TR-82-003

A Note on Protection of Context-Sensitive Information

By

Yang-Chang Hong
Institute of Information Science
Academia Sinica 115
Taipei, Taiwan, R. O. C.

and

Stanley, Y. W. Su
Department of Electrical Engineering
University of Florida
Gainesville, Fl. 32611
U. S. A.

Jan. 29, 1982

ABSTRACT : This paper investigates the problem of invoking a context-dependent decision for security checking. The context problem which is the ability of the security subject to combine partial information to which he has the right to access, in order to produce information he is not entitled to see, is discussed. A protection scheme for disallowing the subject to deduce context-sensitive information based on data semantic dependencies is presented.

Key Words and Phrases : context-sensitive information, semantic dependency, security decision, access path, deduction, protection

1. Introduction

In order to control access to a database, a set of security decisions is generally needed by a DBMS system. These decisions are rules to specify conditions under which a particular subject can be granted access to a particular object (e.g., an attribute or a relation in the relational database context) [4], or define the portion of a database to which a particular subject is entitled to see [2]. They have been classified into different types [2,4,7]. One type of decision is referred to as the context-dependent (CXD) decision [4,5,8]. It does not allow a subject to relate one object to another, but permitting him to see the individuls. An example of a CXD decision could be as follows:

d_1 : The subject S can see the salaries of employees, but not together with their names and vice versa.

This type of decision is context-dependent in the sense that its invocation is dependent on the context of database access.

For a decision which is context-independent, it can be invoked for enforcement when a query (or an application program) accesses or manipulates a set of objects which has the decision [6,10]. However, this cannot be true for a CXD decision, in general. This is because objects in a database are close related semantically and structually. Partial information to which a subject has the right to access may be combined to disclose context-sensitive information. Consider a relation EMP(ENAME, SALARY, STATUS, MGR, AGE, ASSESSMENT) with primary key ENAME. Assume that attribute STATUS is functionally dependent on ENAME, denoted as {ENAME} → {STATUS}, and {STATUS} → {SALARY} [1]. (The functional

dependency: $X \rightarrow Y$ holds in a relation R if any two tuples u and v of the relation R $u[X] = v[X]$ implies $u[Y] = v[Y]$, where X, Y are sets of attributes of R and $u[X]$ is the projection of the tuple u on X.) Consider the following two requests: one is to request the names and status of employees whose manager is 'SMITH' and the other is to request the salaries with STATUS $\leq$ 5. With the invocation mechanism for context-independent decisions, neither will necessarily invoke the decision d_1. The decision d_1 is invoked if a request involves the access of a set of attributes which contains{ENAME, SALARY}. (The first request involves the set {ENAME, STATUS, MGR} and the second request the set {SALARY, STATUS}.) However, one can obtain the names and the associated salaries for those imployees whose manager is 'SMITH' and status is less than or equal to 5 via status, since a transitivity property holds for any two functional dependencies. This example points out that the invocation of a CXD decision depends on not only the objects being interrogated but also the records previously accessed —— i.e., access history information [5,7,8].

For a DBMS to enforce this type of decision, it is necessary to keep all the relevant records previously accessed for subsequent security checking. Since the subject has the right to see individual objects being protected simultaneously, one cannot reasonably hope to design a mechanism which can completely prevent him from deducing any context-sensitive information. However, deduction based on data semantic dependencies can be understood and has to be prevented against. This paper is to provide an algorithm for enumerating all the paths from which one can obtain supposedly protected information. Protection of context-sensitive information is thus reduced to preventing the corresponding subject from forming any of these paths. In the paper, three ways of enforcing schemes are suggested and their inplementation is presented.

2. Definition of CXD Decisions

As defined above, a CXD decision is the one which disallows a subject to relate one object to another, while permitting him to access individuals. With a view toward enforcement, the definition is not so rigorous. Consider a CXD decision d_2 in which the salaries of employees can be read, but not together with the managers; however, it would not be very meaningful if attributes SALARY and MGR in the EMP relation are mutually independent. Consider another CXD decision d_3 that does not allow the subject S to relate ENAME, SALARY, and ASSESSMENT. This decision should imply that the subject S is not allowed to relate neither ENAME and SALARY nor ENAME and ASSESSMENT, if $\{ENAME\} \rightarrow \{SALARY\}$ and $\{ENAME\} \rightarrow \{ASSESSMENT\}$. Otherwise whenever S sees the employee names and their salaries or their assessment, he has obtained partial information that should be protected. Furthermore, if S accesses the names and the associated salaries in one request and the names and the associated assessment in another, he can obtain supposedly protected information via the employee names (i.e., primary keys).

