Architecture Design and Access Control of e-Health Portals

Shuo Lu

A Thesis
in
The Department
of
Computer Science

Presented in Partial Fullfilment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

January 2008

Your file Votre référence
ISBN: 978-0-494-40946-6
Our file Notre référence
ISBN: 978-0-494-40946-6

ABSTRACT

Architecture Design and Access Control of e-Health Portals

Shuo Lu

As an emerging form of enabling technology, Web-based e-Health portals provide patients easier accesses to their healthcare information and services. We design and implement such an e-Health portal which can integrate many backend medical services effectively. A major challenge in designing such a system is to meet critical security requirements, such as the confidentiality of patient data, the integrity of diagnosis results, and the availability of healthcare services. In this thesis I address the issue from the access control perspective. More specifically, I first propose a two-tier approach to access control for e-Health portals. The approach supplements existing Role Based Access Control (RBAC) capabilities with a rule-based access control module based on the classical Flexible Authorization Framework (FAF) model. I study conflict resolution and interaction between the two modules. I also address authentication for real-time services provided by remote service providers.

ACKNOWLEDGEMENTS

First, I would like to express my truly thanks to my supervisor, Dr. Rachida Dssouli, for her help, guidance and support to my study and research work in Concordia.

In addition, thanks to my co-supervisor Dr. Lingyu Wang with his kind suggestions and help. I would also thank all group members, and my friends: Cody, Mario and Frenkie.

To My Parents and Meris

Table of Contents

List of Tables

List of Figures

List of Acronyms

ACL	Access Control Lists
ACM	Access Control Matrix
AS	Authentication Server
ASL	Authorization Specification Language
BP	Blood Pressure
CORBA	Common Object Request Broker Architecture
DAC	Discretionary Access Control
DCOM	Distributed Component Object Model
ECG	Electrocardiograph
EPR	Electronic Patient Record
FAF	Flexible Authorization Framework
GUI	Graphic User Interface
HIS	Health Information System
HISA	Healthcare Information System Architecture
JPF	Java Page Flow
KDC	Key Distribution Center
LDAP	Lightweight Directory Access Protocol
MAC	Mandatory Access Control
MVC	Model View Controller
RBAC	Role Based Access Control
RMI	Remote Method Invocation
RPC	Remote Procedure Call

SOA	Service-Oriented Architecture
SS	Service Server
TGS	Ticket Granting Server
TGT	Ticket Granting Ticket
UDDI	Universal Description, Discovery and Integration
WSDL	Web Services Description Language
WSRP	Web Services for Remote Portlets
XML	Extensible Markup Language

Chapter 1

INTRODUCTION

1.1 Background

"e-Health is an emerging field in the intersection of medical informatics, public health and business, referring to health services and information delivered or enhanced through the Internet and related technologies" [Eys01].

The healthcare industry is undergoing fundamental changes. Examples of such changes include a shift from hospital-centric services to a more ambulatory system (with homecare, day care clinics, and so on) and the treatment of chronic diseases that actively involves the patient himself/herself [Bis05]. The emergence of Web-based e-Health portals is a natural result of such changes because such portals provide patients and healthcare professionals easy accesses to information no matter where they are. According to a recent survey, most patients say they are very interested in and capable of accessing healthcare information and services via a Web-based portal system [Eys01].

1.2 Motivation

The design of e-Health portal is, however, particularly challenging due to its unique functionality and security requirements. First, a traditional design of portal systems will encounter difficulties in integrating heterogeneous e-Health services implemented with different technologies. The complexity of such integration will make it difficult to extend an existing system with new services. Second, a general purpose Web-based portal usually cannot meet the security requirements of an e-Health portal system because the consequence of a security

breach is far more serious in the latter. For example, an inappropriate disclosure of patient data will lead to privacy breaches and legal issues, whereas an improper modification to diagnosis results or a denial of critical healthcare service may threaten a patient's health or even his/her life.

We address the above issues through the design and implementation of a secure Web-based e-Health portal. To meet the functional requirements, we adopt a service-oriented approach to the design of our portal. We then tackle various security issues involved in such a design. More specifically, we outline our solutions for authentication and authorization of users for local and remote services in different operating modes, for trust management between patients and doctors using PKI and biometrics, and for preserving patients' privacy through preference negotiation and database technology. We also discuss implementation issues of the proposed portal system.

1.3 Contributions

My contributions concentrate on the system architecture design and access control of e-Health portal systems. In the architecture design, I adopt Service-Oriented Architecture-based three-tier architecture, which includes portal server, portlet container, and service container. In the design of access control, I design a two-tier access control mechanism, which combines traditional role-based access control with rule-based access control [LHL+07] [HLL+07] [LLH+08].

1.4 Organization of the Thesis

The rest of the thesis is organized as follows:

- Chapter 2 presents technical background and related work of e-Health system architecture and access control models.
- Chapter 3 discusses the functional requirements of our e-Health portal systems and then describes my design of the system architecture.

- Chapter 4 mainly illustrates the requirements of access control in our e-Health portal systems and discusses issues within existing access control models when applied to our case and then proposes our two-tier access control model. Issues within the e-Health portal federation are also discussed in this chapter.
- Chapter 5 gives the implementation details of the system architecture, some medical services, and the access control engine.
- Chapter 6 draws conclusions of the thesis.

Chapter 2

RELATED WORK

This chapter describes the background related to our e-Health portals. Related work of e-Health systems and access control model are also illustrated.

2.1 Service-Oriented Architecture

Service-Oriented Architecture (SOA) provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations [oas]. Many enterprise architects believe that SOA can help businesses respond more quickly and cost-effectively to changing market conditions [Koc05].

SOA is a software system architecture, which enables various applications' components (exposed as services) to communicate with each others via well defined programming interfaces. Those interfaces are hardware platform, software platform and programming language-independent. The most popular SOA implementation has been using the Web services protocol stack. Such services are generally described by platform-independent XML documents, i.e. the Web Services Description Language (WSDL) which will be illustrated in the next section. The services communicate with each other in messages defined in XML since service consumers and providers may reside in heterogeneous environments, and they may know nothing about each other. Services are generally published to a registry by service providers, and consumers can discover and invoke the services regardless of their underlying implementations.

Besides Web Service, there are many other protocols that can be used to implement the

SOA. Such as DCOM (Distributed Component Object Model, which is Microsoft specific), RMI(Remote Method Invocation) and JINI (which are Java specific), and OMG's CORBA (Common Object Request Broker Architecture, which is platform and language independent). CORBA has been successfully applied in various areas ranging from telecommunications, e-commerce, to healthcare etc. However, the biggest challenge faced by CORBA is that it is hard to find a unique RPC middleware to support all the programming languages and all the platforms at a reasonable price [GKS02]. CORBA only supports for UNIX and Windows platform, and Visual Basic does not provide any support for CORBA and therefore limits its usage.

2.2 Web Services

Web services provide a standard means of interoperating between different software applications running on a variety of platforms and/or frameworks [w3c]. Web services are applications that expose their business logic, data and processes through programmatic interface. Unlike traditional Client/Server models, Web services generally use HTTP as the underlying communication protocol which allows messages to go through most of the firewalls (most of the firewalls' setting allow the access to HTTP port). Web services do not provide users with GUI (Graphic User Interface). However, developers can add Web services into their Web page or applications and offer users with GUI which contains the functionalities of the Web services.

The major advantages of implementing an SOA using Web services are that Web services are very simple and programming language and platform-independent. Moreover, there are many extended Web services specifications ranging from transactions, business process, messaging, transport, security, metadata to performance, which are on their way toward standardization. These provide enterprise-level integration in a standard way for services to interoperate with each other without any incompatibility and at the same time ensures the QoS and other requirements. As the three core specifications of the Web services protocol stack, SOAP, WSDL and UDDI form the initial specifications for Web services. Figure 2.1

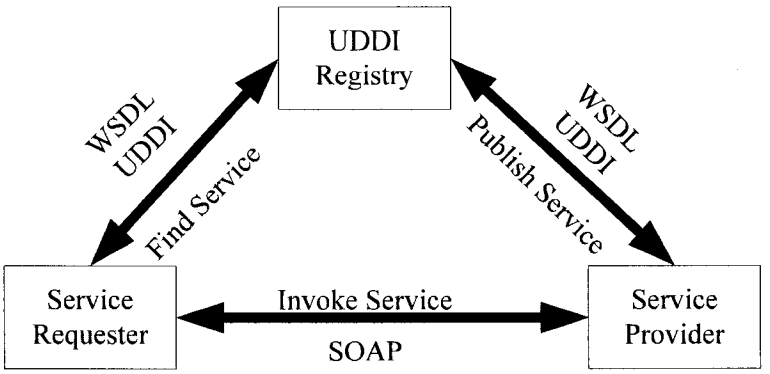

Figure 2.1: Web Services Architecture

describes the architecture of Web services. In the following, each of them will be introduced in details.

2.2.1 SOAP

SOAP is a protocol for exchanging XML-based messages between peers over computer networks. It provides a standard way to access Web services. SOAP is the foundation of the Web services protocol stack and provides the basis upon which other abstract layers are built. It generally uses the HTTP/HTTPS (other protocols like SMTP and XMPP can also be used) as its underlying protocol. The major advantage of using SOAP over other distributed protocols like GIOP/IIOP or DCOM is that SOAP with HTTP works well with network firewalls, whereas other protocols are normally blocked by firewalls.

SOAP 1.1 was proposed to W3C in May 2000 [w3c]. Currently, W3C is working on SOAP 1.2. A SOAP message is an XML document which contains the following XML elements: Envelop, Header, Body and Fault. The SOAP Envelop element is the root element of a SOAP message. It is used to identify that the XML document is a SOAP message and encapsulates all other XML elements. The SOAP Header element is optional and may contain application specific information like security features. The SOAP Body element is required and contains call and response information. The SOAP Fault element is optional and it represents error information when processing the message. Each Fault element must contain a faultCode element and a faultString element. The former one represents the code of the error and the

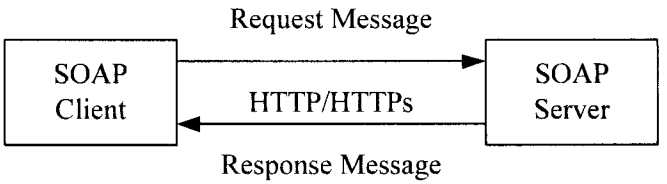

Figure 2.2: Exchanging Message with SOAP

latter one is used to provide the detailed information of the error.

