SHIFT - Secure Heterogeneous InFormation Transfer for Relational Databases

Christopher A. Wood
caw4567@rit.edu

ABSTRACT

Data exchange between organizations is becoming a pervasive problem in the computer security landscape, particularly in the context of health information systems. With the timely access to accurate data, health practitioners are able to make informed decisions about patient treatments. Such data access is particularly important in the cardiovascular domain, which, according to the World Health Organization (WHO), is the primary cause of individual fatalities in developed countries. Access to a patient's medical history is critically important in order to successfully diagnose a wide array of cardiovascular problems [2].

Standard solutions for secure information exchange, such as the Electronics Data Interchange (EDI), have been deployed in clinical settings for several years. However, with the emergence of modern web applications and services, the adoption of solutions based on eXtensible Markup Language (XML) technologies have risen in popularity. With eXtensible Stylesheet Language Transformations (XSLTs), XML documents containing vital data from a separate organization can be molded to match a different, yet compatible, schema. The interoperability properties of XML have thus given rise to data interchange frameworks based on XML solutions in recent years [3].

JavaScript Object Notation, or JSON, is another increasingly popular data interchange format. While it is similar in XML in many regards, its syntactic simplicity makes it a very appealing alternative to XML for data exchange [1]. Since JSON is not a document markup language, it does not have the same extensibility of XML. However, its flat structure enables much easier and more efficient parsing of data, and thus makes it an appealing candidate for addressing the problem of secure data exchange.

SHIFT, a Secure Heterogeneous Information Transfer mechanism for relational databases, is inspired by the mediator design pattern for centralizing and managing pairwise interactions between many subjects using a publish-subscribe enrollment approach. Subjects will register with the SHIFT service by providing their own database schema and other pertinent identification information. This will enable SHIFT to push and pull data from subject databases using their provided schema. Once registered, subjects can push data to other known subjects by transferring data to the SHIFT service. Similarly, subjects can query for data from other known subjects by requesting data from the SHIFT service.

This framework builds upon the model presented in [3] with two very important enhancements. Firstly, JSON, rather than XML, is used as the data interchange format between relational databases. Mappings between database schemas, as well as the schemas themselves, can easily be represented using JSON. Secondly, the notion of data confidentiality while transmissions are being made is not discussed in [3]. Therefore, we plan on extending the framework to include a PKI scheme for encrypting data in transit.

To evaluate SHIFT, we will implement and deploy the service in an environment where two database-driven applications located on separate domains need to exchange sensitive data during normal operation. The context of these applications will be a medical environment, with one application representing an outsourced medical laboratory and the other representing a general practitioner information system belonging to a different domain. The laboratory application will have the following database schema:

- patient
  - fname, lname, bdate, patientId, orgId
- test
  - testId, desc, inputs, restrictions
- result
  - resultId, patientId, testId, outputs, meds
- medication
  - medId, desc, uid

In order to track relevant patient information, the general practitioner information system will have the following database schema:

- patient
  - fname, lname, bdate, patientId, organizationId
- history
  - histId, desc, patientId, date, medication
- visits

  - patientId, date, visitDesc
- medication
  - medId, desc, uid

In a typical scenario, the general practitioner will need to communicate with the laboratory to retrieve test information about patients that may have been entered by other cooperating organizations. In this case, the general practitioner would query SHIFT to fetch all of the required data. Conversely, the laboratory may need to request patient information from the general practitioner in the event that it cannot be provided by cooperating organizations, and would interface with SHIFT to acquire this data.

The laboratory application will only have two designated roles: a lab assistant and general manager. The lab manager is able to select all fields from the patient except the *organizationId* and all fields from the test and result tables, but is only able to read fields from the medication table. Similarly, the general practitioner application will have two roles: the general practitioner and patient. The general practitioner is able to select all fields from all tables, and the patient is able to read all data from all tables with the exception of the *patientId* field from the patient table.

The laboratory application is to be implemented as a console-based program with text-based input. This will enable test automation on this half of the environment. Flask, a Python-based micro framework for web application development, will serve as the endpoint to the SHIFT service for the laboratory application. Conversely, the general practitioner informational system will be implemented as a web-based application built using the Play! web framework. This will be done to expose more attack vectors to the underlying databases, and as a result, increase the comprehensiveness of our security features.

Aside from the security aspects of the SHIFT service used to transfer data between these two applications, we will investigate the following security features for each individual database application:

- Database authentication and password security policies. These features will enforce pre-defined access policies for the underlying databases that store patient-sensitive information.
- Application coding mitigation to prevent SQL injections and buffer overflows. These features will prevent common web-based attacks from threatening the integrity of the database state and contents.
- Database user role privilege management. This feature will ensure that database access is enforced through privileged views designated to each role.
- Encryption for data in transit between the application and database servers. This feature will prevent against wiretapping attacks used to reconstruct data sent between the applications and database servers.

1. REFERENCES

[1] D. Crockford. The application/json media type for javascript object notation (json). 2006.

[2] J. Fayn and P. Rubel. Toward a personal health society in cardiology. *Trans. Info. Tech. Biomed.*, 14(2):401–409, Mar. 2010.

[3] H. Jumaa, P. Rubel, and J. Fayn. An xml-based framework for automating data exchange in healthcare. In *e-Health Networking Applications and Services (Healthcom), 2010 12th IEEE International Conference on*, pages 264 –269, july 2010.