Web Implementation of a Security Mediator for Medical Databases

Gio Wiederhold, Michel Bilello, Chris Donahue
Department of Computer Science, Stanford University
Stanford, CA 94305, USA, (650) 723-0872, gio@cs.stanford.edu, michel@cs.stanford.edu, donny@cs.stanford.edu

Abstract

Internet access to medical data has greatly facilitated information sharing. As health care institutions become more willing or more pressured to share some of their protected information, tools are being developed to facilitate the information transfer while protecting the privacy of the data. To this end, under the TIHI project, we have designed a *security mediator*, a software entity that screens both incoming queries and outgoing results for compliance with a medical institution's policies pertaining to data privacy. The system is under the control of a *security officer*, who enters simple rules into the system that implement the policies of the institution. In this paper, we describe the WWW implementation of the security mediator dual interface. The customer interface allows outsiders to request and receive filtered medical information from a hospital database. The security officer interface permits rule editing and resolution of cases not covered by the rule-set.

Keywords

Internet, medical information, sharing, security, privacy, database

1 INTRODUCTION

The TIHI project (Wiederhold *et al.* 1996, Wiederhold *et al.* 1996) has led to the design of a software system which allows secure sharing of medical information over the Internet (Rindfleisch. 1997). It is designed to support interaction with collaborators, rather than to prohibit attack by foes. Therefore, it is best used in conjunction with more defensive security techniques such as public/private key systems or firewalls.

The central component of the system, the *security mediator*, is a gateway between a medical institution (*e.g.*, a hospital), and outsiders (customers) that have a legitimate right to or interest in the institution's medical information.

Typical customers include:

- Public Health Agencies
- Medical Researchers
- Community or Specialty Physicians
- Insurance Companies

The security mediator is a tool that belongs to the *security officer*, the person responsible for enforcing the medical institution's policies concerning patient data security and privacy. The security mediator helps the security officer enforce these policies by translating a security policy into a set of rules. These rules belong to three categories, depending on whether they affect the customer himself (setup rules), queries submitted by the customer (query rules), or results that follow from queries (result rules). Setup rules verify the customer's name and password, and restrict the days and times when access is allowed (*i.e.*, a billing clerk may not be allowed to access the system on weekends). Sample query rules are Check Tables (the customer is restricted to specific tables in the database) and Check Select (the customer is limited to one *select* statement per query). The most important result rule is Check Dictionary, which checks each word contained in the results against a user-dependent dictionary to ensure that no sensitive textual information is released to the customer.

When a rule violation is detected, the query, the results, or both are sent to the security officer for review. The security officer can either approve the query as is, approve an edited form of the query, or approve a filtered set of results. Results checking is a crucial augmentation to the common model of secure access, in which no further validation is done after authentication, authorization, and certificate issue for access rights. In practice, results checking is a critical step, because the organization of the records in an institution is structured to deal with efficient local use, not with the secure matching of categories to external access rights.

All interactions with the system are recorded in the Audit Trail database. The security officer can use the Audit Trail to fine-tune the system. For example, if a customer has been entering queries in an attempt to circumvent an access restriction, the security officer can force all of the customer's queries to be reviewed manually. On the other hand, if a large number of safe queries get bumped to the security officer unnecessarily, he can relax the rules to allow the queries to pass without manual review. Each clique's dictionary can be incrementally constructed as well, with words being added as the results are manually approved by the security officer.

The subsequent sections give details of the system architecture and the WWW implementation.