Figure 2.2 depicts how SOAP messages exchanged between SOAP client and server. A SOAP client (also termed as service requester) is an application that creates a SOAP message containing the information needed to invoke remote methods in heterogeneous environments. It could be a traditional program, Web services or any other server-based applications (such as servlet, portlet etc.). A SOAP server (also termed as service provider) is a program generally resides in the Web server that listens, distributes and interprets the received SOAP messages. In Figure 2.2, the SOAP client sends a request (a SOAP message) to the SOAP server via HTTP/HTTPs. After receiving this request, the SOAP server sends it to the SOAP processor which resides in the server side and acts as an evaluator to validate this request against the XML schema. If it is valid, the SOAP processor invokes the Web service. Then the Web service processes this request and creates a response. After that, the SOAP processor wraps this response with SOAP message format and sends the response SOAP message back to the SOAP client over HTTP/HTTPs. Like the SOAP server, the SOAP processor in the SOAP client will parse and validate this response message and present the result to the SOAP client user interface.

2.2.2 WSDL

The WSDL (Web Services Description Language) is a specification recommended by W3C [w3c]. The current draft version is 2.0. It is an XML-based language used for describing Web services. Each WSDL document contains five major elements which describe three aspects of Web services. The types, message and portType elements are used to describe what tasks the service provides. The types element defines the data type used by Web services (Generally, for maximum platform independent purpose, WSDL uses XML Schema syntax

to define data types). The message element defines the data elements within each operation. These data elements can be compared to the parameters of a function call in traditional programming languages. The portType element describes the operations that a Web service can perform. It can be compared to a function in traditional programming languages. The binding element defines the message format and underlying transport protocol details for each port and the service element represents the location of the service. The main purpose of the WSDL is to provide human readable and machine interpretable information about the service.

2.2.3 UDDI

UDDI (Universal Description, Discovery and Integration), sponsored by OASIS [oas], is a platform-independent, XML-based registry, which enables Web services implementations to be published and discovered either within or between enterprises. A UDDI business registration contains three components: White Pages (describes basic information about the service provider, such as address, contact, and other identifiers), Yellow Pages (describes services by industry categorizations, service type or geography information) and Green Pages (describes technical information about services, such as interfaces and URL locations etc.).

UDDI eases enterprises to create a platform-independent and open architecture for quickly discovering and publishing businesses and services via Internet. Enterprises can describe and publish their services to the UDDI registry. Once an enterprise finds a potential business partner, they can quickly and easily begin trading. As shown in Figure 2.1, the service provider may publish its WSDL document in the UDDI registry and the service requester may retrieve it by searching the UDDI registry according to the keywords described in the above pages. When the service requester obtains the WSDL document for a service, it may invoke it according to the elements described in the WSDL document.

2.3 Portal

With the expanding of enterprise and business, it is necessary to have a centralized application that can integrate various applications and systems within the same place to share the resources. It also provides various users with a single point to access all of the applications over the Internet. All these can be achieved through portals. A portal lets users view each application or Web page in its own window, called a portlet, and a single browser window can contain one or multiple portlets. It is a collection of resources in an enterprise application that can be displayed in customizable, personalized, and audience-specific views called desktops [bea].

As the basic unit of portal, portlets are pluggable user interface components that are managed and displayed in a portal. Portlets produce markup fragments and then these fragments are aggregated into a portal page. Typically, a portal page contains a collection of portlet windows, where each of them displays a portlet. Thus a portlet resembles a Web-based application (such as a JSP, HTML page, Java Page Flow or Web services etc.) that is hosted in local or remote portal servers. The Java Portlet Specification (JSR168) is a standard that enables interoperability for portlets between different portals. This specification defines a set of APIs for interaction between the portlet container and the portlet addressing the areas of personalization, presentation and security [jsr]. Apache Pluto [plu] is a reference implementation of JSR168. Also, there are many other vendors (such as BEA, IBM, SUN, Oracle etc.) provide commercial implementations of the standard-compatible portlet container.

As shown in Figure 2.3, it is a simple portal platform in which the portal integrates multiple backend medical systems. The interfaces of these systems are represented in a single browser window to clients. Administrators resided in the portal assemble, configure and manage the portal via portal administration tools.

Some characteristics and benefits of portal solution are listed as follow:

- Intelligent resources integration: Integration of various enterprise applications, services and processes together is key to developing an environment that fully supports business

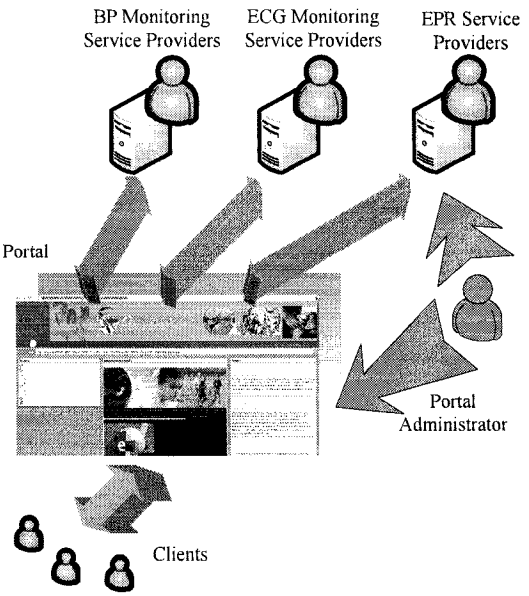

Figure 2.3: A Simple Portal Platform

processes.

- Rendering content on different devices: By using portal technology, content can be delivered to and displayed on multiple devices (such as PC, PDAs, smart phones etc.).
- Personalization: The ability to serve dynamic response to the user based on personal profiles.
- Rapid, easy management of Web content: Portal administrator may easily modify, add or remove content from the portal.
- Single Sign-on: Portal is a place where a user can authenticate once and gain access to the resources of multiple backend systems.
- Federation: This feature enables a portal to include remotely distributed resources in a standard way. It also reduces the cost for maintenance, testing and deployment and increases the reuse of portal components and interoperability.

2.4 Flexible Authorization Framework

Many access control policies have been proposed and are widely applied in information systems. However, in practice, only a specific policy (such as closed policy or open policy) can be applied in a system, which cannot satisfy increasing access control requirements. Thus, Flexible Authorization Framework (FAF) [JSSS01] is proposed to allows multiple access control policies to be applied within a single system. FAF allows positive, negative and hybrid (both positive and negative) policies to be specified to allow or deny an access in a powerful, declarative and flexible way. It also employs meta-policy to solve the conflict resolution problem when hybrid policy is specified.

FAF employs four tiers of policies to manage access control [NZ]. The first stage includes the description of subject and object hierarchies, and a set of authorizations according to the protection requirements. In this stage, conflict problems may occur since both positive and negative authorizations may be derived for executing an action on a given object, thus a subject could be authorized and denied to perform the action at the same time. To solve this problem, in the second stage conflict resolution policies are enforced. However, it is possible that access is neither authorized nor denied. Thus, the third stage employs the decision policies to make final decisions. Finally, integrity rules are used to identify and remove errors that may arise in authorization specification.

Based on this framework, security administrators may use the Authorization Specification Language (ASL), a stratified first-order logic language, to specify access control rules according to the protection requirements. ASL syntax includes a set of stratified predicates. The level of these predicates is corresponding to the stages stated above.

2.5 Kerberos

Kerberos is a network authentication protocol that is based on the Needham-Schroeder protocol. It is designed to provide strong authentication for client/server applications by using symmetric key cryptography [mit]. The core of the Kerberos is the Key Distribution Center (KDC), a trusted third party which contains two logical parts, i.e. an Authentication Server

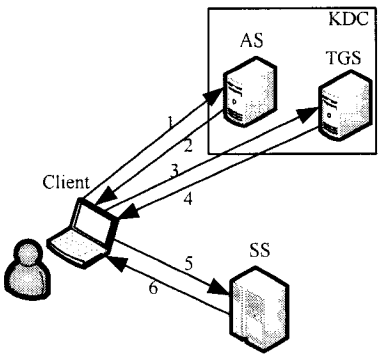

Figure 2.4: Simplified Description of Kerberos Protocol

(AS) and a Ticket Granting Server (TGS). AS maintains a database to store the credentials of users and issues users with Ticket-Granting Ticket (TGT) upon successful authentication. TGS is used to issue users with Client-to-Server ticket upon receiving valid TGT. Kerberos also enables mutual authentication (i.e. both the user and the server verify each other's identity). A free implementation of this protocol is available from the Massachusetts Institute of Technology [mit]. Figure 2.4 is a simplified description of Kerberos Protocol. It contains three phases.

Phase A: The Authentication Service Exchange

1. The client sends a plaintext which contains the identities of the client and the server to the AS.

2. The AS checks the identity of the client. If it is valid, the AS sends back the client two messages: the client/TGS session key (shared between the client and TGS) encrypted using the user secret key and the TGT (which contains the client ID, client IP address, ticket validity period, and the client/TGS session key) encrypted using the secret key of the TGS. The user can get the session key by using his/her secret key to decrypt the first message.

Phase B: The Ticket-Granting Service Exchange

3. When requesting services, the client sends two messages to the TGS: the first one contains the TGT and the ID of the requested service and the second one is an Authenticator (which contains the client identity and a timestamp), which is encrypted by the client/TGS session key.

4. Upon receiving the above two messages, the TGS decrypts the Authenticator using the client/TGS session key and sends the following two messages to the client: client-to-server ticket (which includes the client identity, client ip address, validity period and client/server session key) encrypted using the service's secret key and client/server session key encrypted with the client/TGS session key.

Phase C: The Client/Server Authentication Exchange

5. In order to use the service, the client sends the following two messages to the Service Server (SS): the first one is the client-to-server ticket which is encrypted using service's secret key and the second one is an Authenticator (which contains the client identity and a timestamp) encrypted using client/server session key.

