PROTECTING COLLABORATION

Gio Wiederhold*
Stanford University, CA

Michel Bilello
Stanford University, CA

Vatsala Sarathy
Oracle Corp., Redwood City, CA

XiaoLei Qian
SRI International, Menlo Park, CA

June 26, 1996

Abstract

The TIHI (Trusted Interoperation of Healthcare Information) project addresses a security issue that arises when some information is being shared among collaborating enterprises, although not all enterprise information is sharable. It assumes that protection exists to prevent intrusion by adversaries through secure transmission and firewalls. The TIHI system design provides a gateway, owned by the enterprise security officer, to mediate queries and responses. The enterprise policy is determined by rules provided to the mediator. We show examples of typical rules. The problem and our solution applies not only to a healthcare setting, but is equally valid among collaborating enterprises and in many military situations.

1 Introduction

We address an issue in the protection of information that is starting to arise as the basic infrastructure for secure transmission and storage enters into practice. We assume an environment where encrypted transmission, firewalls, passwords, and private and public keys provide adequate protection from adversaries. The problem which remains, and addressed here, is now to enable selective sharing of information with collaborators, without the risk of exposing related information in one's enterprise domain or enclave that needs to be protected [1]. We will first sketch some examples to clarify the problem and then formulate the informal model for our work.

In a hospital the medical record system collects a wide variety of information on its patients. Most information on a patient must be accessible to the treating healthcare personnel, including community physicians, and a substantial fraction to the hospital billing clerks [2]. Similar data are requested by insurance companies, and certain data and summarization are due for hospital accreditation and public health monitoring. Results for all of these customers must be handled distinctly.

In a manufacturing company collaborations are often formed with suppliers and marketing organizations. Such virtual enterprises are formed to design, assemble, and market some specific products. Design specifications and market intelligence must be rapidly shared to remain competitive. These collaborations overlap, producing security problems which are stated to be the primary barrier to the acceptance of this approach [3]. Uncontrolled sharing of proprietary data is too risky for a manufacturer to grant a supplier. The supplier will also be wary of giving information to the customers.

In a joint military action situation, information must be shared from a variety of sources with a variety of forces, one's own and allies'. The source information ranges from current force status, logistics backup, to intelligence about the opponents. While opponents should be denied all information, not all of one's troops are authorized to access intelligence sources, and one's allies may be further restricted.

*Supported by NSF grant ECS-94-22688

These three scenarios have the following commonality.

1. We are dealing with friends, not enemies, and should provide relevant information expeditiously.
2. The collected information is not organized according to the needs of a security protocol.
3. It is impossible to rigorously classify the data, a priori, by potential recipient.
4. It cannot be fully determined from the query whether the results combine information which should be withheld.

For instance, a medical record on a cardiac patient can include notations that would reveal a diagnosis of HIV, which should not be widely revealed, and withheld from cardiology researchers. A design document on a plastic component, to be outsourced, also indicates the incorporation of a novel component supplied by another manufacturer, which provides a competitive advantage. Military planning information indicates intelligence sources which are not to be made public to one's allies.

Our model formalizes the role of a security officer who has the responsibility and the authority to assure that no inappropriate information leaves an enterprise domain. A firewall protects the domain vis-a-vis invaders. Distinct gateways, each owned and controlled by a security officer, provide the only legitimate pathways out of, and into, the domain. This gateway is best envisaged as a distinct computer system; we refer to such a system as a "security mediator", placed as sketched in Figure 1. In the security mediator the policies set by the enterprise on security and privacy are implemented, under control of, and through interaction with the security officer. Databases and files within the domain can provide services and meta-data to help the activities of the security mediator, but cannot be fully trusted. The security mediator is able to use secure communication and authentication of outside requests.