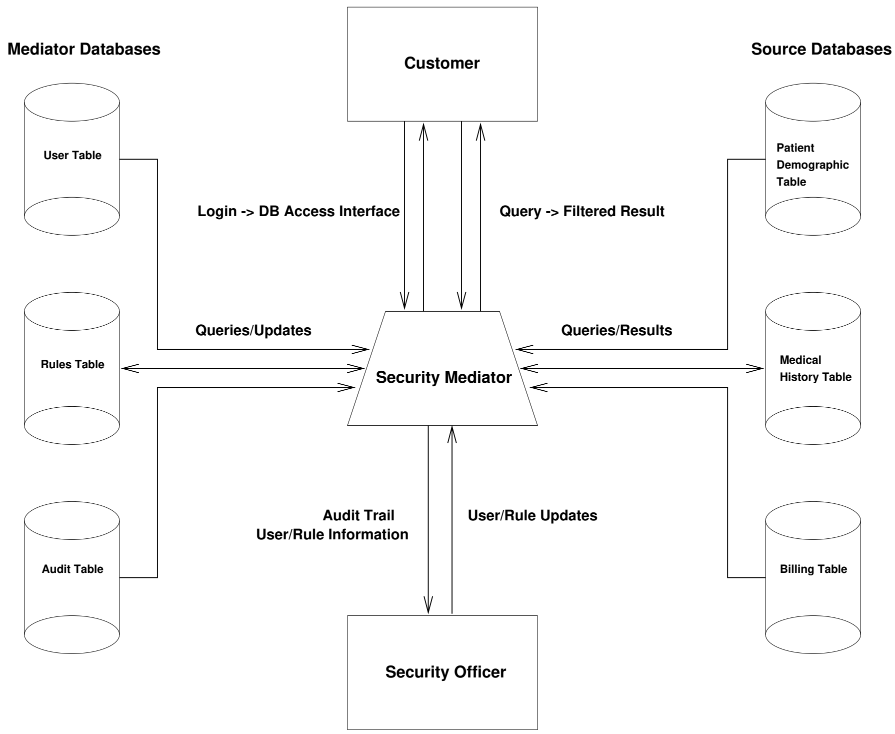

Figure 1 Overall Architecture

2 OVERALL ARCHITECTURE

The security mediator regulates access to database information by screening customers, queries *and* contents of results. The overall architecture is described below and diagrammed in Figure 1.

The back-end of the system is a source database containing the information that the customers are interested in. Tables from this database could include a Patient Demographics Table, a Medical History Table, and a Billing Table. This information resides on a central computer which can only be accessed by authorized personnel inside the medical institution. Therefore, the machine need not be multi-level secure.

Another component of the system is the mediator database, which stores the User Table (containing the usernames and passwords of registered customers), the Rules Table (containing the policy rules that govern login, query, and result screening), and the Audit Table (a record of all transactions, including date, time, user identification, queries, results, and possible rule violation statements). This database typically resides on a Unix workstation protected by a multi-level security system.

The mediator engine sits on the Unix station described above. It consists of

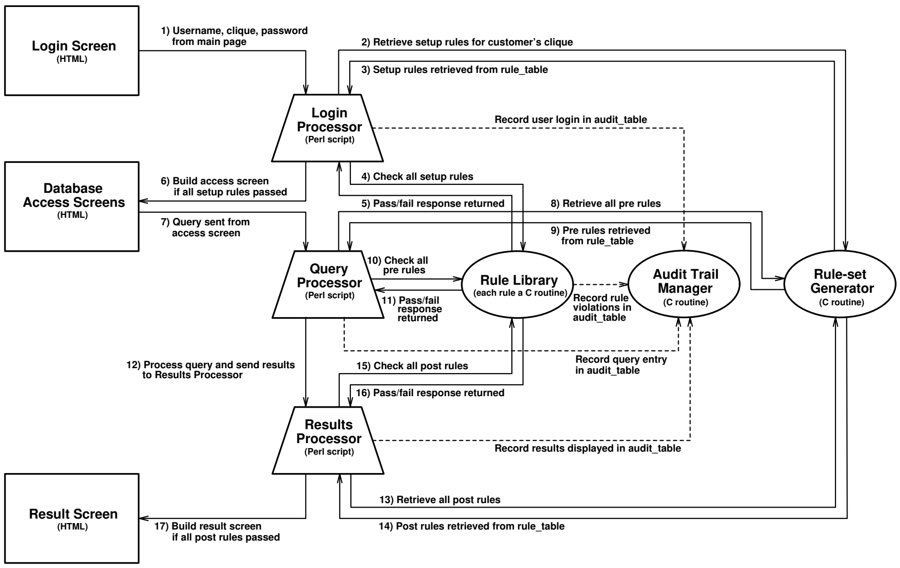

Figure 2 Details of Implementation

a collection of executable routines and scripts that work in concert to access the mediator and source databases in response to customers' queries or to the security officer's input.

