Database Security

Teresa F. Lunt[1]
Computer Science Laboratory
SRI International
Menlo Park, California 94025

Eduardo B. Fernandez
Dept. of Computer Engineering
Florida Atlantic University
Boca Raton, Florida 33431

1 Introduction

Computer security is concerned with the ability of a computer system to enforce a security policy governing the disclosure, modification, or destruction of information. The security policy may be specific to the organization, or may be generic. For example, the DoD *mandatory security* (or multilevel security) policies restrict access to classified information to cleared personnel. *Discretionary security* policies, on the other hand, define access restrictions based on the identity of users (or groups), the type of access (e.g., select, update, insert, delete), the specific object being accessed, and perhaps other factors (time of day, which application program is being used, etc.). Different types of users (system managers, database administrators, and ordinary users) may have different access rights to the data in the system. The access controls commonly found in most database systems are examples of discretionary access controls.

This paper discusses discretionary security and mandatory security for database systems. We outline the current state of research in database security and briefly discuss some open research issues.

2 Discretionary Security

Discretionary Access Control (DAC) models can be represented by the access matrix model developed by Lampson in 1971 [23] and further refined by Graham and Denning [13]. This model defines an access matrix in which the rows represent *subjects* (users, processes), the columns represent *objects* (files, records, programs, subsystems, etc.), and the intersection of a row and a column contains the access types that the subject has authorization for with respect to the object. This model was extended with predicates and other components to make it more useful for database systems [4, 11, 18]. For example, predicates can represent content-dependent access control. Detailed discussions of the meaning of the model components can be found in Chapter 7 of [10] and in [9]. Implementation aspects are discussed in Chapter 10 of [10]. Harrison, Ruzzo, and Ullman [16, 17] showed that the general safety problem for this model is undecidable.

From 1970 through 1975, there was a good deal of interest in these models, in particular in their application to relational databases [11, 14, 18]. Then, most of the database security research turned to multilevel models (see Section 3). The appearance of semantic and object-oriented databases

[1]Lunt's work was supported by the U. S. Air Force, RADC, under contract F30602-89-C-0158.

has created a renewed interest in DAC models. These database models are much richer and more complex than the relational model, and aspects such as inheritance, semantic associations, and composite data structures allow the expression of flexible and arbitrarily specific discretionary policies. The research issues currently under investigation include:

a. A fundamental property of object-oriented systems is inheritance. This is often complemented with semantic associations such as aggregation or relationship. A class object may have many descendants or instances. If a user is authorized to access a class, should that user also be authorized to access all of its instances? Should this inherited authorization also include authorization for attributes that do not exist in the parent class?
b. For practical reasons it is necessary to define groups of users with similar authorizations. If authorizations can be defined both for individuals and for groups, or if a user can belong to more than one group, how should we determine the user's effective authorizations? If groups can be structured in hierarchies or partial orders, how does one define the effective authorizations of a given group?
c. Inheritance of authorization brings along the need to override inherited authorizations. One way to do this is to use negative authorization, which is given a higher priority than positive authorization. Another way is to use explicit authorization to override implicit inherited authorization. Related issues are the combination of access predicates defined a different levels in the object hierarchy and how to combine the effects of multiple inherited rules.
d. Should users own data or does all data belong to the institution? If we adopt the latter approach, then users with special authorizations (adminstrators) should handle the definition of authorization rules. Even in the latter case, should we still allow private subdatabases?
e. What framework can we use to ensure that new authorization rules satisfy institution policies?
f. How can we decentralize the functions of a security administrator, necessary for distributed environments?

Recent work on discretionary models include (the labels in parenthesis refer to which of the aspects above they have especially contributed):

- W. Kim and his group at MCC developed a detailed formal model for object-oriented systems using ORION as an illustration (the ORION object-oriented database system was also developed at MCC) [36]. Implied (inherited) authorization are obtained through the object hierarchy, i.e., from the class, to its components, to its instances. They use the concept of weak and strong authorizations, where strong authorizations can override weak authorizations (overriding is performed using negative authorization rules). (a)(c)
- U. Kelter at the University of Hagen, W. Germany, is developing models for distributed structurally object-oriented database systems [22]. In this type of database there is a hierarchy of nested, overlapping, complex objects describing typically a CAD or CASE design environment. Complex objects are the units of access control, and authorization conflicts may arise if complex objects share components. Group authorizations are inherited, that is, a lower group can inherit authorizations from a higher-level group, rather than through the data hierarchies. Explicit denial of authorzation (negative authorization) is handled using a four-valued logic. The first authorization model for this type of database was developed by K. Dittrich and his group at the University of Karlsruhe, W. Germany, for the DAMOKLES database system [7]. (a)(b)(c)