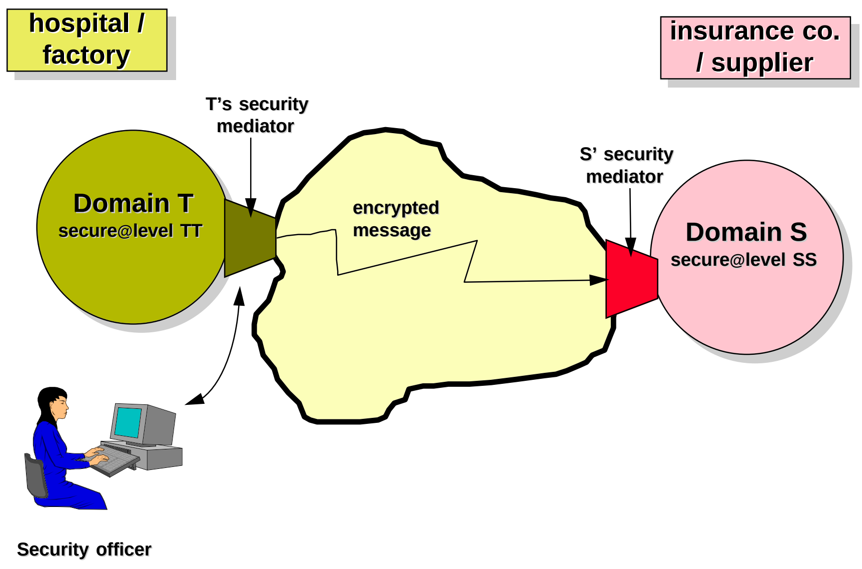

Figure 1: Security mediator setting.

It is important to recognize, as sketched in Figure 2, that validation of communication content must occur both with respect to the query and the responses. For instance, it is inadequate to allow a validated researcher in cardiac diseases to receive all records on cardiac patients, if that also includes HIV cases. Depending on institutional policy, such cases will be omitted or sanitized.

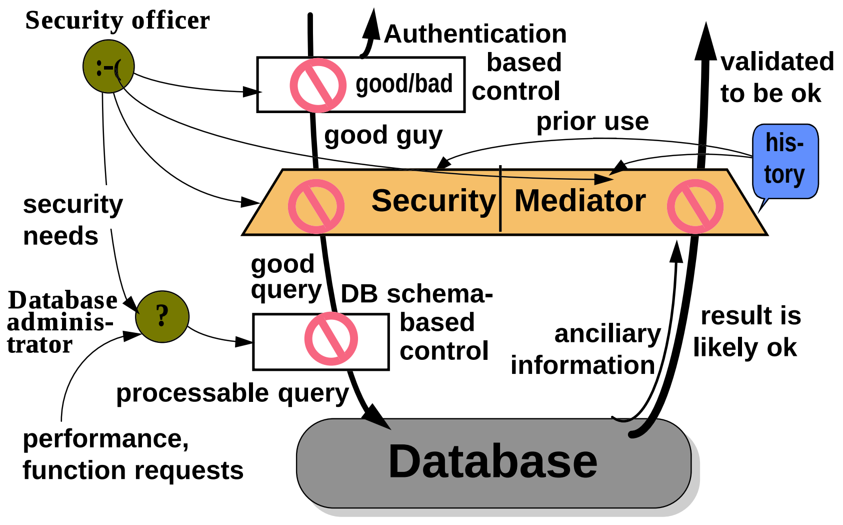

Figure 2: Paths to be checked.

2 System Design

The mediator system consists of modules that performs the following tasks:

- Processing of query (*pre-processing*)
- Communication with databases (submission of query and retrieval of results)
- Processing of results (*post-processing*)
- Writing into a log file

The mediator is designed to safeguard the privacy of the data. There is a two-way fence inside the mediator that intercepts queries coming in and, likewise, results going out. Corresponding to each side of the fence is a set of rules that assesses the legitimacy of queries and results respectively. When a query is sent by a user from the outside world, the mediator applies a set of rules to ensure the query's validity. For example, in a medical application, the mediator will obviously prevent those queries requesting patient names, social security numbers, etc.

The rule system permits fully validated requests and/or validated responses to pass without direct interaction by the security officer, but any other request or response will be presented to the security officer. The security officer then decides whether the request can still be granted. If the results of a query are rejected for rule violation, they are sent to the security officer along with the query and the identity of the user who originated it. If a result should contain information that is questionable, then it is passed to the security officer, who can approve it, edit it prior to approval, or reject it.

The rules balance the need for preserving data privacy and for making data available. Data which is too tightly controlled would be less available and useful for outside users. Conversely, a sufficient level of protection of data privacy must be maintained.

The mediator system can operate fully interactively or partially automatically. A reasonable goal is the automatic processing of say, 90% of queries and 95% responses, but even a fully manual system will provide benefits, as summarized in the conclusions. Even when operating automatically, the security mediator remains under the security officer's control. It does not function like a "black box" but rather keeps the security officer involved in its operation. For example, rules are modifiable by the security officer at all times. In addition, daily logs are accessible to the officer, who can then keep track of the transactions.

The mediator system and the databases typically reside on different machines. Thus, since all queries are processed by the mediator, the database need not be multi-level secure unless it operates in a particularly high security setting.

3 The Rule System

In order to automate the process of controlling access and ensuring the security of information, the security officer must enter rules into the system. The security mediator uses these rules to determine the validity of every query and make valuable decisions pertaining to the dissemination of information. The system helps the security officer enter appropriate rules and update them as the security needs of the organization change.