From the discussion above, it is useful to define a CXD decision more formally. The following definition is based on the functional dependency between sets of attributes of relations. The attributes are the security objects in the relational context.

Definition: A decision d is said to be the context-dependent decision if it does not allow the subject S to produce a relation consisting of a set of attributes (or objects) R_d with the following properties:

(i) There exists one single element set $Y \subset R_d$ such that $X(=R_d - Y) \rightarrow Y$ and

(ii) There is no proper subset X' of X such that $X' \rightarrow Y$. That is, Y is "full" functionally dependent on X.

Let us denote such a decision as $d_{X \cup Y}$. Here, Y is restricted to a single element set because of the fact $X \rightarrow \{A_1, A_2, \ldots, A_k\}$ implies $X \rightarrow \{A_i\}$, $1 \leq i \leq k$. Based on the definition, decision d_1 which disallow S to produce a relation with R_d = {ENAME, SALARY} is a CXD decision since {ENAME} → {STATUS} and {STATUS} → {SALARY} imply {ENAME} → {SALARY}, while decisions d_2 and d_3 are not CXD decisions. However, if d_3 is decomposed into two decisions : one being d_3' that does not allow S to produce a relation with $R_{d_3'}$ = {ENAME, SALARY} and the other d_3'' that does not allow S to produce a relation with $R_{d_3''}$ = {ENAME, ASSESSMENT}, then both d_3' and d_3'' are CXD decisions.

3. An Algorithm for Enumerating Access Paths to be Protected

As defined above, the enforcement of a CXD decision $d_{X \cup Y}$ is the prevention of the corresponding subject from producing any relation which contains the same pairs <x, y> as determined by FD: $X \rightarrow Y$, where x and y are instances of X and Y, respectively. Instead of checking whether the subject has produced such a relation or not, we first compute the set of access paths, from each of which one can derive supposedly protected pairs <x, y> associated with the decision $d_{X \cup Y}$, and then prevent him from obtaining any pair via these paths. The set of access paths associated with a decision $d_{X \cup Y}$ are those that have a lossless join [1] and can be computed by means of the following theorem.

Theorem: Let X, Y, W be sets of attributes with no common element in between and $X \rightarrow Y$. Then $R_1 = X \cup W$ and $R_2 = W \cup Y$ have a lossless join if and only if $W \rightarrow Y$.

Proof: (i) if part: From [1], we know that if $R_1 \cap R_2 \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$, imply R_1 and R_2 have a lossless join. Since $W \rightarrow Y$ implies $W \rightarrow W \cup Y$, we have $R_1 \cap R_2 = W \rightarrow (W \cup Y) = R_2$.

(ii) only if part: Form[1], if R_1 and R_2 have a lossless join, then there exist sets Y_1 and Y_2 such that $R_1 \cap R_2 \twoheadrightarrow Y_1$, $R_1 \cap R_2 \twoheadrightarrow Y_2$, $Y_1 \cap (R_1 \cup R_2) = R_1$,

and $Y_2 \cap (R_1 \cup R_2) = R_2$, where $R_1 \cap R_2 \twoheadrightarrow Y_1$ denotes the multivalued dependency [1,3] of Y_1 on $R_1 \cap R_2$. Let $Y_1 = R_1$ and $Y_2 = R_2$. We have $R_1 \cap R_2 \twoheadrightarrow R_1$ and $R_1 \cap R_2 \twoheadrightarrow R_2$. That is, $W \twoheadrightarrow X \cup W$ and $W \twoheadrightarrow W \cup Y$ and hence $W \twoheadrightarrow X$ and $W \twoheadrightarrow Y$.

Case(a) : $W \twoheadrightarrow Y$

The mixed rule of inference states that if $A \twoheadrightarrow B$ and $C \rightarrow D$, where $B \supseteq D$ and $B \cap C = \phi$(empty set), then $A \rightarrow D$. Let $A = W$, $B = Y$, $C = X$, $D = Y$. Since $W \twoheadrightarrow Y$ (i.e., $A \twoheadrightarrow B$) and $X \rightarrow Y$ (i.e., $C \rightarrow D$), where $Y \supseteq Y$ (i.e., $B \supseteq D$) and $Y \cap X = \phi$(i.e., $B \cap C = \phi$), imply $W \rightarrow Y$ (i.e., $A \rightarrow D$).