- E. B. Fernandez and his group at Florida Atlantic University, Boca Raton, Florida, developed an authorization model for object-oriented semantic databases specially tailored to OSAM*, a CAD/CAM database system being implemented at the University of Florida [24, 37]. This authorization model can also be applied to other data models. The model consists of a set of policies that define authorization inheritance through the data hierarchies, authorization evaluation algorithms (since the rules can be scattered through the hierarchy, tree searches are necessary), and an adminstrative structure that considers policies for delegation and revocation of administrative authorizations as well as the effect of database schema changes. Recent work also considers how to handle negative authorizations and the use of predicates [15]. (a)(c)(d)(f)
- J. D. Moffet and M. Sloman at the Imperial College of London use a model in which administration is separated from the use of the data [34]. They also include the concept of ownership, where owners are those entities in an enterprise, such as managers, that control the operations of the enterprise and that can grant administrative authorizations to security administrators. Recent work from this group considers decentralization aspects of their model [35]. (d)(e)(f)
- T. F. Lunt at SRI International, Menlo Park, California, has developed a model of discretionary security for SeaView, a secure multilevel database system [25, 26]. She is also modeling a discretionary access control mechanism that allows the implementation of arbitrary access control policies. The intent is to allow users to implement a discretionary access contol policy that is tailored to their application, rather than having to work around a specific policy that is wired into the computer system. (b)(c)

3 Mandatory Security

A multilevel database system supports data having different classifications or *access classes* and users having different clearances. In the most general case, the ability to individually classify atomic facts in a database is required. In the relational model, this means that data is classified at the level of individual data elements. Special cases of multilevel relations may be classified at the attribute level (i.e., all the data associated with a particular attribute has the same classification); at the row level (i.e., every tuple has a single classification); or at the relation level (i.e., all the data in the relation has the same classification).

The mandatory access control requirements are formalized by two rules, the first of which protects data from unauthorized disclosure, and the second of which protects data from contamination:

1. A subject S is not allowed to read data of access class c unless $class(S) \geq c$, and
2. A subject S is not allowed to write data of access class c unless $class(S) \leq c$.

In the above rules, a *subject* is a process acting on a user's behalf; a process has a clearance level derived from that of the user.

Mandatory security means that a multilevel relation will appear differently to users with different clearances, because not all data are authorized to all users. Other requirements include the ability to derive classification labels for derived data (as in views, for example).

We represent a multilevel relation R by a schema $R(A_1, C_1, \ldots, A_n, C_n)$, where each data attribute A_i has a corresponding *classification attribute* C_i. Figure 1 illustrates a multilevel relation

with three data attributes. In the figure, the label "S" means the data is classified SECRET; the label "TS" means TOP SECRET.

$A1$	$C1$	$A2$	$C2$	$A3$	$C3$
mad	S	17	S	x	S
foo	S	34	S	w	TS
ark	TS	5	TS	y	TS

Figure 1: Multilevel Relation R

The name of a relation R also has a classification, which we denote by $class(R)$. $Class(R)$ must be at least as low as the classification of any data contained in the relation. A relation R can be accessed by any user S where $class(S) \geq class(R)$. However, S can see only data that S is cleared to see. Figure 2 shows a SECRET view of the multilevel relation of Figure 1.

$A1$	$C1$	$A2$	$C2$	$A3$	$C3$
mad	S	17	S	x	S
foo	S	34	S	null	S

Figure 2: SECRET Relation Instance

Multilevel security affects the data model because not all data are visible to all users. One effect involves two basic properties: entity integrity and referential integrity. Another is polyinstantiation, which we describe shortly.

Entity integrity states that no tuple can have null values for any primary key attribute. In the multilevel case, within a tuple all the primary key elements must have the same access class. Otherwise, a low user would see null values for some of the key elements. For example, if the first two data attributes form the primary key, the tuple (10, S, X, TS, 17, TS) has primary key elements with different access classes. A SECRET user's view of the tuple is (10, S, null, S, null, S), which violates entity integrity because part of the key appears to be null.