6. Upon receiving the client-to-server ticket, the SS decrypts it using its secret key and increases the timestamp within the Authenticator by 1 and then encrypts it using the client/server session key and reply to the client.

Finally, the client decrypts the reply using the client/server session key and checks the value of the timestamp. If it is increased by 1, the client can begin to send the service request to the server.

2.6 Telemedicine and e-Health Systems

The increasing demand for e-Health services has led to many research efforts [KG06]. Different e-Health or other healthcare-related systems have been designed and implemented. Some of them are used in special areas, such as trauma [CG04], cardiology [FGT+03], neurosurgery [RCW+98], pathology treatment [MM04] etc. Some are used with special purposes, such as emergency [SRK+04] [wet98], aeronautic cure [FKM01], marine purpose [CRCM97], patient monitoring [KPK+01] etc. With the advantages of wireless technologies, there are also many wireless-based e-Health systems (i.e. m-Health system) emerging [E05]. For complete surveys of m-Health, refer to [VPI+01] [PKV+02] [IJZ04]. The common feature of these systems is that they only provide limited or special services to users or as the complement of the e-Health industry. Our goal is not to implement an e-Health system used

in a special area or for some purposes. Instead, we concentrate on proposing a framework for general e-Health systems, which can integrate most of the existing healthcare applications, modules or other healthcare related systems.

Many architectures have been proposed for telemedicine and e-Health system. In this section, we give a brief review.

HISA (Healthcare Information System Architecture) [RS99] is a European pre-standard for medical information systems that deals with the architecture of medicine information system. Even though this architecture has no direct relationship to the field of e-Health, we still can benefit from it. It adopts three-layer architecture and uses middleware (CORBA and DCOM) technologies to develop distributed healthcare information systems to enable different health centers to interoperate with each other on the basis of information consistency.

Later, SAMTA [sam] developed an open scalable architecture for multimedia telemedicine, applications which allows designers to efficiently use the network bandwidth. It classifies services into medical services, technical services and physical services according to different levels of complexity. This separation of services enables the technical details behind the medical services to be hidden from users. Among these levels of services, medical services are those offered to the user directly. Each service corresponds to one or more operations in the real world. Technical Services do not have direct relationship with the medical field. It only concentrates on network transmission, local services, security, compression and human interaction. Physical Services refer to the structures, protocols and services used by the technical services. They are close to the underlying hardware and act as an interface between the application and the environments (such as hardware and software platforms).

Authors in [tsi] proposed a Telemedicine System Interoperability Architecture which contains two levels of interoperability. The first level concentrates on how to compose stations within a telemedicine system and how to deliver the functionality within the station. This level is based on three sets of interfaces: station-to-device, station-to-station and station internal interfaces. The second level concentrates on how different stations can discover

each other within a system and then begin transactions. The goal of this level is to allow independent systems to locate (discovery), negotiate (QoS parameters) and interoperate each other. Each of these levels provides a number of features to support different aspects of interoperability.

[XGL03] proposed a distributed framework of Web-based telemedicine system which addresses two types of servers, i.e. Web servers and data servers. This framework is based on CORBA technique.

[MA06] proposed a SOA-based e-Health services architecture. It consists of six main components for defining interactions among different layers. In this architecture, the consumers include the hospital, medical staff, or other e-Health enterprise applications. Support function layer is used to help consumers to discover, deploy, and invoke health services and infrastructures. The control system layer enables the high availability, reliability, fidelity, and QoS. The goal of this layer is to employ different algorithms to achieve the service duplication, fault tolerance, and load balancing. Infrastructures and services layer is a container of health services. The security system provides the access control and other security functionalities. And the management system concentrates on controlling the data flow from one layer to another.

Moreover, [MS04] proposed the general access control requirements of Health Information System (HIS). However, it is based on the small component-based HIS. There are also many other research effort have been conducted with regards to the access control requirements for the healthcare system [RE06]. Most of them concentrate on the access control of a special subsystem, especially the Electronic Patient Record (EPR) system. However, the analysis of the access control from our research is from the perspective of the e-Health system integration.

2.7 Access Control Models

This subsection will focus on related work on access control models that can be applied to e-Health portals.

The access control matrix (ACM) [W.71] is an early access control model which can be represented by a triple <subject, object, right>. The object is an entity (such as file, process etc.) that to be protected and the subject is the entity (such as user, process etc.) who wishes to access the objects. The right specifies the operations that a subject is allowed to perform on the desired object. Access control lists (ACLs) is a common way to implement the ACM. Each object is associated with an ACL to indicate the access right that each subject is authorized to perform action on the object.

In discretionary access control (DAC), accesses are assigned by the owner of objects. DAC also allows a subject to pass its access permissions to another subject. Upon acquiring the permission either directly or indirectly, this subject is authorized to access these objects.

Unlike DAC, Mandatory access control (MAC) is an access control model where the permission of access is determined by the system, instead of the owner. In MAC, accesses are based on security clearances and levels (sensitivity labeling or security labeling). If the sensitivity level associated with the subject is equal or higher than the desired object, the subject is allowed to access the object. MAC is generally used in multilevel systems such as military information system.

Sandhu et al. proposed the RBAC model in 1996 [SCFY96]. The RBAC is an access control model where permissions are assigned to roles rather than directly assigned to users (refer to Figure 2.5). Roles are defined according to different job titles or functions, such as Medical Staff, Patient, and Administrator etc. Users are assigned with roles based on their responsibilities. The authorization of RBAC is split into two independent phases, i.e. user-role assignment and role-permission assignment. Sandhu also specifies four conceptual models derived from RBAC. RBAC-0 is the basic conceptual model which contains users, roles, permissions and sessions. A user can establish a session to activate a set of the roles to which the user is assigned. RBAC-1 encloses RBAC-0 and introduces the concept of role hierarchy. Role hierarchy uses partial order to represent the inheritance between roles which can reflect the structure of an organization in the reality. It also increases the reusability of permissions by which senior roles can inherit permissions from a junior role.

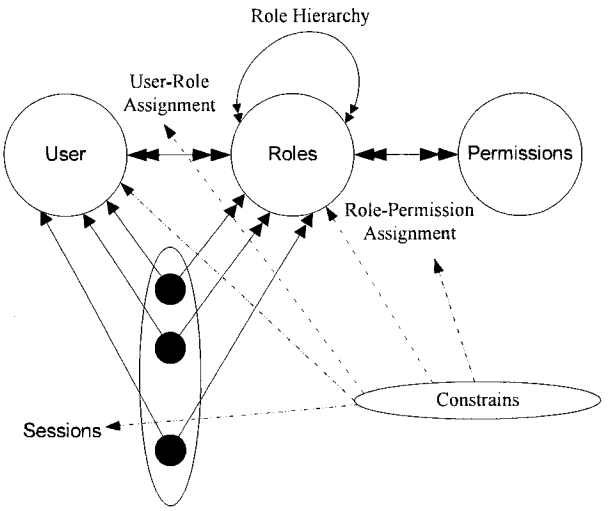

Figure 2.5: Role Based Access Control Model

RBAC-2 extends RBAC-0 with constraints, which add restrictions when assigning users or permissions to roles, or in the activation of roles in sessions. Finally, RBAC-3 combines all features provided by RBAC-1 and RBAC-2.

Besides the access control models stated above, grouping permissions can also be achieved by using rule based access control. Generally, a rule consists of two parts: Left-Hand Side (LHS) and Right-Hand Side (RHS) [FH03]. The LHS represents a set of conditions and the RHS stands for consequences. When those conditions on the LHS are true, actions (conclusions) on the RHS can be executed. Here is an example of rule:

IF LHS (Conditions) THEN RHS (Conclusions)

Thus, rule based access control model allows subjects to access various resources based on predefined rules. Rule engine is the core component of the model which is used to interpret rules at runtime. After loading the rule set from the rule repository, the rule engine will analyze conditions within rules and infer consequences. If consequences are positive, all actions will be executed. On the other hand, executions will be denied. Compared with RBAC, rules are less intuitive than roles and general non-technical administrators are incapable to specify rules.

Chapter 3

ARCHITECTURE DESIGN OF THE E-HEALTH PORTAL SYSTEMS

3.1 Functional Requirements

The design of e-Health portal aims to meet a collection of functional requirements as follows. The first three requirements are in the point of view of users of the portal, whereas the last four are about portal administrators or service providers.

- User Personalization. Users of an e-Health portal can create and save a personalized page including only the content they would like to access. For example, a patient may prefer seeing only the newsfeed in cardiology.

- Content Aggregation. Users of a portal can access related services on a single page, regardless how many service providers are involved or how different those services are implemented. Navigation elements should be provided such that users can easily switch to a different page when necessary.

- Ease of Use. This requirement is particularly relevant to e-Health systems where many users are seniors or have limited knowledge of computer technology and even the installation of client-side software may go beyond their capability.

- Backend Customization. Administrators of a portal can customize the source of services provided by the portal using a content management system, and such modifications should be transparent to normal users.

- Interoperability. The portal must be able to seamlessly integrate heterogeneous medical services implemented on different platforms and with different technologies. Such implementation details should be transparent to users of the portal.

- Extensibility. The integration of new services should imply minimal impact on the normal operation of other services provided by the same portal. Downtime should be minimized because the continuous availability of medical services may have direct impact on patients' health or life. Moreover, service providers should be able to independently produce services and seamlessly plug them into the portal.

- Support of Different Service Modes. A medical service can run in real time, automation or store-and-forward mode, and the portal should support all of above. Real time mode service is a kind of service that different participators can interact with each other at the same time. Generally, video and audio transmissions are involved in such kind of service. Automation mode means the medical data users submitted is processed and analyzed according to some sophistical algorithms automatically and the result will be returned to users. Store-and-forward mode means medical data is sent to a station where it is kept and examined at a later time by a medical staff.

3.2 System Architecture Design

We adopt the Service-Oriented Architecture (SOA) for the e-Health portal. Traditional designs of software systems usually have difficulties in meeting the aforementioned functional requirements. The interoperability among heterogeneous components requires a complicated integration process, which then implies unacceptable downtime and efforts for the integration. In contrast, the SOA exposes resources of each software component as standard-conforming services that can be accessed without understanding the underlying implementation details.