Communicating with the security mediator engine are the Web-based customer and security officer interfaces. The customer interface allows customers to submit queries and to retrieve results from remote sites which run any operating system that supports a WWW connection. The security officer interface permits rule updates and audit trail look-ups. The security officer need not operate from the Unix station that holds the mediator engine and database.

3 IMPLEMENTATION

The current version of the TIHI prototype has been implemented using HTML forms and CGI scripts to connect the front-end Web interfaces with the internal databases. The architecture consists of four layers: an HTML forms user interface, Perl CGI scripts, C routines, and embedded SQL database functions. Details can be found in Figure 2. The interfaces for both the customer and the security officer are Web-based, and accessible from any browser.

3.1 Customer Interface

The customer interface consists of three screens: the login screen, the custom medical database access screen, and the result screen. Controlling the access and result screens are three Perl scripts: the Login Processor, the Query Processor, and the Result Processor.

The Login Processor reads the username, clique (membership group), and password from the login screen (Figure 3), and retrieves from the Rules Table the setup rules associated with the customer's clique. It then cycles through the relevant rules, calling each rule's corresponding C routine. Each routine returns a Pass or Fail flag. If a rule violation is detected, the Login Processor generates a standard error screen (in HTML) and returns it to the customer. No explanation is given to the customer as to why the login failed, since, given information, the customer may be able to circumvent the rule that restricted access. If all the setup rules pass, the customer is provided with a custom database access screen, which imposes a customer-dependent type of query. For example, a billing clerk would be prompted to enter a patient ID number, not a patient name, because billing clerks need not (and probably should not) know patient names in order to perform their transactions (Figure 4). Finally, the Login Processor records a successful login entry into the Audit Table.

The customer then enters a query either in SQL or by filling out custom forms, depending on the clique. For example, members of the patient clique can only request their own record, so the query is built by the mediator using the patient's name. Medical researchers, however, are allowed to enter full SQL requests. The query, as well as information about the customer and the clique, is sent to the Query Processor via HTML forms. The Query Processor then obtains the pre-processing (query processing) rules associated with the customer's clique, and cycles through the rules in the same manner as did the Login Processor. If the query passes all relevant rules, then the results are retrieved and processed by the Result Processor. All successful queries are recorded in the Audit Table. Unsuccessful queries are sent to the Review Queue (explained below).

Successful queries cause the mediator to retrieve the corresponding results and to screen them using the Result Processor. Post-processing (result processing) rules are retrieved from the Rule Table and applied to the results. If no rule violation occurs, the results are presented to the customer in HTML tables format. A rule violation causes the query that yielded the results to be sent to the Review Queue.

If a query is unsuccessful because of a rule violation, an entry is made in the Audit Table section called the Review Queue. The username, clique, query, and the rule broken are all stored in one entry of the Review Queue (Figure 5). The Security Officer can examine each entry and decide whether the query should be allowed. The Security Officer has the option of editing the query or rejecting it altogether. In the former case, the security officer edits either the query or the results (or both), and sends the results via e-mail to the query issuer (Figure 6). Otherwise, the customer is notified via e-mail that the request was rejected.

3.2 Security Officer Interface

The Security Officer enforces the security policies of the medical institution using the TIHI system. She builds cliques and rule-sets, monitors system usage, and approves or rejects queries and results that the Security Mediator disallowed.

The Security Officer HTML interface main page gives the Security Officer a choice of six functions which can be divided into two categories: system monitoring and general maintenance.