The key class must also be at least as low as the access classes of all other elements in the tuple; otherwise a low user would see nulls for the primary key. For example, if the first two data attributes form the primary key, the tuple (20, TS, Y, TS, 34, S) has a key class greater than the class of the remaining data element. A SECRET user's view of the tuple is (null, S, null, S, 34, S) which violates entity integrity because the key appears to be null.

Referential integrity states that no tuple in a relation can refer to another tuple unless the referenced tuple exists. In a multilevel database, this means that a tuple of a low access class cannot reference a tuple of a high access class because the referenced tuple would appear to be nonexistent to users with low clearances.

Multilevel security has a further unexpected but unavoidable effect, which we call *polyinstantiation* [29]. To illustrate, consider what happens when a user inserts a tuple that has the same primary key value as an existing but invisible (because it has a high classification) tuple. Notifying the user of the conflict would tell the user the value of high information. Thus we add a *second*

tuple to the relation with the same primary key but different access class; we call these *polyinstantiated tuples.* For example, if a SECRET user adds a tuple with primary key "ark" to the relation instance shown in Figure 2, then the outcome, as seen by a TOP-SECRET user, is as shown in Figure 3.

$A1$	$C1$	$A2$	$C2$	$A3$	$C3$
mad	S	17	S	x	S
foo	S	34	S	w	TS
ark	TS	5	TS	y	TS
ark	S	22	S	z	S

Figure 3: A Polyinstantiated Tuple

Now consider what happens if a user updates what appears to be a null element in a tuple, but which actually hides data with a higher access class. For example, if a SECRET user now replaces the perceived null value for tuple "foo," attribute $A3$ (see Figure 2) with the value "u," the outcome, as seen by a TOP-SECRET user, is as shown in Figure 4. We call these *polyinstantiated elements.*

$A1$	$C1$	$A2$	$C2$	$A3$	$C3$
mad	S	17	S	x	S
foo	S	34	S	w	TS
foo	S	34	S	u	S
ark	TS	5	TS	y	TS

Figure 4: A Polyinstantiated Element

To implement mandatory security, we assign security levels to processes, derived from the clearance of the user. Traditional practice is to segregate those functions enforcing mandatory security in a *security kernel* or *reference monitor.* The reference monitor mediates each reference to an object by any process, allowing or denying the access based on a label comparison. The reference monitor must be tamperproof; it must be invoked for *every* reference; and it must be small enough to be subject to complete analysis and test. When assurance is important, the reference monitor is formally verified through a formal mathematical proof (that may be carried out using automated tools) that it correctly enforces the mandatory security policy. The DoD has developed evaluation criteria for trusted computer systems [6] that incorporate the concept of reference monitor and include requirements for assurance as well as many other security requirements. The most stringent of the evaluation classes is called Class A1.

There have been three efforts to design Class A1 relational database systems.

- SeaView, being developed at SRI by T. Lunt et al., provides element-level labeling and derives labels for derived data. It includes a multilevel query language called MSQL for defining and manipulating multilevel data. Its design has been partially verified using EHDM, a formal verification system developed at SRI [1, 2, 3, 5, 39, 40, 41, 42]. SeaView uses a conventional relational engine and a commercially available reference monitor [29, 31, 32].

- A group at Secure Computing Technology Corporation (SCTC) has produced a design for LOCK Data Views (LDV), a relational system that allows an application to specify rules for how incoming and outgoing data are to be labeled. LDV relies on special-purpose security hardware being developed at SCTC [38].
- ASD is a prototype developed at TRW. It provides row-level labeling [19].

In addition, several vendors, including Oracle Corporation and Sybase, have announced or released products designed to meet some of the DoD criteria.