The main components of our design are e-Health portals interconnected via the Web Services for Remote Portlets (WSRP) protocol. The portals constitute an e-Health portals

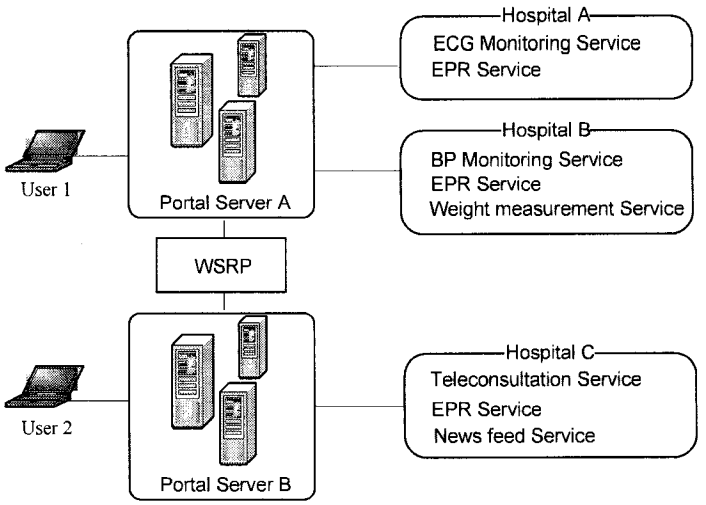

Figure 3.1: An Example Deployment of e-Health Portals Federation

federation (in the reality, portal federation can also include systems with WSRP-enabled components). Each e-Health portal is a Web-based application that provides users a unified interface to all the services provided by these medical organizations. The portal allows its users to personalize desired content through creating customized Web page, namely, portal interface. Each portal interface also plays the role of a content aggregator by including a collection of related services. Navigation elements inside each portal interface allow users to easily navigate among different collections of services just like surfing the Web. In our design, all client-side processing is supported either by the built-in functionalities of a standard Web browser or through applets/activeX controls that are automatically downloaded and executed in the browser. Users are not required to possess sophisticated computer skills in order to use services provided by the portal. Figure 3.1 is a simple deployment of such federation, which contains two e-Health portals. Portal A integrates medical services provided by Hospital A and B, and Portal B contains three services provided by Hospital C.

More details of the proposed architecture are depicted in Figure 3.2. End users interact through browser with the portal server via HTTP or HTTPs. The portal server consists of a portal engine and a portlet container. The main responsibility of the portal engine is to aggregate content from different sources and to serve these content to multiple devices. The portal engine also performs the role based access control (will be discussed in chapter 4) and

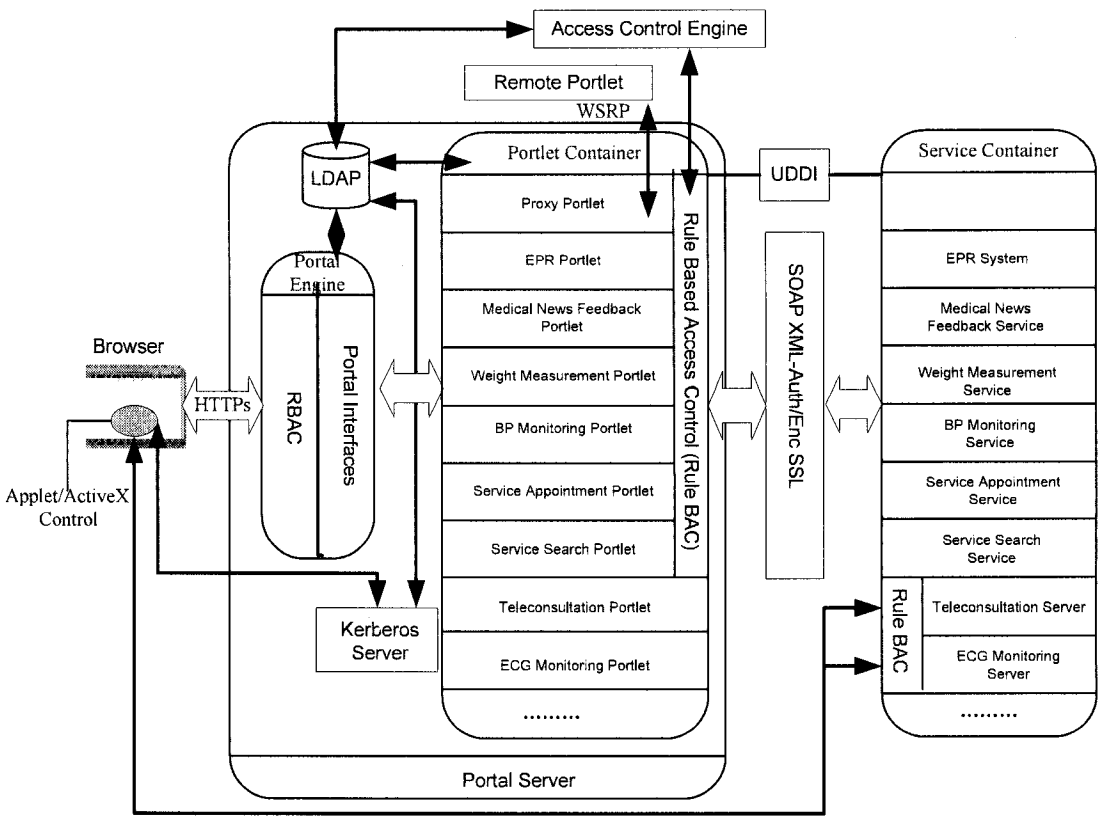

Figure 3.2: A Service-Oriented Architecture for e-Health Portal

redirects users to appropriate portal interfaces.

A portlet container provides a runtime environment for portlets implemented according to the Portlet API. In this container, portlets can be instantiated, used and finally destroyed. The separation of portal interfaces from portlets allows portal administrators to easily customize the source of services using a content management system.

Interoperability and extensibility are both inherent to the architecture because most of the backend services (run in automation or store-and-forward mode, e.g. the BP monitoring service is running in automation mode) can be integrated in a Web services manner. Service provider (backend services) and service requester (portlet) can publish and find desired medical services via UDDI, respectively. Once a new medical service is integrated into the system, it can be described by a WSDL file and this file will be sent to the UDDI. After discovering the binding information of the services via UDDI, service requester will communicate with backend services through standard SOAP regardless of what kinds of platforms are used in each side.

To access a service, a user connects to the portal server via a standard Web browser. According to the user's personalized settings and the entitlement, a portal interface is displayed with a collection of portlets inside it. When the user clicks on a button encapsulated in a portlet, the corresponding action of the service will be performed at backend service providers (which may be governed by a remote portal). As an exception, services run in real-time mode (such as the ECG Monitoring service and Teleconsultation service) are allowed to bypass the portal since they usually demand better performance than what SOAP can provide. Thus, an applet or activeX control downloaded to the browser will perform the required actions on behalf of the users. From the above descriptions, the proposed architecture clearly meets all requirements stated in Section 3.1.

There are also some other components in the architecture. Whose functionalities and features will be described in following chapters.

Chapter 4

ACCESS CONTROL IN E-HEALTH PORTALS

4.1 Overview of Access Control in e-Health Portals

4.1.1 Related Factors

The ultimate goal of access control is to decide whether an access request from a specific user is allowed or denied. There are several factors related to the access control mechanism in the e-Health portals.

- Portal domain: A portal domain contains a portal server, which integrates backend services, and a LDAP server which stores users' credentials and attributes, etc.
- User roles: Roles are created for various job functions, and users are assigned to roles based on their qualifications and responsibilities. Each portal domain has its own role structure, which can either be hierarchical or flat. Each domain may also name their role as needed.
- User: Users are the subject of access control. Each username is associated with only one portal domain. The user credential and other user attribute values are stored in this domain's LDAP server. Users use services via associated portal interface.
- Hospitals and medical organizations: Generally, these are service providers. They can either perform their own authorization or delegate all or partial authorization to the portal server. Each provider may provide its services to one or more portal servers.

- Services: Service providers may expose their services as web services or traditional application interfaces. They can either register their Web services with the UDDI or negotiate with portal administrators and technicians for business collaboration.

- Actions: Traditional access control strategies are expressed in terms of read or write accesses. It is generally applicable to file systems or database systems. However, the service components and backend applications in our environment only expose one or more interface methods and encapsulate internal details. These methods are more complex than just read/write. So, actions in our system are service provider defined operations, such as "start", "read", "update" or "submit" etc.

- Business policy: Business policy is a contract negotiated between service provider and portal mangers. It is a human readable rule to guide access control rule decision.

- Condition restrictions: Condition restrictions are generally restrictions on environment parameters such as time scope, IP address, role, server load or any other attributes.

4.1.2 Requirements

In this section, the requirements of access control for portal-based e-Health system environment is discussed.

1. In our system, services are provided by different hospitals or medical organizations (refer to Figure 3.1: Hospital A provides ECG Monitoring and EPR services, and Hospital B provides BP Monitoring, EPR and Weight measurement services). Each service encapsulates one or more actions (taking ECG Monitoring service as an example, the service interface provides patients with "Dis/connect to ECG Monitoring Server" actions) and we assume that these services are independent of each other. Access control must be able to guarantee that only authorized users can trigger the action when some conditional restrictions are satisfied.

2. Access control for e-Health environment must be dynamic, that is, it should be possible to specify and change policies at runtime depending on the environment dynamics and

changes in business policies. It is natural that service providers may change their business policies. As a result, the access control policies should be updated accordingly. The access control strategy must support dynamic policy changing without redeployment of the system.

3. Access control for e-Health environment should be able to deal with many users and many service objects. Services may be removed and new services may be integrated into the portal at any time. Moreover, the entitlement of individual or group of users may be changed. Therefore, the design for authorization, assignment of access rights, and their modifications should be easy.