The rules are simple, short and comprehensive. They are stored in the database with all edit rights restricted to the security officer. If no rules are entered into the database, then the system operates in the manual default mode, whereby access is still possible but all queries and responses pass via the security officer. Some rules may be related to others, in which case the most restrictive rule automatically applies. The rules may pertain to users, sessions, tables or any combinations of these.

Once they are entered into the system by the officer, all the rules will be checked for every query issued by the user in every session. All applicable rules will be enforced for every user and the query will be forwarded only if it passes all tests. Unless a rule permits explicit pass through, it goes to the security officer. In the event a rule is violated by a query, the error message will be directed to the security officer and not to the end user. Thus, in such cases, the users will not see the error message. This is necessary because even error messages could be interpreted and meaningful inferences could be made, or the user could rephrase the query to bypass the error. The errors as well as all queries will be logged by the system for audit purposes.

Because the results retrieved for a given query can be highly unpredictable, it is not sufficient to validate queries. Thus, even when the query has been validated, the results are also subject to screening by a set of rules. As before, all rules are enforced for every user and the results are accessible only if they pass all tests. Also, if the results violate a rule, an error message is sent to the security officer but not to the user.

Not only are the rules easy to comprehend and to enter into the system, they are also powerful enough to enable the officer to specify requirements and criteria accurately, so that whenever users may see all information, they should be allowed to do so and whenever information is restricted, they should not have access to it. The users in the system are grouped as cliques and rules may apply to one or more cliques. The security officer has the authority to add or delete users from cliques and to create or drop cliques. Similarly, columns in tables can be grouped into segments and query/results validations could be performed on segments.

The rules can be classified as *set-up* or maintenance rules, *pre-processing* (query) rules and *post-processing* (result) rules. Some rules may be both *pre-* and *post-processing* rules. Examples of *pre-processing* rules include number of queries per session for the clique, session time, session hours, statistical query only, etc. *Post-processing* rules include minimum rows retrieved, session time, intersection of queries, user hours, vocabulary matching. A more comprehensive list of rules can be found in appendix. The rule type is indicated in parenthesis.

3.1 Application of rules

The following sequence of rules is applied for every request.

- When the user enters a query, the mediator parses the query. If parsing is not successful, an error message is sent out to the security officer.
- Next, the security mediator checks to see if the user belongs to a clique. If not, an error message is sent to the security officer.
- Then, it checks to see if access to all the columns specified in the SELECT and WHERE clauses in any segment is permitted to the members of the clique. If not, an error message is sent to the security officer.
- It then looks at every rule in the system of type *pre-processing* and validates the query against each. If any rule is violated, an error message is sent to the security officer.

At this point, the query is actually processed and results are obtained by the mediator.

- Now the *post-processing* rules are applied.
- On textual results, rules may specify that all words must come from a specified vocabulary. Any unknown term will be presented, with surrounding context, to the security officer, and if not approved, no result will be returned.
- Security officers can edit documents brought to their attention before releasing them. That should include 'whiteing-out' portion of graphics and design drawings.
- Lastly, further result modification is done as specified by the rules. Operations that can be invoked include random falsification of data and aggregation.
- Now the results are sent back to the user. Then the mediator updates internal statistics such are number of queries for the session, duration of the session, etc. It also updates the log files appropriately. This last step is done in all cases, whether or not there were errors.

4 View-Based Access Control

Most databases in place today were originally developed for internal use only. The security mechanisms available in these systems are intended for access by only a known, controllable, observable, and predominantly loyal internal user population, rather than unknown, unseen, and potentially adversarial external user populations [4]. Consequently, while internal access control based on user discretion might be satisfactory, external access control should support mandatory enforcement, before an enterprise can comfortably share its data with other partners in a collaboration.

Notice that the tables referred to in rules do not have to be base relations. They can be derived relations or views defined by arbitrary SQL queries. Hence, the set of rules collectively specifies a view-based access control policy.

Views in relational databases have long been considered ideal as the objects of access control, because they have a higher degree of logical abstraction than physical data and hence enable content-based or context-based security, as opposed to container-based security provided in operating systems.

View-based access control in relational databases was first introduced in IBM's System R [5], in which views expressed in SQL are the objects of authorization. It has been adopted by most commercial relational DBMSs. However, view-based mandatory access control has not been in widespread use because of the safety problem [6]. The safety question asks the following. Is there a database state in which a particular user possesses a particular privilege for data in a specific view? In container-based access control, different containers do not share contents. Hence, a secret label on a container guarantees that data in the container are not accessible to unclassified users. In view-based access control however, views might overlap because the same data might satisfy more than one view. Hence, a secret label on a view does not guarantee that data contained in the view are not accessible to unclassified users.