Ongoing research is focused on the following areas:

- *Defining an operational semantics for multilevel database operations* [30]. There is a need to define the intended semantics for the basic data manipulation operations insert, update, and delete for a multilevel database system with classification at the element level. This will effect, for example, how much polyinstantiation can occur as a result of an update operation, and which polyinstantiated instances are deleted in a delete operation.
- *Extending multilevel security to other data models* [12, 20, 21, 28, 33]. Preliminary work has begun to develop multilevel security models for object-oriented databases, entity-relationship databases, and knowledge-based systems.
- *Extending multilevel security to distributed database systems* [8]. Researchers are beginning to address such issues as architectures for secure distributed database systems and balancing the security, consistency, and availability of distributed data.
- *Solving inference problems* [27]. The inference problem arises when a collection of data is more sensitive than the individual pieces (this is sometimes called the *aggregation problem*, or when a collection of data can be used to partially infer data of a higher sensitivity. In some cases, even learning of the existence of the information may be unacceptable. This problem commonly arises when the individual data items may be classified low, but the relationship among the data items is considered to be highly sensitive. Data design strategies have been proposed to segregate the sensitive associations and to give those sensitive associations high classifications while giving the individual data items low classifications. This has the result that low data is available to users with low clearances while users with high clearances can make sensitive associations among the data.

References

[1] EHDM *Specification and Verification System Version 4.1—User's Guide.* Computer Science Laboratory, SRI International, Menlo Park, CA 94025, November 1988. See [3] for the updates to Version 5.1.

[2] EHDM *Specification and Verification System Version 5.0—Description of the* EHDM *Specification Language.* Computer Science Laboratory, SRI International, Menlo Park, CA 94025, January 1990. See [3] for the updates to Version 5.1.

[3] EHDM *Specification and Verification System—Version 5.1 Supplement to User's and Language Manuals.* Computer Science Laboratory, SRI International, Menlo Park, CA 94025, April 1990.

[4] R. W. Conway, W. L. Maxwell, and H. L. Morgan. On the implementation of security measures in information systems. *Communications of the ACM*, 15(4), April 1972.

[5] J. S. Crow, R. Lee, J. M. Rushby, F. W. von Henke, and R. A. Whitehurst. EHDM verification environment: An overview. In *Proceedings of the 11th National Computer Security Conference*, October 1988.

[6] *Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD.* Department of Defense, December 1985.

[7] K. R. Dittrich, M. Hartig, and H. Pfefferle. Discretionary access control in structurally object-oriented database systems. In *Proceedings of the 2nd IFIP WG11.3 Workshop on Database Security*, October 1988.

[8] A. Downing, I. Greenberg, and T. F. Lunt. Issues in distributed database security. In *Proceedings of the 5th Aerospace Computer Security Conference*, December 1989.

[9] D. D. Downs, J. R. Rub, K. C. Kung, and C. S. Jordan. Issues in discretionary access control. In *Proceedings of the 1985 IEEE Symposium on Security and Privacy*, 1985.

[10] E. B. Fernandez, R. C. Summers, and C. Wood. *Database Security and Integrity.* Addison-Wesley, Reading, Massachusetts, 1981.

[11] E. B. Fernandez, R. C. Summers, and C. D. Coleman. An authorization model for a shared data base. In *Proceedings of the 1975 ACM SIGMOD International Conference*, 1975.

[12] T. D. Garvey and T. F. Lunt. Multilevel security for knowledge-based systems. In *Proceedings the EISS Workshop on Database Security*, European Institute for System Security, Karlsruhe, W. Germany, April 1990.

[13] G. S. Graham and P. J. Denning. Protection—principles and practice. In *Proceedings of the Spring Joint Computer Conference*, volume 40, Montvale, New Jersey, 1972. AFIPS Press.

[14] P. P. Griffiths and B. W. Wade. An authorization mechanism for a relational database system. *ACM Transactions on Database Systems*, 1(3), September 1976.

[15] E. Gudes, H. Song, and E. B. Fernandez. Evaluation of negative and predicate-based authorization in object-oriented databases. *Proceedings of the 4th IFIP WG11.3 Workshop on Database Security*, Halifax, UK, Sept. 1990.

[16] M. A. Harrison. Theoretical issues concerning protection in operating systems. *Advances in Computers*, 24, 1985.

[17] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8), August 1976.

[18] H. R. Hartson and D. K. Hsiao. A semantic model for database protection languages. In *Proceedings of the Second International Conference on VLDB.* North-Holland, 1976.

[19] T. H. Hinke, C. Garvey, N. Jensen, J. Wilson, and A. Wu. A1 secure DBMS design. In *Proceedings of the 11th National Computer Security Conference - Appendix*, October 1988.