4. Access control for e-Health environment must support both Browser/Server (B/S) based and Client/Server (C/S) based applications. Generally, a portal is a Web based-application (refer to Figure 3.2, the BP Monitoring, News Feedback, EPR, Weight measurement services are Web-based services. Users submit their requests from corresponding portlet and these requests will pass by the portal server). However, there are some real-time audio/video services (refer to Figure 3.2, the ECG Monitoring and Teleconference services are such services) could also be integrated into our system. The characteristic of such services is that the data exchanged between end user and real time server bypasses the portal server. Thus, the access control strategy should support both cases but in a uniform framework.

5. Access control should support active access, i.e. a subject's permission can be effective only with the right time scope, at a right location and/or under other constrains. In the context of e-Health services, some of the services need appointments in advance (In Figure 3.2, real-time ECG Monitoring service needs available medical staffs to monitor patients' ECG signal. So, patients must set up the appointment in advance). Once the appointment is approved, the user can trigger the actions of the service when corresponding conditions are met.

6. General users and service providers cannot freely change security attributes, such as the access rights of services. Changing security attributes may induce an leak of information. Only security administrators can do it (upon negotiation with service provider).

7. Access control must be enforced in a distributed manner. In a federated environment,

remote services from one or many portal servers are collected at runtime. There is no centralized authorization for access control. Thus, access control becomes more complicated. We assume that all e-Health system portals employ the same access control mechanism proposed in this thesis.

4.2 A Two-Tier Approach to Access Control

4.2.1 Problem Formulation

Even though many access control models have been proposed, they do not completely meet the requirements of the Web based e-Health portal environment. In this section, I first illustrate why these models are inappropriate for our case.

Increases in the number of subjects and objects require an increase in the cost of managing the ACL or ACM, which violates the requirement 3 stated in section 4.2.1.

DAC employs access policies determined by the owner of the resources (such as actions or services). The owner decides who is allowed to access to the resources and what privileges they have. According to our requirement 6, only the security administrator can have this privilege. Moreover, in the context of our e-Health system, we cannot decide which resource has higher security level than another. Thus, MAC is unsuitable for our system.

Many e-Health portal systems are built with off-the-shelf software components, such as BEA WebLogic used in our implementation. As a widely accepted standard, the role-based access control (RBAC) is typically a built-in feature of those softwares and can usually meet the basic security requirements of simple applications. However, such RBAC features become insufficient once applied to a complex environment like an e-Health portal (notice that the limitations I shall describe are about the implementation of RBAC, but not in the RBAC model itself [SCFY96]). An e-Health portal naturally demands access control capabilities that go beyond the limited support of RBAC found in commercial software components.

First, accesses to medical services are usually subject to conditions that depend on specific users instead of roles. For example, the real-time ECG monitoring service can only be accessed by a registered patient during his/her appointment time. Enforcing the first half

of the requirement, that is only a registered patient can use the service, is straightforward with RBAC. An administrator can create a role named "registered patient", and assign the permission to use the service to this role. Any user must thus activate the role in order to use the service. However, to enforce the second half of the requirement, that is the service can only be accessed during an appointment, is not as simple because appointment time is unique for each patient. Such a user-specific requirement can be handled by constraints in the original RBAC model [SCFY96]. However, as an add-on feature of RBAC, constraints receive only limited support in many built-in RBAC modules. For example, WebLogic only allows constraints to be specified for roles. In order to accommodate a user-specific constraint, such as a patient's unique appointment time, a separate role may have to be created for every user, which leads to too many roles and essentially diminishes the value of RBAC.

Second, some medical services may be intended for public accesses by any user with only a few exceptions. That is, they are controlled by an open policy where accesses are by default allowed unless explicitly denied. The negative permissions required for denying accesses are supported in the RBAC model through constraints [San95]. However, the implementation of RBAC in all commercial or freeware portal servers only supports positive permissions. Moreover, the co-existence of both positive and negative permissions raises some subtle issues like conflict resolution. The support for RBAC constraints found in many commercial software components is typically insufficient for dealing with such issues.

4.2.2 Solution Overview

To remove the limitation of existing RBAC modules, we adopt a two-tier approach as shown in Figure 4.1. We leverage the existing RBAC capability and supplement it with a rule-based access control module. The RBAC tier provides coarser-grained control by assigning users to roles and roles to portal interfaces (that is, collections of portlets). The rule-based access control tier provides finer-grained control through user-specific conditions expressed in logic rules. Informally, the RBAC tier determines whether a user can see a requested service, whereas the rule-based access control tier determines whether the user can use the

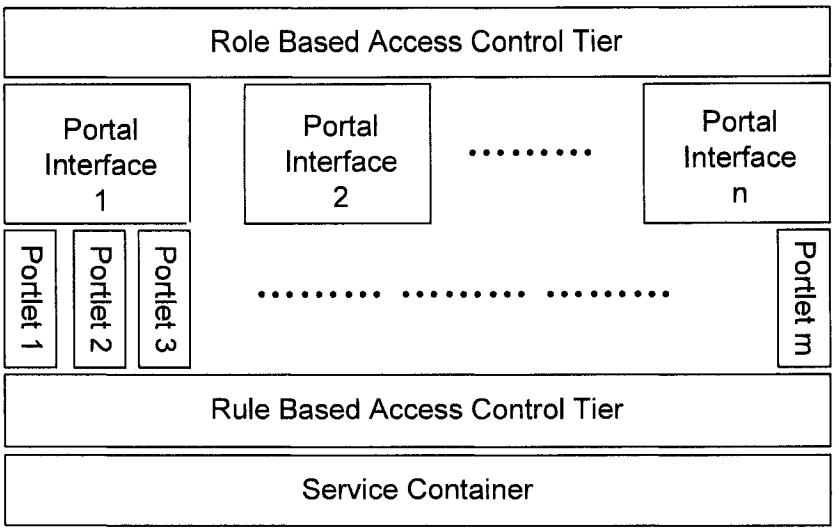

Figure 4.1: Two-Tier Access Control

service in a specific way. To trigger an action, such as starting a service, a user must first activate an appropriate role, so RBAC will redirect the user to a portal interface containing the requested portlet (service). After the user clicks on an action button on the portlet, the rule-based access control engine verifies the legitimacy of this request against applicable rules. Potential conflicts between positive and negative permissions are resolved based on predefined meta-policies, as we shall discuss in more details shortly.

This design choice of using a two-tier model has advantages over other alternatives. First, we can certainly stick to the RBAC model and supplement the existing module with full-fledged support for constraints. However, this is possible only if the existing RBAC module is fully extensible, which is usually not the case with commercial softwares. Moreover, the administration of access control rules is usually less intuitive than with roles. Completely discarding RBAC may thus render access control less manageable. Third, we can let the portal server to simply pass the responsibility of access control to backend services. However, this diminishes the very advantage of using a portal server, such as the central control of accesses.

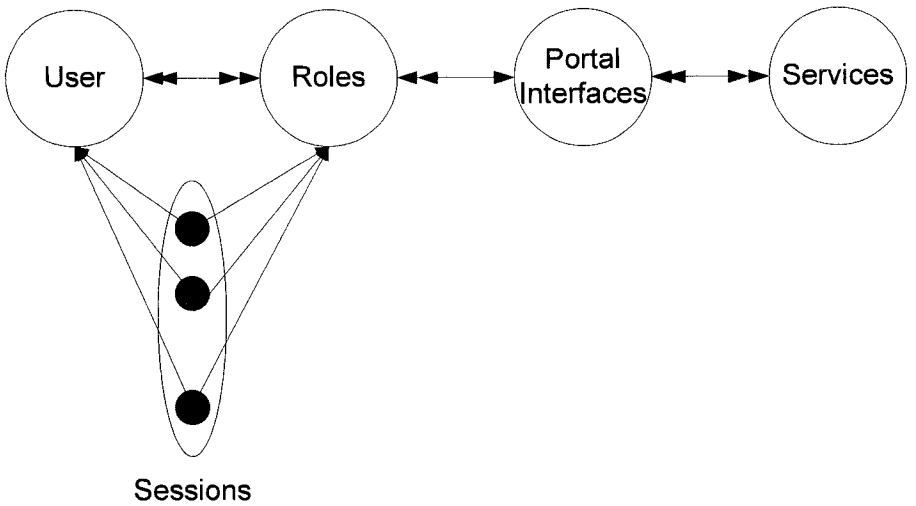

Figure 4.2: V-RBAC for e-Health System Portal

4.2.3 RBAC Tier

Once a user logins the portal, the system will redirect the user from the login page to a portal interface according to the role assigned and activated to the user.

This is a variation of traditional RBAC model (V-RBAC, Figure 4.2), in which we replace the permission with portal interface and services, and extend the Role-Permission relationship with Role-Portal and Portal-Service relationships. It is convenient for administrator to:

1. User-Role assignment. If a user possesses multiple roles which are assigned to different portal interfaces, the user must make a decision which interface s/he wishes to be redirected. In other word, for each user, only one role can be activated in a session.

2. Portal Interface-Role assignment.

3. Service-Portal Interface assignment.

In the RBAC tier, each role is assigned with one portal interface and each portal interface contains one or more portlets. We give the formal definition of the V-RBAC model as follows.

The model of this tier has the following components:

- U, R, PI, SE and S (users, roles, portal interface, service and sessions, respectively);

- $PIA \subseteq PI \times R$, a one-to-many portal interface-to-role assignment relation. Each portal interface can be assigned to multiple roles. However, each role can be only assigned one portal interface.

- $UA \subseteq U \times R$, a many-to-many user-to-role assignment relation. Each user can be entitled multiple roles and each role can be assigned to multiple users. However, each user can activate at most one role at a time.

- $SEA \subseteq SE \times PI$, a many-to-many service-to-portal interface assignment relation. Each service can be installed on multiple portal interfaces and each portal interfaces may contain multiple services.

- User: $S \rightarrow U$, mapping each session Si to an individual user: $user(Si)$ (the session's lifetime is a constant);

- Roles : $S \rightarrow 2^R$, mapping each session Si to a set of roles: $roles(Si) \subseteq \{r \mid \exists r^{'} \leq r[(user(Si), r^{'}) \in UA]\}$(this mapping can change with time) and session Si has the permission to access the portal interface $pi \mid (\exists r^{"} \leq r)[(pi, r^{"}) \in PIA]$.

