Proc. of the IFIP WG11.3 Workshop on Database Security,
Monterey, California, September 5-7, 1989

MANDATORY CONTROLS FOR DATABASE INTEGRITY

Ravi Sandhu

Department of Information Systems and Systems Engineering
George Mason University
Fairfax, Virginia 22030-4444, USA

1 INTRODUCTION

Our goal is to develop a scientific understanding of the kinds of mandatory controls needed to support data integrity. There is general consensus that integrity is an important problem and requires some kind of mandatory controls for its solution. At the same time, inspite of considerable effort [17, 18, for instance], there is little consensus on what is meant by the two key terms: mandatory controls and data integrity.

In this paper we outline our position on these issues and suggest avenues of research. Our fundamental claims are as follows.

1. Mandatory controls for data integrity cannot be reduced to traditional lattice-based controls [1, 2, 7] on read and write operations alone.

2. Data integrity requires a more sophisticated view of mandatory controls than provided by the traditional black and white distinction between mandatory and discretionary controls.

These claims find support to various degrees in the literature [3, 4, 14, 15, 19, 20, 24, for instance]. Our objective is to consider their ramifications for database management systems. We actually believe that both claims also apply to data and information secrecy, particularly the second one. However for the moment we choose to focus on data integrity.

The paper relates most closely to the following research questions identified by IFIP WG 11.3 as being of current significance.

Q1. What is a good framework for ... stating database security policy requirements (including application-dependent requirements ...)? ...

Q2. How can security considerations be integrated in software development methodologies used in database system development?

Q4. How can database audit information be generated and reviewed most effectively to detect and discourage security violations?

2 BACKGROUND

Agreement on the meaning of integrity remains elusive. A recent workshop organized by NIST [18] was unable to make much progress on this issue. For our purpose here we simply accept the common viewpoint that integrity is concerned with information modification rather than information disclosure or availability. That is integrity is something different from confidentiality or denial of service.

Recent interest in data integrity has been stimulated in large part by the seminal paper of Clark and Wilson [4] who set out to establish that:

> "First, there is a distinct set of security policies, related to integrity rather than disclosure ... Second, some separate mechanisms are required for enforcement of these policies, disjoint from those of the Orange Book [8]."

Contrast this with what might be called the traditional view, exemplified here by the following quote from Gasser [9].

> "Fortunately, techniques to protect against information modification are almost always the same as (or a subset of) techniques to protect against information disclosure."

The techniques that Gasser talks about can be roughly summarized as follows: impose application independent label-based mandatory controls with respect to read and write operations (with append and execute sometimes included). Our objections to this traditional view are given below.

1. The published examples of the use of such techniques for integrity [2, 10, 12, 13] inevitably require trusted subjects who bypass the mandatory controls. In many cases a significant fraction of all subjects need to be trusted. One wonders what is the point of imposing mandatory controls that are mostly bypassed.

2. Because reads and writes are the only point of control, the authors are obliged to make the audit trail writable by everybody and therefore of low integrity. This is disturbing since the whole point of an audit trail is to have high integrity.

3. The combination of independent integrity labels and confidentiality labels does not provide any additional power than obtained by each in isolation, i.e., precisely the same controls can be enforced using integrity labels alone or confidentiality labels alone. So the Biba view of integrity [2] is essentially equivalent to lattice based confidentiality [1, 7].

Clark and Wilson [4] have suggested that we look for different kinds of mandatory controls to directly support the "two mechanisms at the heart of fraud and error control: the well-formed transaction and segregation of duties." It is clear that, at the very least, current label-based controls need to be extended in non-trivial ways to achieve this effect.

3 MANDATORY CONTROLS

So we need a more general notion of mandatory controls which is not tied to lattice-based labels. Consider the following quote from the first WIPCIS workshop [16]: "... two types of mandatory controls are considered here—label-based mandatory controls (enforcing separation based on hierarchical or lattice oriented labels, as in the Orange Book) and general mandatory (which lies between label-based mandatory and discretionary controls)." We are troubled by this characterization of general mandatory as lying between label-based mandatory and discretionary controls. On the contrary we propose that label-based mandatory controls should be a special case of whatever we call general mandatory.