Case(b) : $W \twoheadrightarrow X$

Since $X \rightarrow Y$, imply $X \twoheadrightarrow Y$. If $W \twoheadrightarrow X$ and $X \twoheadrightarrow Y$, using the transitivity property of multivalued dependencies, we have $W \twoheadrightarrow (Y - X) = Y$ since $X \cap Y = \phi$. Now we have the same case as (a).

Complete the proof of the theorem.

It is noted that in fact W and X have a mutually mutivalued dependency, i.e., $W \twoheadrightarrow X$ and $X \twoheadrightarrow W$. Any set of attributes, say W, on which Y is dependent will form (with X) a path : X-W-Y from which one can obtain the same pairs <x,y> as determined by FD: $X \rightarrow Y$. We say that paths X-Y and X-W-Y are equivalent. Given a CXD decision $d_{X \cup Y}$, we defined the set F(Y) consisting of sets of attributes as follows :

(1) Y is in F(Y).

(2) If V is in F(Y), and $U \rightarrow V$, then U is in F(Y).

(3) No element is in F(Y) unless it so follows from (1) and (2).

Thus the set F(Y) has at least two elements. Define $\tilde{F}(Y)$ is the set F(Y) excluding X and Y, that is, $\tilde{F}(Y) = F(Y) - \{X, Y\}$. Thus, any element in $\tilde{F}(Y)$ will form (with X) a path equivalent to X-Y. Furthermore, if U and V are in F(Y), then paths U-Y and U-V-Y (or V-Y and V-U-Y) are equivalent. Since X-U-Y (or X-V-Y)

is equivalent to X-Y, thus X-U-V-Y (or X-V-U-Y) is equivalent to X-Y. If $\tilde{F}(Y)$ has n elements, the total number S(n) of paths, originating in X and ending in Y, equivalent to X-Y is $S(n) = \sum_{i=1}^{n} C_i^n \cdot i! = \lfloor n!e - 1 \rfloor$, where e is a natural number and $\lfloor x \rfloor$ is the greatest integer less than or equal to X.

Based on the theorem above, we can conclude that the total number of paths originating in X and ending in Y, from which one can obtain the same pairs <x,y> as determined by FD: X → Y is S(n) + 1 (including the path X-Y). Since if W' is not in $\tilde{F}(Y)$ but the path X-W'-W-Y is equivalent to X-W-Y, where W is in $\tilde{F}(Y)$, this means X-W'-W and X-W are equivalent. That is , $R_1 = X \cup W'$ and $R_2 = W' \cup W$ has a lossless join. From[1] (see (ii) of the theorem), we have W' ↠ W and W' ↠ X. Following the same proof as cases (a) and (b) of the theorem (since W → Y and X → Y), we have W' → Y. This implies W' has to be in F(Y) which contradicts the assumption.

Consider a database consisting of a single relation EMP' (EMP#, ENAME, SALARY, STATUS, AGE) with EMP# and ENAME as candidate keys (i.e., having the unique identification property). Assume that {ENAME} → {STATUS}, {STATUS} → {SALARY}, and {ENAME} → {AGE}. We further assume that the subject S is subject to the decision d_1. Associated with the decision d_1 is the set F({SALARY}) = {{EMP#},{ENAME}, {STATUS},{SALARY}}. And $\tilde{F}$({SALARY}) = {{EMP#},{STATUS}} and S(n) = S(2) = $\lfloor n! \cdot e - 1 \rfloor = 4$. Thus, the total number of paths that should be protected is 5. They are {ENAME} - {SALARY},{ENAME} - {EMP#} - {SALARY},{ENAME} - {STATUS} - {SALARY},{ENAME} - {EMP#} - {STATUS} - {SALARY}, and {ENAME} - {STATUS} - {EMP#} - {SALARY}.

The computation of access paths to be protected depends on the set of FDs associated with the database. The complete set of FDs is the union of the given set of FDs and the set of FDs implied by the class of semantic dependencies of FDs, multivalued-dependencies, and join-dependencies [9].

4. Enforcement and Implementation

Each CXD decision $d_{X\,UY}$ is associated with a set of access paths, denoted as P_d, from each one can obtain the pairs <x,y> that should be protected. Enforcement of the decision is reduced to preventing the corresponding subject to obtain these pairs via any of these paths.