System Monitoring:

- Edit Results: The Security Officer can edit unacceptable results of queries in the Review Queue, and either send the filtered results to the customer or reject the request altogether.
- Edit Query: The Security Officer can either edit unacceptable queries and send the results to the customer, or reject the request altogether.
- Edit Dictionary: The Security Officer can add to or delete words from each clique's dictionary. The Edit Clique and Edit Dictionary functions, used in conjunction with the Review Queue, allow the Security Officer to refine a clique's rule-set and dictionary in response to the results being requested.
- View Audit Trail: The Security Officer can make a custom query on the Audit Trail database for reporting and investigative purposes, or to improve the rule-set.

General Maintenance:

- Create Clique: The Security Officer enters the clique's name, and the names and e-mail addresses of the new clique's users. She can also choose a rule-set for the new clique from the catalogue of rules in the system.
- Edit Clique: The Security Officer can add or delete users, add to or delete rules from the clique's rule-set, or change the parameters for the active rules in the clique's rule-set.
- Edit User Database: The Security Officer can add to or delete users from the database. If a deleted user is the only member of a clique, the clique is deleted as well.
- Edit Default Rules: The Security Officer can add or delete rules, and change the parameters for the active rules in the default rule-set.

Throughout the login/query entry/result retrieval process, the activities of the customer and any intervention by the Security Mediator are recorded in the Audit Table. This information is then used by the Security Officer to generate reports and uncover suspicious trends in the use of the system. The

Mediator itself uses the Audit Table to retrieve information necessary for rule application (*e.g.*, the Last Login Time rule).

4 FUTURE IMPLEMENTATION

The next generation TIHI system, which is under initial development, will differ from the current prototype in several respects.

First, the functionality of the rule-set will be increased. Instead of the static collection of rules currently used, the security officer will enjoy a dynamic rule environment. A rule compiler will be added, that will allow the Security Officer to construct rules. For example, suppose a Billing Clerk should have different access rights depending on the time of day. The Check Times, Check Tables, and Check Columns rules would be combined to create a rule that would give the Billing Clerk access to a particular set of tables and columns before 5 pm, and to a smaller set after 5 pm.

Another possible improvement would be to port the entire system to Java. A Java environment would allow for greater interactivity in both the customer and Security Officer interfaces. It would also simplify the underlying structure of the system, shrinking the number of layers from four to two (Java would be used both for the back-end routines that provide database access and for the front-end user interface screens that provide user input). version of your submission where possible.

5 SUMMARY AND CONCLUSIONS

The TIHI system consolidates the security needs of an institution's database system, placing the burden on the Security Mediator. By moving the security element of the system from the databases themselves to the Security Mediator, we have accomplished several goals. First, we have created a solution that can manage an institution's data sources while disregarding its specific physical instantiation. By rigorously parsing queries and filtering results, the Security Mediator is able to overcome security holes found in the underlying data organization and storage. Second, the Security Mediator serves as a security policy implementation, designed to be used by institutional management rather than by database or network administrators. This high-level approach places the control of computer-based data resources in the hands of those responsible for an institution's information, not those responsible for its computers.

The Security Mediator concept is not limited to the health-care domain. It is applicable wherever there is collaboration between different user domains (either within an institution, or between institutions) and users' access rights have little or no correlation to the underlying structure of the data. Military and manufacturing domains are potential future test-beds for security mediator technology.

REFERENCES

G. Wiederhold, M. Bilello, V. Sarathy, X.L. Qian (1996) A Security Mediator for Health Care Information. *Proceedings of the 1996 AMIA (formerly SCAMC) Conference*, 120-124.

G. Wiederhold, M. Bilello, V. Sarathy, X.L. Qian. (1996) Protecting Collaboration. *Proceedings of the NISSC 1996 National Information Systems Security Conference*, Baltimore, MD, 561-569.

D.R. Johnson, F.F. Sayjdari, J.P. Van Tassel (1995) Missi security policy: A formal approach, Technical Report R2SPO-TR001, National Security Agency Central Service.

B. Braithwaite (1996) National health information privacy bill generates heat at SCAMC. *Journal of the American Informatics Association*, 3(1):95-96.