To support view-based mandatory access control, queries have to be analyzed and answers have to be filtered to ensure that data in a view are accessible by all and only those users who are authorized to access the view. We envision two types of query analysis.

1. *Analysis of single queries.* A query should be sufficiently constrained such that it only accesses those views to which the issuer of the query has authorization.

2. *Analysis of a sequence of queries.* A sequence of queries by the same issuer should be sufficiently constrained such that the issuer cannot compute, from the sequence of answers, data in views to which he does not have authorization.

4.1 Single Queries

The easiest way of enforcing mandatory access control is of course to require that a query be formulated in terms of those views to which the issuer of the query has authorization. For example, suppose that the following view is defined:

```
CREATE VIEW Drug_Allergy (patient_name, drug_name, notes)
SELECT      Patients.name, Drugs.name, Allergy.text
FROM        Patients, Drugs, Allergy
WHERE       Patients.id = Allergy.patient_id
AND         Drugs.id = Allergy.drug_id
```

on which the following rules are specified:

```
CREATE CLIQUE X
ADD USER      John_Doe X
LIMIT         X Drug_Allergy.
```

Then queries issued by user John Doe have to be formulated in terms of the view Drug_Allergy. For example, the following query by John Doe will be rejected by the security mediator,

```
SELECT Patients.name, Allergy.text
FROM   Patients, Drugs, Allergy
WHERE  Patients.id = Allergy.patient_id
AND    Drugs.id = Allergy.drug_id
       Drugs.name = xd_2001
```

even though it is equivalent to the following query, which will be accepted by the security mediator.

```
SELECT patient_name, notes
FROM   Drug_Allergy
WHERE  drug_name = xd_2001.
```

Therefore, the security mediator should not base acceptance decision of a query on the condition that the issuer of the query has authorization to all relations mentioned in the query, base or derived. Instead, the security mediator should try to reformulate the query using those views that the issuer of the query has authorization. If a reformulation is possible, then the reformulated query will be evaluated in place of the original query. Otherwise the original query is rejected. This approach will also facilitate the evolution of the security policy enforced by the security mediator.

4.2 Sequence of Queries

Access control on a per-query basis might not be sufficient. Even when a user has authorization to every query issued, he might be able to combine answers from a sequence of queries to derive data in a view to which he does not have access authorization. Such scenarios necessitate the need for the security mediator to keep track of the access history for every clique/user. For example, even if user John Doe is not authorized to access the view Drug_Allergy, he could issue the following two queries, assuming that he is authorized to both, and obtain data contained in the view Drug_Allergy.

```
SELECT Allergy.patient_id, Allergy.drug_id, Patients.name, Allergy.text
FROM   Patients, Allergy
WHERE  Patients.id = Allergy.patient_id.
```

```
SELECT Allergy.patient_id, Allergy.drug_id, Drugs.name, Allergy.text
FROM   Drugs, Allergy
WHERE  Drugs.id = Allergy.drug_id.
```

A critical issue in analyzing a sequence of queries is what we can assume about the computational capability of the user in combining the sequence of answers. For the above example, John Doe has to be able to perform join over the answers of the two queries in order to compromise the view Drug_Allergy. A reasonable assumption is that users have the same computational capability as in single queries. In other words, if users can issue project-select-join queries, then they can perform project, select, and join operations on a sequence of answers.

Another important problem is when queries are interleaved with updates, because even though John Doe might have already accessed a portion of the data in the view Drug_Allergy, say the first query above, enough time might have elapsed before he issues the second query above that the join between the two answers is empty. This could happen if for example the base relation Allergy only contains data for the most recent month, and John Doe waited over a month to ask the second query. In this case, the history log for queries on relation Allergy could safely be bound to one month.

Therefore, the security mediator should try to reformulate the view Drug_Allergy that John Doe is not authorized to using queries issued by John Doe. If a reformulation is possible, then the security policy on Drug_Allergy is violated.

5 Conclusion

We are addressing privacy and security maintenance in collaborative settings, where information has to be selectively protected from colleagues, rather than withheld from enemies. The problem only arises once a basic secure infrastructure is established. Today, privacy protection in healthcare is preached, but ignored in practice, putting many institutions at risk. In crucial settings, corporate and military security officers control input and output, but do so on paper, so that interactions are typically delayed by weeks, and high costs are incurred due to delays and misunderstandings. The primary barrier, as stated in [3], to the realization of virtual enterprises is 'Insufficient security controls. The corporations participating in a virtual enterprise are independent and frequently compete against one another'.

