Specification of Content-Dependent Security Policies

David L. Spooner
Mathematical Sciences Department
Rensselaer Polytechnic Institute
Troy, NY 12181

Abstract: The protection of information from unauthorized disclosure is an important consideration for the designers of any large multiuser computer system. A general purpose database management system often requires the enforcement of content-dependent security policies in which a decision to allow access must be based on the value of the data itself. Several authors ([Har76], [Sto76], [Gri76], [Sum77], [Min78], [Spo83], and others) have proposed mechanisms for implementing content-dependent security policies. Few authors, however, have investigated the properties of models for the specification of such policies.

This paper identifies several problems created by inadequate models for the specification of content-dependent security policies. If a specification model is too liberal in the types of policies it can express, it may provide an increased opportunity for compromise of data. If the specification model is too conservative, it cannot express many desirable policies. Thus a flexible model which will allow a compromise between these two extremes is needed for specifying content-dependent policies. Such a model is proposed here.

1. Introduction

The protection of information from unauthorized disclosure is an important consideration for the designers of any large multiuser computer system. The rapid increase in the numbers of new computer users and the increasing use of computer networks make the protection of information an even more critical issue. The problem is further complicated by the increasing complexity of computer systems.

The tried and true methods of protecting information using traditional operating system primitives (i.e. capabilities and access lists) are not adequate for enforcing data security in a general purpose database management system (DBMS). Such systems demand more sophisticated and flexible protection mechanisms. One of the most important differences is that a DBMS often requires the enforcement of content-dependent security policies in which a decision to allow access to data must be based on the value of the data itself.

The need for content-dependent security policies occurs frequently in DBMS applications. For example, consider a file of employee records for a large corporation. Each record contains the fields: NAME, JOB, SALARY, and MANAGER. A reasonable security requirement for this file is that a manager may only access records of employees he manages. Thus a particular manager may access a record only if he is listed in the MANAGER field of the record.

Content-dependent security policies are usually expressed using access rules with Boolean predicates describing subsets of the data which a user is authorized to access. Unlike other types of security policies, content-dependent policies must be enforced at run-time when the data is actually retrieved and can be used to make an access decision. Several authors ([Har76], [Sto76], [Gri76], [Sum77], [Min78], [Spo83], and others) have suggested mechanisms for implementing content-dependent security policies. Section 2 defines the concept of content-dependent security in more detail.

Little work has been focussed on the properties of models for the specification of content-dependent policies. In section 3, several problems created by inadequate models for the specification of such policies are discussed. If a specification model is too liberal in the types of policies it can express, it may provide an increased opportunity for compromise of data. If the specification model is too conservative, it cannot express many desirable policies. Thus a flexible model which will allow a

compromise between these two extremes is needed for specifying content-dependent security policies. Such a model is discussed in section 4.

2. Content-Dependent Security

To facilitate further discussion, a notation is needed for expressing access rules for users. The following notation will be used in this paper. It is assumed that all data is stored in the form of relations or tables. Suitable modifications can be made for handling other data structures. The basic access rights which a user may have to the columns of a relation include such things as read, and write access. The complete set of basic access rights is unimportant for this discussion.

An access rule for user u has the following form:

```
u:relation name[column list] (rights list)
          where  predicate
```

This rule states that user u has the basic rights listed in the rights list to the columns of the named relation listed in the column list whenever the specified predicate is true. If the access rule is to apply to all columns of a relation, the column list can be replaced by the word "all."

No restrictions are placed on the set of data items which may be used in a predicate. Range variables for each relation are used in the column list and predicate to avoid ambiguity. A range variable for a relation is a variable whose value can be any row of the relation. Thus individual data items in a relation are named as:

```
 relation range variable .  column name
```

The access rules have a syntax very similar to the syntax for queries in QUEL, the query language for the INGRES database system [Sto76].

As an example, consider the following relational database of employee information for a large corporation.

EMPLOYEE

NAME	JOBTITLE	JOBRANK	SALARY	MANAGER
.	.	.	.	.

DIRECTORY

NAME	DEPT	BUILDING	ROOM	PHONE
.	.	.	.	.

Suppose that a clerk in the payroll department is to be allowed access to the name, salary, and department of all programmers in departments 5 and 6. This content-dependent policy can be expressed as follows:

Example 1.

```
Range of e is EMPLOYEE
Range of d is DIRECTORY

clerk: EMPLOYEE[e.NAME, e.SALARY] (read)
       where e.JOBTITLE="PROGRAMMER"
             & e.NAME=d.NAME &
             (d.DEPT=5 | d.DEPT=6)

clerk: DIRECTORY[d.DEPT] (read)
       where e.JOBTITLE="PROGRAMMER"
             & e.NAME=d.NAME &
             (d.DEPT=5 | d.DEPT=6)
```

The first access rule says that the clerk has read access to the NAME and SALARY fields of any row in relation EMPLOYEE for which the JOBTITLE of that row has the value "PROGRAMMER" and for which there is a row in the DIRECTORY relation with the same value for the NAME field and the DEPT field equal to 5 or 6. The second access rule is similar. Note the use of the range variable e to specify that the values for fields NAME, SALARY, and JOBTITLE in the column list and the predicate must all come from the same row of relation EMPLOYEE. Range variable d serves a similar purpose for relation DIRECTORY.