M. Hardwick, D.L. Spooner, T. Rando, K.C. Morris (1996) Sharing Manufacturing Information in Virtual Enterprises. *Comm. ACM*, 39(2):46-54.

P.P. Griffiths, B.W. Wade (1976) An Authorization Mechanism for a Relational Database System. *ACM Transactions on Database Systems*, 1(3):242-255.

M. Schaefer, G. Smith (1995) Assured Discretionary Access Control for Trusted RDBMS. *Proceedings of the Ninth IFIP WG 11.3 Working Conference on Database Security*, 275-289.

T.C. Rindfleisch (1997) Confidentiality, Information Technology, and Health Care. *Communications of the Association of Computing Machinery*, Report SMI-97-0663.

National Research Council (1997) For the Record: Protecting Electronic Health Information, National Academy of Sciences.

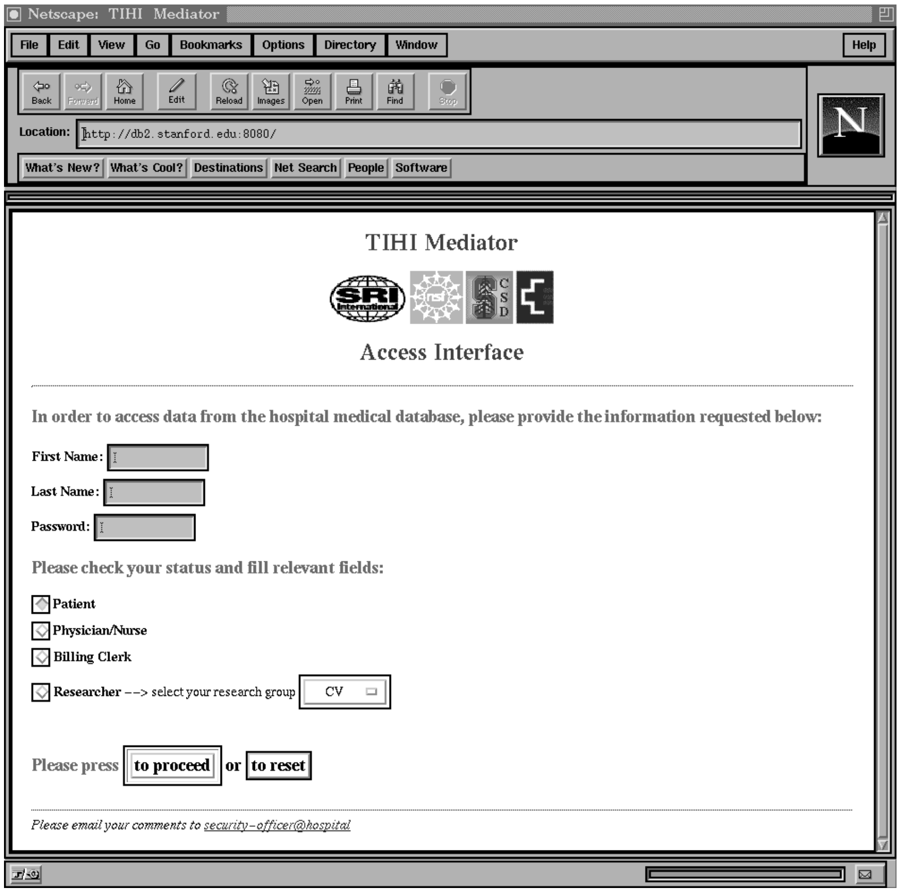

Figure 3 Login Screen.