4.2.4 Complementing RBAC with Rules

Once the user is redirected to an appropriate portal interface, s/he could use the services provided on that portal interface. According to requirement 5, some of the services need to make appointment in advance, such as the real-time ECG monitoring service (there must be an available cardiologist). Once the appointment is approved by the coordinator, the user's requesting this service will be considered as valid upon some conditional constrain. However, there are also many other kinds of services that do not need to make appointments in advance but must meet some conditional constraints (such as Medical News Feed service). Moreover, each service encapsulates a set of actions. In order to simplify and automate the administration of access control by allowing the security administrator to specify what kind of individual or/and combinational condition parameters that have to be met for triggering the actions, we associate a set of rules with each action. It is convenient for security administrator

to specify rules if new services are integrated into the system or if business policies change. Once a portlet or a backend application receiving a user's request, the access control proxy within the portlet will send a request (a triple <object, subject, action>) to the access control engine, which will evaluate the rules associated with the action that the user wants to perform. If the evaluation result is positive, the user can trigger this action. Otherwise, the request will be denied.

Many languages can be used to express the security policy or rule specification. Logic-based languages has been widely applied in the field of security policy specification [DBSL02].

To express fine-grained access control requirements as rules, we apply the classical Flexible Authorization Framework (FAF) [JSSS01] to our design of e-Health portals. FAF employs stratified first-order logic to represent access control rules, which gives it a well-defined semantics. User-specific conditions can be represented as a single rule with variables, which will be instantiated on the fly for each request. Logic inferences based on pre-defined meta-policies can easily handle conflict resolution between positive and negative permissions. We extend FAF rules by defining application-specific predicates and meta-policies that are especially suitable for e-Health portals.

We restate the formal definition of some concepts, notions and predicates used in this thesis.

Definition 1: (Hierarchy). A hierarchy is a triple $(X, Y, \leq)$ where:

(1) X and Y are disjoint sets

(2) $\leq$ is a partial order on $(X \cup Y)$ such that each $x \in X$ is a minimal element of $(X \cup Y)$; an element $x \in X$ is said to be minimal iff there are no elements below it in the hierarchy, that is iff $\forall y \in (X \cup Y) : y \leq x \Rightarrow y = x$. [JSSS01]

A hierarchy is used to capture the structure of data items, users/groups, and roles. In our system, we capture the roles as a hierarchy $RH = (\phi, R, \leq_R)$ where R is a set of roles and $x \leq_R y$ if x is a specialization of y. A role is a specialization of another role if it refers to more specialized activities. For example, Specialist, GP, Nurse can be seen as a specialization of Medical-Staff. Similarly, we also capture the portal interface items in a hierarchy. Each

portal interface contains one or more portlets. Moreover, each portlet represents the interface of a backend service. We use a triple $PPH = (P_t, P_i, \leq_{PiPt})$ to capture this feature where P_t is a set of portlets. P_i is a set of portal interfaces and $\leq_{PiPt}$ is a partial ordering such that for any portlet $x \in Pt$ and portal interface $y \in Pi, x \leq_{PiPt} y$ iff x is deployed on y.

Definition 2: (Authorization). An authorization is a triple of the form $(o, s, < sign > a)$ where $o \in AO$ (Authorization Objects), $s \in AS$ (Authorization Subjects), $a \in SA$ (Signed Actions) and "sign" is either "+" or "-". [JSSS01]

Subject "s" is used to identify a user, a service consumer, a running process etc. Each subject assumes the identity and the privileges of a single principal. In our system, the subject refers to a user. Object "o" refers to a service. Action "a" refers to an operation that a subject performs on an object, such as Read, Start, Appoint, etc. In our system, each service encapsulates one or more actions. For each action, we may define a set of rules to restrict the access to trigger the action.

In order to represent the rules, we first introduce some fixed predicates defined in ASL [JSSS01], and then define our application-specific predicates.

- Predicate 1: $cando(o, s, +/-a)$

  The first argument of "cando" is an authorization object term, the second one is an authorization subject term, and the last one is a signed action term. This predicate represents the accesses that the security administrator wishes to allow or deny (depending on the sign associated with the action).

- Predicate 2: $dercando(o, s, +/-a)$

  Each argument of "dercando" has the same meaning as in the predicate "cando". This predicate represents authorizations derived by the system using logical rules of inference. It can be used to express different kinds of implication relationships between authorizations.

- Predicate 3: $do(o, s, +/-a)$

This predicate also has the same argument as the "cando" and "dercando". This predicate is different from the above two predicates, it represents the accesses must be granted or denied. Intuitively, "do" enforces the conflict resolution and access decision policy.

ASL also applies two hierarchical predicates.

- Hierarchical predicate 1: $in(x, y, H)$

  This predicate contains three arguments. The first two belong to elements of AO∪AS and the third one is a ground term equal to either PPH or RH. It represents the ordering relationships in the PPH and in the RH hierarchies.

- Hierarchical predicate 2: $dirin(x, y, H)$

  This predicate has the same argument as "in". However, it represents the direct membership relationships in the PPH and in the RH hierarchies.

Conditional predicates capture the possible conditional parameters that may need to be taken into account by the access control system.

- Conditional predicate 1: $isRole(s, role)$

  The first argument of "isRole" is an authorization subject term (it refers to a user in our system) and the second one is a type term. This predicate represents whether the subject "s" is entitled with "role". We treat role as an attribute of a user.

- Conditional predicate 2: $typeof(o, service - type)$

  The first argument of "typeof" is an authorization object term (it refers to a service instance in our system) and the second one is a type term. This predicate represents whether the object "o" belongs to the specified service type.

- Conditional predicate 3: $isLeader(s, o)$

  The first argument of "isLeader" is an authorization subject term (it refers to a user in our system) and the second one is an authorization object term. This predicate

represents whether the subject "s" is the leader of the service instance "o". Generally, this predicate is used in the teleconference scenario where a leader is needed to control the conference.

- Conditional predicate 4: $status(o, status)$

  The first argument of "status" is an authorization object term (it refers to a service instance in our system) and the second one is a type term. This predicate represents whether the object "o" is in the specified "status". Generally, this predicate represents the status of appointment-based service instance, such as real-time ECG monitoring etc. These statuses could be one of the following: unapproved(U), approved(A), running(R), finish(F) or canceled(C).

- Conditional predicate 5: $hasRegisteredService(s, o)$

  The first argument of "hasRegisteredService" is an authorization subject term (it refers to a user in our system) and the second one is an authorization object term. This predicate represents whether the subject "s" has already registered with the service object "o".

- Conditional predicate 6: $timeValid(c, b, e)$

  The first argument of "timeValid" is a time/date term (it refers to the current time/date of the server). The second and third ones are also time/date term, which represents the upper and lower limit of the time scope. This predicate represents whether the current time "c" is in the time scope between appointed beginning time "b" and appointed ending time "e".

- Conditional predicate 7: $serviceMode(o, mode)$

  The first argument of "serviceMode" is an authorization object term (it refers to a service instance in our system) and the second one is a type term. This predicate represents whether the service object runs the in mode of "m", such as: real-time, store and forward, or automated.

Based on specific requirements, we may define other conditional predicates in the access control system.

Definition 3 (Authorization Rule): An authorization rule is a rule of the form:

cando(o, s, <sign>a) ← L1..... Ln

where "o", "s" and "a" are elements of AO, AS and SA respectively, n≥0, <sign> is either "+" or "-", and L1..... Ln are hierarchical and conditional predicates. [JSSS01]

Definition 4 (Derivation Rule): A derivation rule is a rule of the form:

dercando(o, s, <sign>a) ← L1..... Ln

where"o", "s" and "a" are elements of AO, AS and SA respectively, <sign> is either "+" or "-", and L1..... Ln are either cando, dercando, done, hierarchical or conditional predicates. All literals appearing in the body of a derivation rule must be positive. [JSSS01]

Definition 5 (Decision Rule): A decision rule is a rule of the form:

do(o, s, <sign>a) ← Ll..... Ln

where Ll..... Ln are cando, dercando, and hierarchical and condition predicates. A decision rule definitely decides whether a subject can be allowed, or denied to do the action of the object. Intuitively, "do" rules model access decisions of the system on the user requests to access objects. [JSSS01]

4.2.5 Conflicts and Inconsistency

When the portal server integrates a new service, the administrator should discuss with the service provider and create a set of rules for each action according to the business policies. However, this manual procedure may introduce problems, such as under specification (no rule specified), over specification (conflicts occurred when both positive and negative rules are specified for an action). In order to solve these potential problems, we employ meta-policies. From the point of view of authorization specification, we identify three main categories of meta-polices [JSSS01]:

- Closed: Only positive authorizations can be specified. A user is authorized to access only if s/he has been granted a positive authorization.

- Open: Only negative authorizations can be specified. A user is authorized to access only if s/he has not been granted a negative authorization.

- Hybrid: Both positive and negative authorizations can be specified. However, in such cases, conflicts may arise due to the coexistence of positive and negative authorizations. The access control system must decide which of the authorizations should be allowed.

Conflict resolution rules make the design for resolving conflicts. The following are possible solutions:

- No conflicts allowed: Coexistence of both positive and negative authorization is considered as inconsistent and therefore it is not accepted.

- Permissions take precedence: The access is authorized if both positive and negative authorization coexist. In such cases, the positive authorization takes precedence over the negative one.

- Denials take precedence: The access is denied if both positive and negative authorization coexist. In such cases, the negative authorization takes precedence over the negative one.

Next, we give examples of using the extended ASL to specify rule sets and to resolve potential rules conflicts.

Scenario 1 (User-Specific Condition) Real time ECG monitoring can be started by users who have appointed with the service, with the appointment request approved, and with the start time of the service within the appointed time range. We use a closed policy as follows.

$$cando(o,\ s,\ +start) \leftarrow typeof(o,\ ECG-Monitoring), hasAppointed(s,\ o),$$
$$status(o,\ approved), timeValid(o,\ begin,\ end),$$
$$serviceMode(o,\ real-time)$$

$$do(o,\ s,\ +start) \leftarrow cando(o,\ s,\ +start)$$

$$do(o,\ s,\ -start) \longleftarrow\!\!\!\!\!\!\neg\ do(o,\ s,\ +start)$$

The first rule states that a service "o" can be started by a user "s" if "o" is an ECG-Monitoring service, "o" has been appointed for the user and approved (by a coordinator), the service mode is real-time, and the current server time is between the appointed time scope. The next two rules state that a user's request for starting the service will be granted if he/she has a positive authorization (given by the first rule), and he/she will be denied if such a positive authorization is absent. That is, a closed policy is enforced. Clearly, this single rule is enough for regulating accesses to the service by any number of users. The rule will be instantiated by each user upon his/her request at run time.