A reasonable working definition is given by Clark and Wilson [5] as follows: "... the word "mandatory." In the paper, we want to use it in the more general way, to describe any mechanism which is not put into place at the control of the owner of the data, but which is a necessary part of the operation of the system." However there are situations where mandatory controls are defined by the owner. For example the owner of checks is responsible for defining the well-formed transactions which can operate on checks as well as for defining the separation of duty requirements for processing checks. Once these decisions are made the resulting controls are mandatory for all other users. Clark and Wilson also have another working definition [6] as follows: "In this case we define mandatory as those controls which are unavoidably imposed by the operating system between user and data." This is very broad and can be interpreted to include discretionary controls.

We propose to define mandatory controls as controls based on properties of the object and/or the subject. This is as broad and open ended as the Clark and Wilson definitions above. However it does suggest that one can categorize mandatory controls in terms of the properties on which the controls are based. In the military non-disclosure context these properties turn out to be best expressed as partially ordered labels. In other contexts these properties are more naturally obtained in other ways. For instance the type of an object determines what operations can be executed on that object. Subjects are divided into two classes for this purpose: the type manager who can execute arbitrary operations and all others who can only execute operations exported by the type manager.

Label-based controls are obviously a special case of our definition. Identity based discretionary controls are also a special case. We can explicitly exclude discretionary controls by refining the definition of mandatory controls to be "controls based on properties of the object and/or the subject (excluding identity)." On the other hand it is equally reasonable to consider discretionary controls as a special case of mandatory controls. We believe the traditional black and white distinction between discretionary and mandatory controls is inappropriate in many contexts. All authority in a system is ultimately obtained by means of somebody's discretionary decisions [14, 15]. The real difference is to what extent discretionary ability can be granted and acquired during the normal operation of a system, and to what extent it gets fixed at system initialization.

Our proposal allows us to categorize mandatory controls along the following progression of properties on which the controls are based.

1. Controls based on *identity*. These include what are traditionally called discretionary controls. We are assuming here that the identity of a subject or object is an immutable property which can never change.

2. Controls based on *static properties* of the object and subject. These static properties are determined at creation and do not change thereafter. Label-based controls of the Bell and LaPadula model [1] with strong tranquillity (i.e., labels are static) are a well-known example. Various kinds of type based controls provide more general examples [3, 19, 21, 23, 24].

3. Controls based on *dynamic properties* of the object and subject. That is the properties on which the controls are based are themselves changeable, presumably in some controlled manner requiring proper authorization. Controls based on the history of an object and the role of a subject, such as enforced by transaction control expressions [20], are one example. Another example is label-based controls without tranquillity (i.e., labels can be modified).

If we assume identity is immutable, 1 is a special case of 2. Similarly, 2 is a special case of 3 if dynamic is interpreted to include static. So there is a logical progression. We can make a finer distinction regarding controls based on dynamic properties as follows.*

3a. Controls based on dynamic properties *excluding data values*.

3b. Controls based on dynamic properties *including data values*.

Group membership does not figure in the above categorization. This is deliberate. If group membership is a static attribute we could include it in 1 or 2 above. On the other hand consider the dynamic policy that (1) a project group must have a majority of members from within the department, and (2) only the project supervisor can enroll outsiders. How does this policy relate to our characterization of mandatory controls? The first requirement falls under 3b in that the project supervisor's discretion is constrained by a value dependent rule. Classification of the second requirement depends on further details of the policy. If the assignment of project supervisors is static then this is an example of 2. On the other hand if the designation of a project supervisor is itself dynamic, say at the discretion of the department head, this would be an example of 3a.

The point is that even the simplest situations exhibit interaction between different aspects of the overall policy. Which controls are discretionary and which mandatory is a relative issue, depending on whose perspective is considered. A black and white separation between discretionary and mandatory controls is therefore too simplistic. With multilayered interacting discretionary and mandatory controls safety becomes a major issue. That is we need to be able to predict what privileges can be acquired by particular subjects given various assumptions about cooperation among themselves and other subjects.

*A similar distinction might be made regarding controls based on static properties. However controls based on static data values (i.e., constants) appear to confer little, if any, additional power over controls based on static value independent properties.