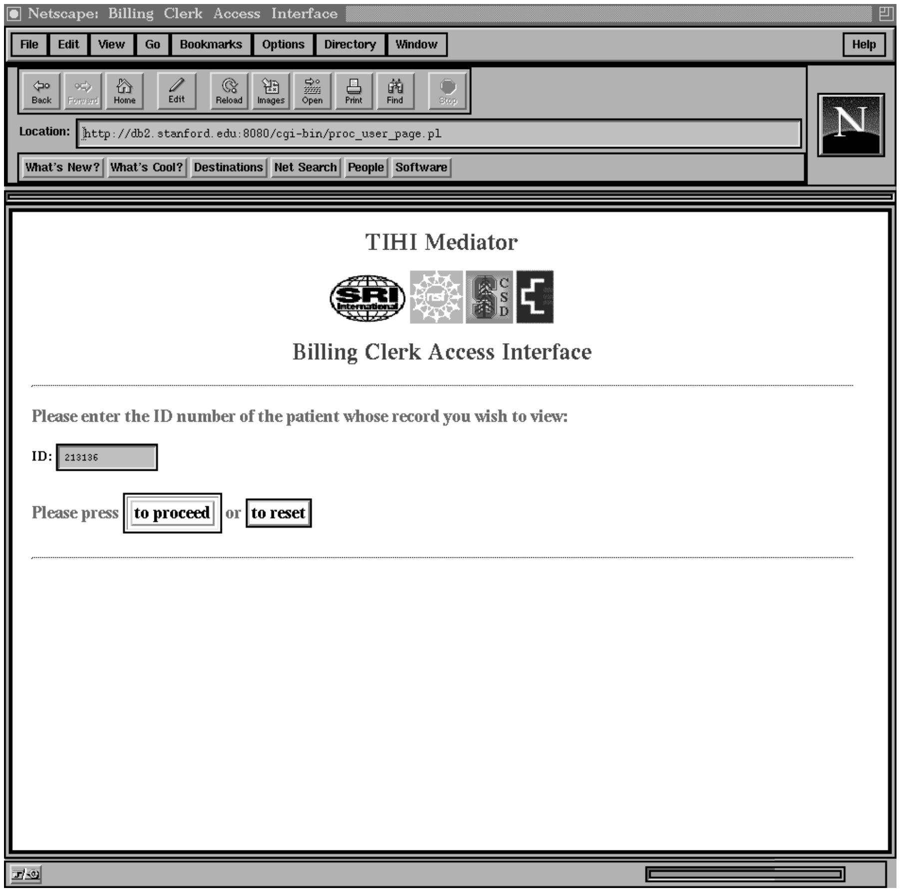

Figure 4 Custom DB Access Screen.

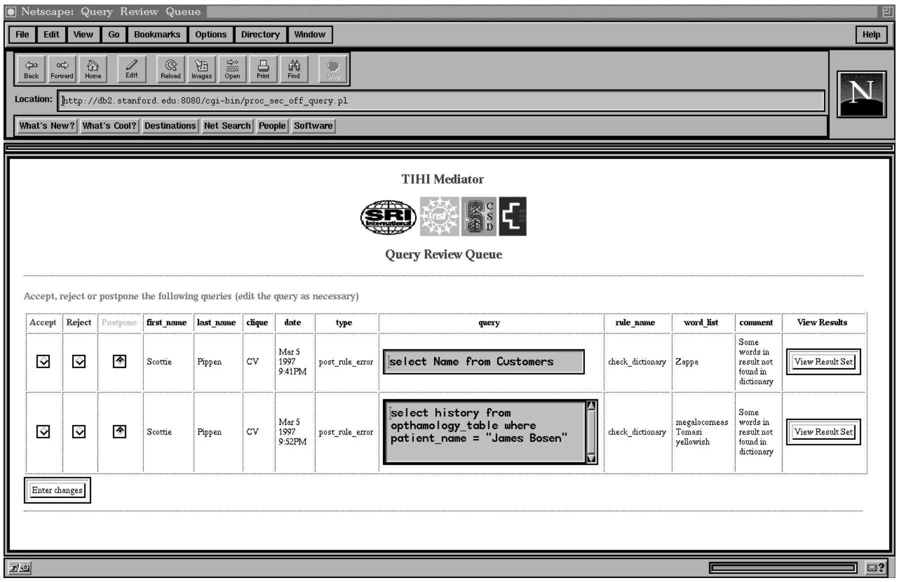

Figure 5 Review Queue.