There are at least three enforcement schemes which can prevent a subject from obtaining the supposedly protected pairs <x,y>. The first one is to prevent the subject from seeing the instances x's or y's, but not necessarily both. Any access to x's or y's is not disallowed. This scheme views each CXD decision as a context-independent decision, which is not a suitable one because it does not allow the subject to obtain the data he has the right to see. The second one is to keep the subject from seeing a specific pair of adjacent sets of objects (or attributes) in each path. For example, it may want to disallow the subject to see the ith pair (X_{i-1}, X_i) of the path : $X_0(=X) - X_1 - X_2 - \ldots - X_{m-1} - X_m$ (= Y). This scheme converts the decision $d_{X\,UY}$ to S(n) + 1 decisions without depending on access history informatin, where S(n) + 1 is the total number of paths in P_d. Any access to a set of objects which contains the specific pair of sets of objects will be disallowed. The checking of whether an access violates any security decision or not depends on the objects being interrogated only. This scheme gives the subject larger flexibility of access than the first one.

The third scheme is to keep the subject from accessing the pair of adjacent sets of objects last accessed in each path. If the jth pair (X_{j-1}, X_j) of the above path is the one last accessed when the database has been used for a certain period, then a history independent CXD decision on the pair (X_{j-1}, X_j) will be added to the system. Any further access to that pair (not individuals of the

pair) will be subject to the decision. The pair of adjacent sets of objects which is to be protected is determined by sequence of requests that have been submitted to the system. We believe that this scheme gives the subject maximum flexibility of access. It requires the system to keep (possibly) a great amount of access history information, however. The history information can be kept by associating a counter with each path in the set P_d. Whenever a query is submitted, a checking is made which examines whether there is any pair in each path contained in the set of attributes of the query. (Each query defines a set consisting of all attributes appearing in the query.) Any relevant counter have to be properly updated and the corresponding pair in each path has to be deleted so that each pair will not be counted twice in each path. Once a counter has the value one less than the number of pairs in the corresponding path (i.e., there is one pair not yet been accessed), a CXD decision on the last accessed pair should be added to the system. The new decision will reside in the system and will be invoked for enforcement if any further access to that pair occurs.

The second scheme is obviously easiler implemented than the third one, since the system does not need to have long "memory" to keep track of previous accessed records in order to grant or deny a present request. Furthermore, it can be implemented to perform adequately without significant increase in response time, and (security) cost. It is therefore recommended.

5. Conclusion

We have defined and enforced a class of CXD security decisions. A CXD decision $d_{X \cup Y}$ which exihibits history information of access can be enforced by first enumerating all the possible access paths which can relate X-instances and Y-instances and then protecting each path by a history independnet CXD decision. Any deduction of context-sensitive information based on data semantic dependencies thus is disallowed. We have also suggested three ways of implementation.

6. References

[1] Aho, A. V., Beeir, C., and Ullman, J. D., "The Theory of Joins in Relational Databases," ACM TODS 4,3, September 1979, pp.297-314.

[2] Conway, R. W., Maxwell, W. O., and Morgan, H. L., "On the implementation of Security Measures in Information Systems," Comm. ACM 15, 4, April 1972, pp.211-220.

[3] Fagin, R., "Multivalued Dependency and a New Normal Form for Relational Databases," ACM TODS 2,3, September 1977, pp.262-278.

[4] Fernandez, E. B., Summers, R. C., and Coleman, C. D., "An Authorization Model for a Shared Data Bases," Proc. ACM-SIGMOD Conf., San Jose, Cal., May 14-16, 1975.

[5] Hartson, H. R., and Hsiao, D. K., "Full Protection Specifications in the Semantic Model for Database Protection Languages," Proc. of ACM Annual Conf., October 1976, Houston, Texas, pp.90-95.

[6] Hong, Y. C., and Su, S. Y. W., "A Protection Mechanism for Cellular-Logic Devices," paper under review for the IEEE Transactions on Software Engineering.

[7] Hsiao, D. K., Kerr, D. S., and Modnick, S. E., "Privacy and Security of Data Communications and Data Bases," Proc. 4th Int'l Conf. on VLDB, W. Berlin, W. Germany, September 1978.

[8] Hsiao, D. K., Kerr, D. S., and Nee, C. J., "Context Protection and Consistent Control in Data Base Systems," Technical Report OSU-CISRC-TR-73-9, Computer and Information Science Research Center, Ohio State Univ., February 1974.

[9] Maier, D., Mendelzon, A. O., and Sagiv, Y., "Testing Implications of Data Dependencies," ACM TODS 4, 4, December 1979, pp.455-469.

[10] Stonebraker, M., and Wong, E., "Access Control in a Relation Database Management System by Query Modification," Proc. of ACM Annual Conf., November 1974, pp.180-192.