4 TRANSACTION CONTROLS

We now briefly consider some mechanisms for enforcing the progression of mandatory controls identified in the previous section. We take as our basis the standard concept of a transaction from the database literature. We understand a transaction to be a serializable and failure atomic unit of activity.

At the very least a DBMS should be able to enforce the restriction that only transactions can update the database. This gives the minimum basis for avoiding inconsistencies due to concurrency control and failures. It is also clear that the ability to create new transactions must be carefully regulated with a strong separation between application programmers and production users. Such controls are relatively easy to implement by static controls because it can be assumed that application programmers and production users are disjoint.

We agree with Clark and Wilson [4] that production users must be further constrained in the transactions they can execute and the data to which these transactions are applied. Clark and Wilson suggest two relations for stating these controls, but the net effect is equally well specified as an access-control triple of the form <user, data-object, transaction>. Clark and Wilson seem to imply these relations or triples be explicitly maintained. We would rather state these controls indirectly as <user-role, data-object, transaction>. The power of a user is typically derived from his role (position) in an organization rather than being a function of his individual self. If a user's role changes so does his authority. Since such changes are inevitable it is appropriate to explicitly relate authority and responsibility to positions rather than to the individuals occupying them at a given moment.

Furthermore, a transaction is too elementary a unit for integrity purposes. For example, consider a situation in which payment in the form of a check is prepared and issued by the following sequence of events.

1. A clerk prepares the voucher.
2. The voucher is approved by a supervisor.
3. The check is issued by a clerk, who must be different from the clerk in step 1.

From a concurrency control and recovery perspective this sequence is best viewed as three separate transactions, one for each step. The activities they represent are separated in time by unpredictable and possibly large intervals. For instance, the first two steps may be performed on line while the third is executed in batch. Moreover different users have responsibility and authorization for these activities. On the other hand, from an integrity perspective we must view this sequence as a single unit. The third step, in particular, makes explicit the constraint that the clerk executing it be different from the clerk in the first step. So it will take collusion of two clerks and a supervisor to perpetrate fraud. Since the check is presumably issued against some account the above sequence is more properly expressed as follows.

1. A clerk prepares the voucher and assigns an account.
2. The voucher and account are approved by a supervisor.

3. The check is issued by a clerk who must be different from the clerk in step 1. Issuing the check has the side effect of debiting the account assigned in step 1.

That is the voucher contains a reference to an account, which is another information object in the system. Strictly speaking for double entry accounting the reference should be to two accounts, one to be debited and the other credited in step 3. The important point for us is that transactions executed on the voucher have side effects on objects such as accounts.

A voucher and an account are two rather different kinds of objects. The voucher is a *transient object* which comes into existence, has a finite sequence of operations applied to it and then disappears (possibly leaving a record in some archive). The account on the other hand is a *persistent object* with a long life in the system with a potentially unbounded sequence of credit and debit operations performed on it. Of course, at some point the account may be closed. The key point is that we cannot prescribe its history as a finite sequence of actions. Both kinds of objects are essential to the logic and correct operation of an information system. Transient objects embody a logically complete history of transactions corresponding to a unit of service provided to the external world. Persistent objects embody the internal records required to keep the organization functioning with accurate correspondence to its interactions with the external world.

Our fundamental thesis is that integrity can be achieved by enforcing controls on transient objects, for the most part. The crucial idea, which makes this possible, is that transactions should be executed on persistent objects only as a side effect of executing transactions on transient objects. The details of our proposal are discussed in [21]. The basic idea is to associate a *transaction control expression* with each object. The expression specifies what transactions can be executed by what user (in terms of roles) on that object. The expression is incrementally converted to a history (audit trail) as the execution proceeds. This mechanism can enforce the controls required for the above example as well as the additional requirement that a supervisor should not approve checks for accounts created by himself. The history maintained with each object gives us the necessary component of the audit trail to enforce sequencing of transactions and separation of duties with respect to that object. Moreover the mechanism can be used in an enforcement or detection mode. Finally it falls under 3a of our progression of mandatory controls. In a DBMS context it can be easily extended to cover 3b also by allowing the transaction control expression to be conditioned on data values.