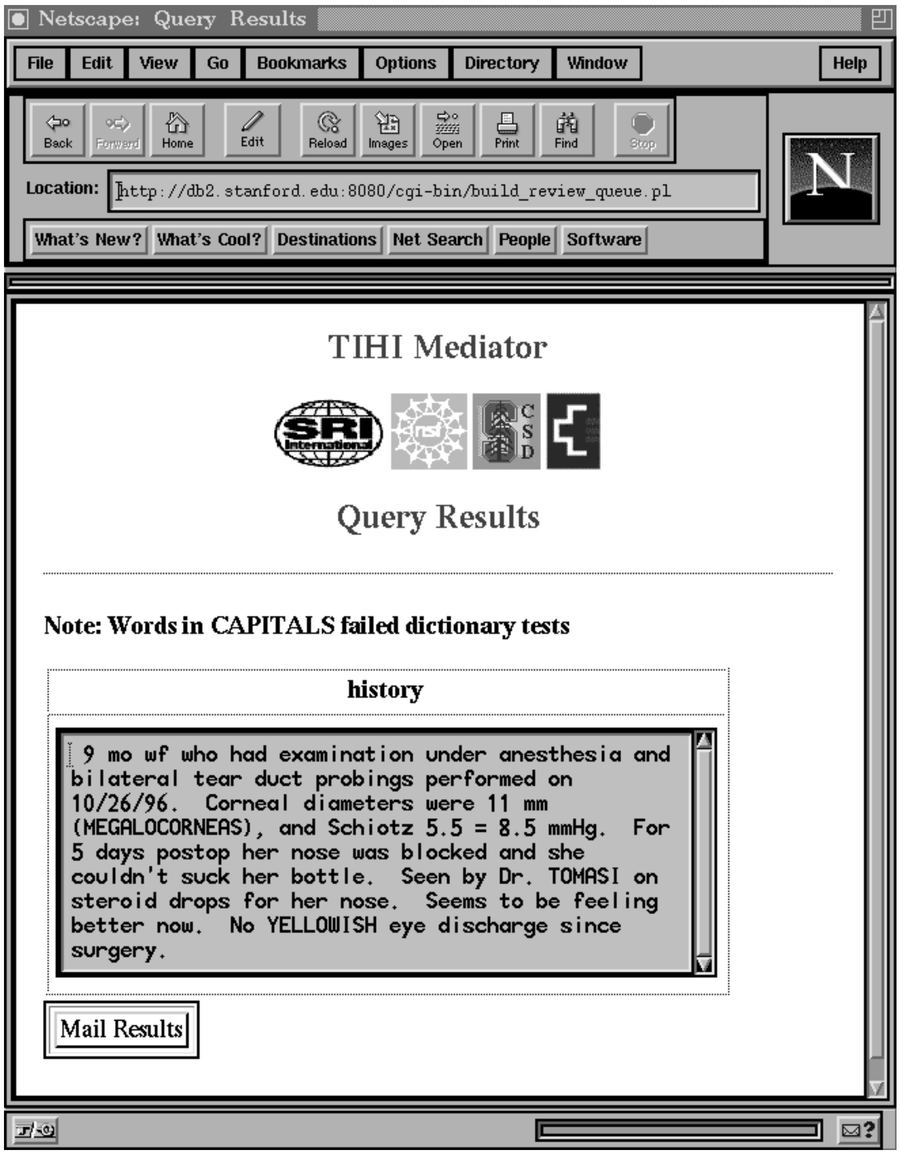

Figure 6 Results Editing Screen.

6 BIOGRAPHY

Gio Wiederhold is a professor of Computer Science at Stanford University, with courtesy appointments in Medicine and Electrical Engineering. Since 1976 he has supervised 27 PhD theses in these departments. His research focuses on large-scale software construction, specifically applied to information systems, the protection of their content, often using knowledge-based techniques.

Michel Bilello received his PhD in Electrical Engineering from Stanford University and is currently a medical student at the Stanford University School of Medicine.

Chris Donahue graduated with a BS in Computer Science from Stanford University.