✪ **Be helpful to customer**	➲ **Be helpful to security officer**
✪ **Tell cust. re problems,** ***query may be fixed***	➲ **Tell cust. re problems,** ***sec. off may contact cust.***
✪ **Exploit DB meta-data**	➲ **Exploit customer info.**
✪ **Isolate transactions**	➲ **Use history of usage**
✪ **Ship result to customer**	➲ **Ship result to sec. off. with result description (source, cardinality)**

Figure 3: Differences in mediation for queries and for protection.

The approach we are developing provides tools for a security officer. Database systems have provided tools to control queries, under the aegis of the database administrator. We illustrated above that query-only tools are inadequate in complex settings, and we emphasized the need for view-based access control. In addition, the major role of a database administrator is to help customers get maximal relevant data, a task that often conflicts with security concerns as illustrated in Figure 3. Furthermore, the majority of data is not in database systems that provide security, and even less resides in costly, validated multi-level secure systems.

The concept of security mediator as an intelligent gateway protecting a well-defined domain is clear, simple, and the cost of modern workstations make it feasible to assign such a tool to a security officer. Like most security measures, the security mediator cannot offer a 100% guarantee, especially with respect to statistical data security. But having a focused node, with a complete log of requests and responses, and an incrementally improving rule collection, provides a means to ratchet protection to a level that serves the enterprise needs and policies effectively.

Examples of Rules

Rule	Remarks
1. `set logfile "x"` (Set up)	The table or path name to the log file
2. `create clique x` (Set up)	Create a clique of users called x
3. `add user user_name clique_name` (Set up)	add user called user_name to clique_name
4. `delete user user_name clique_name` (Set up)	
5. `drop clique x` (Set up)	
6. `create segment segment_name` (Set up)	
7. `set stat_only true/false` (Pre)	Only statistical info (average, median) allowed
8. `set clique stat_only true/false` (Pre)	Only statistical info (average, median) allowed for user
9. `set segment stat_only true/false` (Pre)	Only statistical info (average, median) allowed for queries on given table
10. `set user table stat_only true/false` (Pre)	Only statistical info (average, median) allowed for user, table combination
11. `limit queries_per_session x` (Pre)	Number of queries allowed in a session
12. `limit clique queries x` (Pre)	For a given user, number of queries allowed per session
13. `limit clique segment` (Pre)	limit all users in clique to columns/tables in segment. This specifies explicit pass through of results.
14. `set random on/off` (Post)	Random falsification of data to be performed or not
15. `set random on/off clique` (Post)	Random falsification of data to be performed or not for user
16. `set random on/off segment` (Post)	Random falsification of data to be performed or not for queries on given table
17. `set user table random on/off` (Post)	Random falsification of data to be performed or not for user/table combination
18. `limit min_rows_retrieved x` (Post)	Minimum number of matching rows for a given selection criterion
19. `limit clique min_rows x` (Post)	Minimum rows retrieved for a query by a given user
20. `limit segment num_queries x` (Post)	Number of queries allowed on a given table
21. `limit clique segment num_queries x` (Post)	Number of queries allowed on a given table for a given user
22. `limit intersection x` (Post)	No two queries can have an intersection greater than x rows
23. `limit clique intersection x` (Post)	No two queries by user can have an intersection greater than x rows
24. `limit segment intersection x` (Post)	No two queries on table can have an intersection greater than x rows

References

[1] D. Randolph Johnson, Fay F. Sayjdari, and John P. Van Tassell. Missi security policy: A formal approach. Technical Report R2SPO-TR001-95, National Security Agency Central Service, July 1995.

[2] Bill Braithwaite. National health information privacy bill generates heat at scamc. *Journal of the American Informatics Association*, 3(1):95–96, Jan/Feb 1996.

[3] Martin Hardwick, David L. Spooner, Tom Rando, and KC Morris. Sharing manufacturing information in virtual enterprises. *Comm. ACM*, 39(2):46–54, February 1996.

[4] G. Rettig. Use of multi-level secure systems in commercial environments. January 1991.

[5] P. P. Griffiths and B. W. Wade. An authorization mechanism for a relational database system. *ACM Transactions on Database Systems*, 1(3):242–255, September 1976.

[6] M. Schaefer and G. Smith. Assured discretionary access control for trusted RDBMS. In *Proceedings of the Ninth IFIP WG 11.3 Working Conference on Database Security*, pages 275–289, 1995.