTA: A Framework for specifying and reasoning about transaction structure and behavior, *Proc. of the ACM SIGMOD conference*, pages 194–203, 1990.

[4] *Database Transaction Models for Advanced Applications*, A.K. Elmagarmid (Editor), Morgan Kaufmann Publishers, San Mateo, California, 1992.

[5] Graham, G.S. and Denning, P.J. "Protection - Principles and Practice." *AFIPS Spring Joint Computer Conference* 40:417-429 (1972).

[6] M.H. Harrison, W.L. Ruzzo, and J.D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8), pages 461–471, 1976.

[7] Lampson, B.W. "Protection." *5th Princeton Symposium on Information Science and Systems*, 437-443 (1971). Reprinted in *ACM Operating Systems Review* 8(1):18-24 (1974).

[8] Sandhu, R.S. "The Schematic Protection Model: Its Definition and Analysis for Acyclic Attenuating Schemes." *Journal of ACM* 35(2):404-432 (1988).

[9] Sandhu, R.S. "Separation of Duties in Computerized Information Systems." In *Database Security IV: Status and Prospects*, (Jajodia, S. and Landwehr, C.E., editors), North-Holland 1991, 179-189.

[10] R.S. Sandhu. Transaction control expressions for separation of duties. *Proc. of the Fourth Computer Security Applications Conference*, pp. 282–286, 1988.

[11] R.S. Sandhu. Separation of duties in computerized information systems. *Database Security IV, Status and Prospects*, S. Jajodia and C.E Landwehr (Editors), Elsevier Science Publishers B.V. (North-Holland)

[12] Sandhu, R.S. "The Typed Access Matrix Model." *Proc. IEEE Symposium on Research in Security and Privacy*, Oakland, California, May 1992, pages 122-136.

[13] Snyder, L. "Formal Models of Capability-Based Protection Systems." *IEEE Transactions on Computers* C-30(3):172-181 (1981).

[14] H. Wachter and A. Reuter. The ConTract Model. In *Database Transaction Models for Advanced Applications*, A.K. Elmagarmid (Editor), Morgan Kaufman Publishers, San Mateo, California, 1992.