3. Problems in the Specification of Content-Dependent Policies

Suppose that user u has an access rule for data item A and in the predicate of the rule is a reference to data item B. If user u has no access rules for data item B, or has only rules specifying content-dependent read access to data item B, user u may not be authorized to access data item B to evaluate the predicate for A. Evaluation of the predicate for A may therefore compromise the value of data item B.

The remainder of this section discusses potential solutions to this problem and the consequences each solution has on models for the specification of content-dependent security policies. There are two cases to consider: 1) user u has unconditional read access to data item B, and 2) user u has content-dependent read access or no read access to data item B.

3.1 Unconditional Read Access

If user u has an access rule giving

him unconditional read access to data item B, then the predicate for data item A can be evaluated without compromising the value of data item B. Suppose that a policy specification model requires that a user have unconditional access to all data items referenced in predicates of access rules. As is seen in the next example, this can limit the policies which can be expressed in the model.

Consider the EMPLOYEE relation defined in section 2, and suppose that salary is directly related to job rank. A security policy that one might wish to enforce is that a clerk in the personnel department may look at employee names and job titles for all employees of rank no greater than 10, say. The clerk should not be able to determine any individual salary. He is therefore not authorized to access the JOBRANK or SALARY columns of the relation. Such a policy can be expressed as follows:

Example 2.

```
Range of e is EMPLOYEE

clerk: EMPLOYEE[e.NAME, e.JOBTITLE] (read,
       write) where  e.JOBRANK<=10
```

If the requirement is made that a user must have unconditional read access to data items referenced in a predicate, then this policy specification is illegal since the clerk has no access to the JOBRANK column. This policy, and others like it, cannot be expressed in a policy specification model with such a restriction.

It may be argued that since the clerk has no access rules allowing him to manipulate the JOBRANK column, it doesn't matter if the security system consults the job rank of an employee in making an access decision for the clerk. This is a policy decision for the security officer of the system.

3.2 Content-Dependent Read Access or No Read Access

Consider now the situation where user u has an access rule for data item A with a content-dependent predicate referencing data item B. He also has an access rule giving him content-dependent read access to B. User u may or may not be authorized to read the value of data item B to evaluate the predicate of A.

Spooner, Gudes, and Fischer [Spo80] in their model recognized this problem and chose as a temporary solution to ignor the access rules for B when B is referenced in a predicate. A similar approach has been taken by others ([Sto76], [Gri76], [Har76], and others).

Summers and Fernandez [Sum77] chose the opposite solution. In their model the predicate in the access rule for B is "anded" to the predicate in the access rule for A. Access to A is allowed only if this new predicate evaluates to true. This approach seems inherently more secure, but can lead to several problems. If access to A depends on the content of B, access to B depends on the content of C, and so on, it may be necessary to evaluate a predicate of nearly unbounded length to authorize access to A. In addition, this approach reduces the range of policies which can be expressed in a specification model. Returning to Example 2, suppose that the access rules for the clerk are extended to allow him to read the JOBRANK column for employees with a job rank less than 5. This can be expressed with the following access rules.

Example 3.

```
Range of e is EMPLOYEE

clerk: EMPLOYEE[e.NAME, e.JOBTITLE] (read,
       write) where e.JOBRANK<=10

clerk: EMPLOYEE[e.JOBRANK] (read)
       where e.JOBRANK<5
```

Using the enforcement scheme of Summers and Fernandez, the clerk is able to access the name and job title only for employees with job rank less than 5. This is clearly not what was intended. Using this enforcement scheme, there is no way to express the desired policy.

Regardless of the method used for enforcement of such policies, there is a more serious problem. Suppose that a user has knowledge of the predicates defined in his access rules (a resonable assumption in many if not most situations). He may be able to infer information about the value of a data item referenced in the predicate of an access rule by attempting to use the rule and noticing whether or not he is denied access. In essense, the predicates in the access rules create many of the same problems encountered in protecting statistical databases [Den78]. The problem is made even more difficult by noticing that a user may receive access rules from several different people for different applications. These many different authorizers may be unaware of the potential compromise of data made possible by using the access rules together.

Consider now the final situation in which user u has an access rule for data item A with a predicate referencing data item B, and has no access rules for data item B. This case is similar to the situation where user u has content-dependent access to B, and all the

comments above apply. Only with the enforcement approach which ignors the access rules for B, will access to A be permitted.

4. Content-Dependent Specification Model

From the discussion above, it is clear that the choice of any model for the specification and enforcement of content-dependent security policies has the potential for introducing unwanted side-effects. These side-effects range from limiting expressibility of policies, to introducing the potential for compromise of data items. The side-effects exist in varying degrees in all models. As a result, the security officer for a system must be given the ability to choose between several models so that he may select the model which most closely satisfies his needs.