Scenario 2 (Conflict Resolution) A medical survey is conducted to investigate the relationship between blood pressure and living locations in Canada with the participants age between 40 and 60 except those who live in Yukon. We specify both positive and negative permissions and a meta-policy for denials-take-precedence:

$$cando(Survey - LS,\ s,\ +submit) \leftarrow ageBetween(s,\ 40,\ 60),$$
$$CountryLocation(s,\ Canada)$$
$$cando(Survey - LS,\ s,\ -submit) \leftarrow CountryLocation(s,\ Canada),$$
$$ProvinceLocation(s,\ Yukon)$$
$$do(Survey - LS,\ s,\ +submit) \leftarrow cando(Survey - LS,\ s,\ +submit),$$
$$\neg\ cando(Survey - LS,\ s,\ -submit)$$
$$do(Survey - LS,\ s,\ -submit) \longleftarrow\!\!\!\neg\ do(Survey - LS,\ s,\ +submit)$$

The first rule states that the people who live in Canada and whose ages are between 40 and 60 are authorized to complete the survey. The second rule states that those who live in Yukon, Canada are not allowed participating in the survey. Conflicts will arise for those who satisfy both rules. To resolve such conflicts, the next two rules together enforce the denials taking precedence meta-policy.

Scenario 3 (Business Rule Changing) Medical magazine is a service that provides up-to-date professional medical articles. According to the business policies, medical magazines can be read by any authorized user unless explicitly denied. In such cases, we use the open policy (only negative cando can be specified) since all users entitled with a non-anonymous

role can read the medical magazine, thus we do not need to specify the positive rule for each authorized role. Open policy means if no negative policy is specified, then the access is allowed.

$cando(Med - Mag,\ s,\ -read) \leftarrow isRole(s,\ anonymous)$

$do(Med - Mag,\ s,\ +read) \leftarrow\!\!\!\rightarrow cando(Med - Mag,\ s,\ -read)$

$do(Med - Mag,\ s,\ -read) \leftarrow\!\!\!\rightarrow do(Med - Mag,\ s,\ +read)$

Suppose latter on the business policy is changed to be that the coordinator cannot read the medical magazine. Then the administrator adds a rule to the rule set:

$cando(Med - Mag,\ s,\ -read) \leftarrow isRole(s,\ coordinator)$

The above example shows that if we do not use the meta-policy (open policy), an administrator may define a rule set by mistake as:

$cando(Med - Mag,\ s,\ +read) \leftarrow isRole(s,\ administrator)$

$cando(Med - Mag,\ s,\ -read) \leftarrow isRole(s,\ coordinator)$

A user may be entitled to both "administrator" role and "coordinator" role at the same time. Thus, both positive and negative authorizations will be issued. So, conflict occurs. By using the open policy, the positive cando rule cannot be specified.

The two-tier approach inherits the advantage of both RBAC (to ease the administration of access control) and rule-based access control (such as the support for conflict resolution). However, we need to consider potential inconsistency that may arise between the two tiers. For example, a user's role may allow him/her to see a portal interface but some rules disallow his/her requests for using a service on that portal interface. This situation is quite normal (the reason of the denial may be, say, the absence of an appointment). On the other hand, a user may be allowed to use a particular service by the rule-based access control but at the same time he/she is denied for accesses to the portal interface containing the corresponding portlet (service). This second situation is not desirable because the user will not be able to use a service if he/she cannot even see it.

To model the phenomenon that a user must be able to see a portal interface before using any of the services contained by the portal interface, we define a partially ordered set (poset)

$< P \cup A \cup I, \ \leq >$ where P denotes the set of portlets, A the set of actions on portlets, and I the set of portal interfaces. The relation $\leq \subseteq \ P \times A \times I$ is defined so that holds for each portal interface i, every portlet p on i, and every action a on p. That is, any action on a portlet dominates portlet interfaces that contain the portlet. We incorporate this partial order on authorization objects into our system through enforcing a consistency checking as follows. For each positive permission on any action a on a portlet p, at least one of the portal interfaces i dominated by (containing) the portlet p must be accessible according to current RBAC policies. The condition is enforced upon adding a positive permission in the rule-based access control tier and upon removing a user-role or role-permission assignment in the RBAC tier (an inconsistency between the two tiers can arise in those cases).

For example, there is a service "SE" that can be accessed by nurse. The security administrator may specify a set of rules like:

$cando(o, \ s, \ +read) \leftarrow isRole(s, \ nurse), typeof(o, \ SE)$

$do(o, \ s, \ +read) \leftarrow cando(o, \ s, \ +read)$

$do(o, \ s, \ -read) \leftarrow\!\!\!\rightarrow do(o, \ s, \ +read)$

However, the portlet of this service is not deployed on the nurse's portal interface. The result is that nurse can use this service but s/he cannot see the portlet.

To illustrate the rule inconsistency problem, we first describe the procedure to specify rule set for services:

Step 1. The User-Role and Role-Portal Interface assignments have been specified in advance and the assignment of Role-Portal Interface is seldom changed.

Step 2. The administrator places the portlet which encapsulates the service interface on the appropriate portal interface. This step is called Portal Interface-Service assignment.

Step 3. The security administrator discusses the business policies with the business provider.

Step 4. The security administrator selects the appropriate meta-policies for the rule set.

Step 5. Finally, the security administrator makes the rule set according to the business policies.

In order to avoid the inconsistency between these two tiers, we propose a method that translates V-RBAC policies into rule sets. When importing new rules or changing existing rules in the Rule BAC tier, not only the conflicts of rule sets within the Rule BAC tier will be checked, the inconsistency between translated rule sets and rule sets within the Rule BAC tier will be checked, too. We give the method for translating the RBAC policies into rules and for avoiding the inconsistency.

In the V-RBAC tier, the security administrator can specify policies like: The user who is entitled to role R can access the portal interface PI. The following expression is the translation of the RBAC policy to Rule-BAC rules.

$cando(o,\ s,\ +access) \leftarrow in(s,\ R,\ RH), in(o,\ PI,\ PPH)$

The translated rule must be a positive rule. According to the business policy, the security administrator may specify multiple rules as the rule above. We give some examples as follow:

$cando(o,\ s,\ +access) \leftarrow in(s,\ patient,\ RH), in(o,\ patient - pi,\ PPH)$

$cando(o,\ s,\ +access) \leftarrow in(s,\ nurse,\ RH), in(o,\ mstaff - pi,\ PPH)$

$cando(o,\ s,\ +access) \leftarrow in(s,\ administrator,\ RH), in(o,\ admin - pi,\ PPH)$

To avoid the inconsistency, our solutions are illustrated as follow:

1). The head of the authorization rule is positive.

Recall that in Figure 3.2, each portal interface is assigned to one or more roles and each portal interface encapsulates one or more services. Thus, when specifying authorization rules with positive head, if the rule body contains "isRole" or "typeof" predicates, the role specified in the "isRole" predicate must be allowed to access the portal interface that contains the service specified in the "typeof" predicate. And this service must be in the hierarchy associated with the portal interface. In other word, the permission to see the portal interface is the precondition of actions on the service deployed on that portal interface.

2). The head of the authorization rule is negative.

There is no inconsistency in such cases because a user is allowed to access a portal interface does not imply s/he can take the action on the service deployed on the portal interface.

3). The head of the authorization rule is hybrid.

If the security administrator specifies the hybrid policy for the rule set, only the positive ones should be checked.

4.3 Inter-Portal Access Control

We have illustrated the architecture of the e-Health portal systems in chapter 3 and have briefly introduced the e-Health portals federation. The two-tier access control approach we proposed in the previous section is generally based on the system which contains only a single e-Health portal. When multiple e-Health portals interoperate with each other, as an e-Health portal federation, some problems may arise. In this section, firstly, we introduce the concept of a portal federation, and then we discuss the user authentication issues in our system. Finally, we demonstrate how to apply the two-tier access control approach to the federated e-Health portals.

4.3.1 Portal Federation

A federated portal is a portal that includes remotely distributed resources. As shown in Figure 4.3, a consumer is a Web application that collects remote portlets and offers them in a unified portal to end users who use a browser to view and interact with the portal. Typically, a consumer does not include the business logic, data, or user interface parts of a portlet. It simply collects user interface markup delivered from producers and presents that user interface to users. A producer is also a Web application, typically running on a remote system from the consumer. The producer acts as a container for portlets that are offered to consumer portals. The producer is where the user interface, data, and business logic for remote portlets reside. While a consumer is administration centric, a producer is application centric.

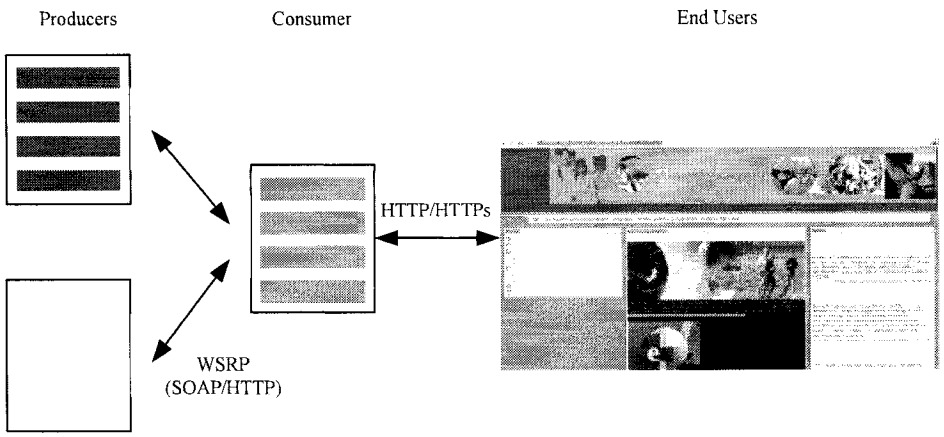

Figure 4.3: Federated Portal

4.3.2 User Authentication in e-Health Portals

Authentication is an important feature of access control. It is the process of determining whether the user is who he/she claims to be. The authentication is performed before authorization. The characteristics of Web-based and client-server-based authentication are different, so we treat them differently. Portal is a Web-based application. User employs a browser, such as Microsoft IE, to access the system. Considering that the users of our system generally do not possess much knowledge of computer security, using IPSec (hard to configure for both client and server side) and Client certificate (user may not know how to use the certificate, and the use of certificate may bring extra cost) is impractical. The form-based username/password authentication combined with one way SSL can provide mutual authentication, and ensure data confidentiality and integrity at the same time. Therefore, we choose such a common solution for Web-based authentication.