[20] S. Jajodia and B. Kogan. Integrating an object-oriented data model with multilevel security. In *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, May 1990.

[21] T. F. Keefe, W. T. Tsai, and M. B. Thuraisingham. SODA: A secure object-oriented database system. Technical report, TR89-12, University of Minnesota, Computer Science Department, 1989.

[22] U. Kelter. Group-oriented discretionary access controls for distributed structurally object-oriented database systems. Informatics Report N 93, Fern Universitat Hagen, Hagen, Germany, 1990.

[23] B. W. Lampson. Protection. In *Proceedings of the 5th Princeton Symposium on Information Science and Systems*, March 1971. Reprinted in ACM Operating Systems Review, Vol. 8 (1), January 1974.

[24] M. M. Larrondo-Petrie, E. Gudes, H. Song, and E. B. Fernandez. Security policies in object-oriented databases. In *Database Security III: Status and Prospects, D.L. Spooner and C. Landwehr (Eds.).* Elsevier, 1990.

[25] T. F. Lunt. Access control policies for database systems. In C. E. Landwehr, editor, *Database Security II: Status and Prospects.* North Holland, 1989.

[26] T. F. Lunt. Access control policies: Some unanswered questions. *Computers and Security*, February 1989.

[27] T. F. Lunt. Aggregation and inference: Facts and fallacies. In *Proceedings of the 1989 IEEE Symposium on Research in Security and Privacy*, May 1989.

[28] T. F. Lunt. Multilevel security for object-oriented database systems. In D. L. Spooner and C. E. Landwehr, editors, *Database Security III: Status and Prospects.* Elsevier, 1990.

[29] T. F. Lunt, D. E. Denning, R. R. Schell, W. R. Shockley, and M. Heckman. The SeaView security model. *IEEE Transactions on Software Engineering*, June 1990.

[30] T. F. Lunt and D. Hsieh. Update semantics for a multilevel relational database system. In *Proceedings of the 4th IFIP WG11.3 Workshop on Database Security*, Halifax, England, September 1990.

[31] T. F. Lunt, R. R. Schell, W. R. Shockley, M. Heckman, and D. Warren. A near-term design for the SeaView multilevel database system. In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, April 1988.

[32] T. F. Lunt, R. R. Schell, W. R. Shockley, M. Heckman, and D. Warren. Toward a multilevel relational data language. In *Proceedings of the Fourth Aerospace Computer Security Applications Conference*, December 1988.

[33] J. K. Millen and T. F. Lunt. Secure knowledge-based systems. Technical Report SRI-CSL-90-04, Computer Science Laboratory, SRI International, Menlo Park, California, August 1989.

[34] J. D. Moffet and M. S. Sloman. The source of authority for commercial access control. *Computer*, 21(2), February 1988.

[35] J. D. Moffet and M. S. Sloman. Delegation of authority. Domino Rept. B1/IC/4, Dept. of Computing, Imperial College of Science and Technology, London, July 1990.

[36] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A model of authorization for next generation database systems. Technical Report ACA-ST-395-88, MCC, November 1988.

[37] H. Song, E. B. Fernandez, and E. Gudes. Administrative authorization in object-oriented databases. In *Proceedings of the EISS Workshop on Database Security*, European Institute for System Security, Karlsruhe, W. Germany, April 1990.

[38] P. D. Stachour and B. Thuraisingham. Design of ldv: A multilevel secure relational database management system. *IEEE Transactions on Knowledge and Data Engineering*, 2:2, June 1990.

[39] F. von Henke and J. Rushby. *Introduction to* EHDM. Computer Science Laboratory, SRI International, Menlo Park, CA 94025, September 28, 1988.

[40] F. von Henke, N. Shankar, and J. Rushby. *Formal Semantics of* EHDM. Computer Science Laboratory, SRI International, Menlo Park, CA 94025, January 1990. This document describes EHDM Version 5.0, see [3] for informal descriptions of the changes in Version 5.1.

[41] R. A. Whitehurst and T. F. Lunt. The SeaView verification. In *Proceedings of the Second Workshop on the Foundations of Computer Security*, June 1989.

[42] R. A. Whitehurst and T. F. Lunt. The SeaView verification effort. In *Proceedings of the 12th National Computer Security Conference*, October 1989.