Suppose that a new basic access right for data items, say "pred," is introduced into a specification model. This new access right, when given to a user for a data item, allows the security system to access the data item on his behalf when evaluating a predicate in an access rule. It allows no other type of access. This new access right can be added to the model of Summers and Fernandez [Sum77]. The model is modified so that if user u has an access rule for data item A with a predicate referencing data item B, and user u also has the pred right to B, then the evaluation of the predicate for A takes place unchanged. If the user does not have the pred right for B but has another access rule giving him content-dependent access to B, then the predicate from the access rule for B is anded to the precicate in the access rule for A before it is evaluated as before. If the user has no access rule for B, access to A is denied. This model provides the security officer with the flexibility needed to selectively decide which users may access which data items for the purposes of predicate evaluation. It also increases the range of potential content-dependent security polcies which can be expressed using the model of Summers and Fernandez.

Consider again Example 2. An access rule can be defined for the clerk giving him the pred right for the JOBRANK column.

Example 4.

```
Range of e is EMPLOYEE

clerk: EMPLOYEE[e.NAME, e.JOBTITLE] (read,
         write) where e.JOBRANK<=10

clerk: EMPLOYEE[e.JOBRANK] (pred)
         where true
```

This allows the security system to access the JOBRANK column on behalf of the clerk when evaluating the predicate in the access rule for the NAME and JOBTITLE columns.

As a second example, reconsider Example 3. The security officer, at his own discretion, can define an access rule giving the clerk the pred right to the JOBRANK column. If this is done, the policy is enforced as intended.

Two problems must be addressed when the pred right is introduced into a specification model. The first problem concerns what to do when a user has an access rule giving him the pred right for a column, but with a content-dependent predicate. This can be handled like all other access rules by "anding" the predicate from the access rule for the pred right to any predicate referencing the column.

The second problem concerns who may grant the pred right to a user. Suppose in Example 4 that one person (X) is responsible for defining access rules for the NAME and JOBTITLE columns and that another person (Y) is responsible for defining access rules for the JOBRANK column. Should person X be able to grant the pred right for the JOBRANK column to the clerk? It may be desirable to allow this since the access rule for JOBRANK is defined only to ensure that the access rule for NAME and JOBTITLE is enforced correctly. One solution is to allow Y to grant to X the ability to define access rules containing the pred right for the JOBRANK column.

Introduction of the pred right solves several problems in the specification of content-dependent security policies, but it does nothing to eliminate the potential compromise of data items referenced in predicates of access rules. No simple solution to this problem is evident. In fact, a complete solution is impossible in the case where users know the predicates in their access rules or can discover them. A content-dependent access rule partitions the occurrences of a column into two sets, those which can be accessed and those which cannot. The user can always infer some information from noting which occurrences can and cannot be accessed. More work is required to find ways to minimize the problem. Clearly the work in protecting statistical databases is relevant [Den78].

Under general operating conditions, a user may receive access rules from several sources for different applications. These rules can be used together in unpredictable ways to compromise data in the system. Others ([Har76]) have identified this problem and noted that the predicates from the access rules can be "anded" or "ored" together. Either solution can lead to access policies being enforced differently from what is

intended. Restrictions can be placed on the ways access rules for data items can be used together, perhaps by identifying those access rules belonging to a particular application. This idea has been investigated by Summers and Fernandez [Sum77]. More research is needed to achieve a general yet effective implementation of this idea.

References

[Den78] Denning, D. E. "A Review of Research on Statistical Database Security." Foundations of Secure Computation, R.A. DeMillo, et. al. Eds, Academic Press, New York, 1978.

[Gri76] Griffiths, P. and B. Wade. "An Authorization Mechanism for a Relational Database System." ACM Transactions on Data Base Systems, 1 (1976), 3 (Sept), pp 242-255.

[Har76] Hartson, H. R. and D. K. Hsiao. "A Semantic Model for Date Base Protection Languages." Systems for Large Data Bases, North-Holland Publishing Company, pp. 27-42, 1976.

[Min78] Minski, N. "An Operation-Control Scheme for Authorization in Computer Systems." International Journal of Computer and Information Sciences, 7 (1978), 2 (June), pp. 157-191.

[Spo80] Spooner, D. L., E. Gudes, and P. C. Fischer. "The Logical Object Model for Data Base Operating Systems." Proc. Computer Software and Applications Conference, IEEE, pp.769-775, 1980.

[Spo83] Spooner, D. "The Design of a Unified Security Model for a Database Operating System," Proc. Sixteenth Hawaii International Conference on Systems Sciences, pp. 75-82, 1983.

[Sto76] Stonebraker, M., and P. Rubinstein. "The INGRES Protection System." Proc. ACM Conference, Houston, pp. 80-84, 1976.

[Sum77] Summers, R. C. and E. B. Fernandez. "A System Structure for Data Security." IBM Los Angeles Scientific Center Report, G320-2689, April, 1977.