We mention that transaction control expressions were influenced by Karger's work [11] but they go well beyond his proposal. As stated by Karger himself for his proposal "... the user interface for specifying separation of duties appears extremely complex." On the other hand our proposal gives a natural notation and our transient objects correspond to paper forms with initialed entries as the fundamental basis for control. Moreover our viewpoint makes it clear that data integrity is not a property of the data in isolation but is intimately tied to both data and transactions.

5 CONCLUSION

We have presented our position that mandatory controls for data integrity cannot be reduced to lattice-based controls on read and write operations alone. Also that data integrity requires a more sophisticated view of mandatory controls than provided by the traditional black and white distinction between mandatory and discretionary controls. We have proposed mandatory controls be defined to mean controls based on properties of the object and/or the subject. We have given a characterization of mandatory controls based on the nature of properties on which they depend. Finally we have considered mechanisms to support the progression of mandatory controls thus identified.

References

[1] Bell, D.E. and LaPadula, L.J. "Secure Computer Systems: Unified Exposition and Multics Interpretation." MTR-2997, Mitre, Bedford, Mass. (1975).

[2] Biba, K.J. "Integrity Considerations for Secure Computer Systems." Mitre TR-3153, Mitre Corporation, Bedford, Mass., April 1977.

[3] Boebert, W.E. and Kain, R.Y. "A Practical Alternative to Hierarchical Integrity Policies." *8th National Computer Security Conference*, 18-27 (1985).

[4] Clark, D.D. and Wilson, D.R. "A Comparison of Commercial and Military Computer Security Policies." *IEEE Symposium on Security and Privacy*, 184-194 (1987).

[5] Clark, D.D. and Wilson, D.R. "Comments on the Integrity Model." In [17].

[6] Clark, D.D. and Wilson, D.R. "Evolution of a Model for Computer Integrity." In [18].

[7] Denning, D.E. "A Lattice Model of Secure Information Flow." *Communications of ACM* 19(5):236-243 (1976).

[8] Department of Defense National Computer Security Center. *Department of Defense Trusted Computer Systems Evaluation Criteria.* DoD 5200.28-STD, (1985).

[9] Gasser, M. *Building a Secure Computer System.* Van Nostrand Reinhold (1988).

[10] Jueneman, R.R. "Integrity Controls for Military and Commercial Applications." *Fourth Aerospace Computer Security Applications Conference*, 298-322 (1988).

[11] Karger, P.A. "Implementing Commercial Data Integrity with Secure Capabilities." *IEEE Symposium on Security and Privacy*, 130-139 (1988).

[12] Lee, T.M.P. "Using Mandatory Integrity to Enforce "Commercial" Integrity." *IEEE Symposium on Security and Privacy*, 140-146 (1988).

[13] Lipner, S.B. "Non-Discretionary Controls for Commercial Applications." *IEEE Symposium on Security and Privacy*, 2-10 (1982).

[14] Moffett, J.D. and Sloman, M.S. "The Source of Authority for Commercial Access Control." *IEEE Computer* 21(2):59-69 (1988).

[15] Murray, W. H. "On the Use of Mandatory." Position paper in [17].

[16] Parker, D.B. and Neumann, P.G. "A Summary and Interpretation of the Invitational Workshop on Integrity Policy in Computer Information Systems." In [17].

[17] *Report of the Invitational Workshop on Integrity Policy in Computer Information Systems.* NIST Special Publication 500-160 (1989).

[18] *Report of the Invitational Workshop on Data Integrity.* NIST Special Publication 500-168 (1989).

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

[20] Sandhu, R.S. "Expressive Power of the Schematic Protection Model." *Computer Security Foundations Workshop*, 188-193 (1988).

[21] Sandhu, R.S. "Transaction Control Expressions for Separation of Duties." *Fourth Aerospace Computer Security Applications Conference*, 282-286 (1988).

[22] Sandhu, R.S. "Terminology, Criteria and System Architectures for Data Integrity." In [18].

[23] Sandhu, R.S. "Transformation of Access Rights." *IEEE Symposium on Security and Privacy*, 259-268 (1989).

[24] Wong, R.M. and Ding, Y.E. "Providing Software Integrity Using Type Managers." *Fourth Aerospace Computer Security Applications Conference*, 287-294 (1988).