Another issue is authentication in the federated environment. In the federated portal environment, we build trust relationship among portals. When users access the remote portlet via WSRP, they do not need to authenticate themselves to the remote portal. In other words, the producer always trusts that the request from the consumer is authenticated.

For the authentication of client-side downloadable applications, we compare potential solutions in table 4.1. In contrast to other solutions, Kerberos can provide mutual authentication, is transparent to users, and supports cross domain authentication, which is a very

Solutions	Advantages	Disadvantages
Password based	1. Ease of use 2. Low performance overhead	1. Plaintext password transmission 2. Users have to input the password by themselves 3. Multimedia server authenticating the user may bring security issues 4. Server cannot be authenticated to users
IPSec	1. Transparent to users 2. Suitable for real-time video application authentication	Hard to configure for both client and server side
Two-way SSL	Mutual authentication	1. Users may not know how to obtain and use certificate 2. Extra cost for the certificate
Kerberos	1. Mutual authentication 2. No password transmitted between client and server 3. Transparent to users 4. Cross domain (Inter-realm) authentication	1. Greater overhead on the client 2. Kerberos is designed for use with single-user client systems 3. Require clock synchronization

Table 4.1: Comparison of Authentication Solutions

important feature for distributed environments. Another reason is that delegation of authentication to multimedia servers may introduce more security risks since the server will need to know too many details about each portal domain's authentication mechanism, which is inflexible. We integrate the Kerberos client-side code into an applet and setup a Kerberos server for each portal domain.

Figure 4.4 is a scenario for real-time ECG telemonitoring service authentication using Kerberos. Patient and cardiologist belong to different portal domains. They use the ECG telemonitoring service provided by hospital A to perform real-time ECG telemonitoring. The patient logins the portal server A. The browser will download the applet from the portal server A automatically. Then, the applet will exchange multiple messages with Kerberos Server (Realm A) for mutual authentication and finally get the service granting ticket. Like

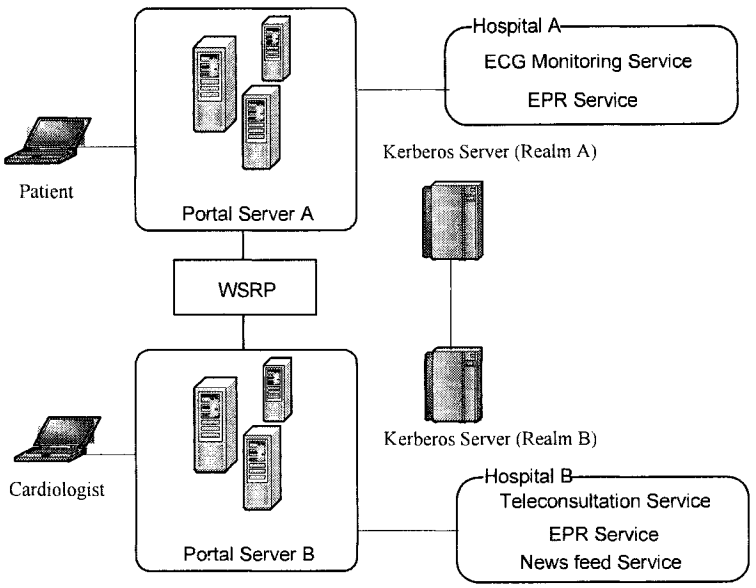

Figure 4.4: Cross-Domain Authentication with Kerberos

the patient, the cardiologist will exchange multiple messages with Kerberos Server (Real B) to get the ticket for remote TGS (Realm A). The cardiologist then uses this remote TGS ticket to get the cross-domain client-server ticket from Kerberos Server (Realm A). After that, both patient and cardiologist can use the client-server ticket to access the service.

4.3.3 Analysis of the Inter-Portal Authorization

An e-Health system portal federation integrates existing, possibly distributed, heterogeneous e-Health portal servers. We assume that every e-Health portal server employs the logic-based rules to check whether for a given request, either a permission or a prohibition can be inferred. This approach is straight forward in centralized systems or distributed systems with a centralized control. In order to apply the two-tier access control model to the e-Health system portal federation, we must also consider the following issues.

The consumer-side portal handles RBAC for users and redirects users to an appropriate portal interface upon successful logins. When a user sends a service request from the consumer-side portlet to the producer-side portlet, either the consumer-side or the producer-side can handle the rule-based access control (in the following statement, we assume the producer side handles the rule-based access control). However, some issues may arise in such

a case.

1. Predicate attribute naming inconsistency

To illustrate this issue, firstly we give an example. User1 that belongs to the Portal A (consumer) wants to access the news feed service provided by the Portal B (producer). We assume that the rule for "read" action is:

$cando(News - Feed,\ \ s,\ \ +read) \leftarrow\ isRole(s,\ \ patient)$

In order to evaluate the rule, Portal B must know which role User1 is entitled to. However, the role assignment for User1 in the Portal A is "sickman", which cannot be understood by Portal B. We term such issue as attribute naming inconsistency. Such an issue can also arise when both consumer and provider sides have different time/date formats, different service status naming, or other discrepancies. Thus, we classify the attributes into two types, i.e. consumer attribute and provider attribute. Provider attribute and consumer attribute mean that predicate attributes can be directly acquired and understood by the provider side and consumer side, respectively. However, each portal domain is autonomous, which results in that some attributes cannot be directly acquired or are even incomprehensible. There are two possible solutions.

The first solution is to specify a rule for each potential consumer and name the attributes according to the naming criterion used in the consumer-side portal. When evaluating the rule set, attributes should be transmitted from the consumer side to the provider side. Following the above example, we should specify the rule as follows.

$cando(News - Feed,\ \ s,\ \ +read) \leftarrow\ isRole(s,\ \ PortalA : sickman)$

where "PortalA" means a potential consumer and "sickman" is a role name at the Portal A (consumer) side.

The second solution is to employ a centralized server who acts as an attribute mapper to translate the attribute naming from one domain to another. The drawbacks of this solution are that it is inflexible and it requires to design a dedicate interface to send various attributes values. It also has synchronization issues when any attribute mapping is changed at one side.

Common to the above two solutions is that different portals administrators should agree

on consistent attribute naming/format mapping schema in advance.

2. Restricted attribute

For example, User1 in Portal A wants to access a gynecology service provided by Portal B. We assume that the access rule for the "start" action of this service in Portal B is:

$$cando(o,\ s,\ +start) \leftarrow isRole(s,\ patient), typeof(o,\ gynecology),$$
$$gender(s,\ female)$$

However, suppose due to privacy reason, portal A is not allowed to provide the gender information of the user to portal B. We term such issue as restricted attribute. In such a case, the provider side cannot finish the access control evaluation process.

4.3.4 Collaborative Inter-Portal Access Control

To solve the above issues, we split the authorization rule into two parts and use two new predicates, i.e. partial authorization predicates $pcando(o, s, +/-a)$(provider-side partial authorization predicates) and $ccando(o, s, +/-a)$(consumer-side partial authorization predicates) to specify the two parts, respectively. We put these two partial authorization predicates in both sides. The following shows an example of ECG telemonitoring service,

$$cando(o,\ s,\ +start) \leftarrow isRole(s,\ specialist), serviceMode(o,\ real-time)$$
$$typeof(o,\ ECG-Monitoring), hasRegisteredService(o,\ s),$$
$$status(o,\ approved), timeValid(current,\ begin,\ end),$$

where "specialist" in the "isRole" predicate is at the consumer side. This value cannot be acquired directly from the producer side. Values specified in the rest of the application-specified predicates can be directly acquired from the producer side. Thus, the partial authorization rules can be written as:

$$ccando(o,\ s,\ +start) \leftarrow isRole(s,\ specialist)$$
$$pcando(o,\ s,\ +start) \leftarrow typeof(o,\ ECG-Monitoring),$$
$$hasRegisteredService(o,\ s), status(o,\ approved),$$
$$timeValid(current,\ begin,\ end), serviceMode(o,\ real-time)$$

The above example shows the case when positive authorization rule is specified. Both *ccando* and *pcando* predicates must be positive. For negative authorization rules, we require

that *ccando* and *pcando* must both be negative. For the case of hybrid rules, we treat positive and negative rules separately.

The next step is to make final authorization decision. It can be made either at the consumer side or the provider side.

1. Authorization decision is made at the consumer side.

In such a case, the user's request will be sent from the consumer side to the provider side and the provider side evaluates the partial rule, i.e. *pcando*. Then the service will be invoked and provider will return the consumer side with both the result of *pcando* and the result of the service request. When the consumer side receives these two, it will evaluate the *ccando* with the result of *pcando* and derive the combined *cando* and finally make a decision with meta-policies. If the decision is positive, then returns the service result to the user, otherwise returns error message.

2. Authorization decision is made at the provider side.

In such a case, the consumer side will evaluate the *ccando* and forward the result and the user's request to the provider side. Upon receiving the request, the provider will evaluate the *pcando* with the result of *ccando* and derive the combined result for *cando* and finally make a decision with meta-policies. If the decision is positive, the service will be invoked and the service result will be returned to the consumer side and then passed to the user. Otherwise, the error message is returned.

Chapter 5

IMPLEMENTATION

This chapter describes the implementation details of a prototype of the e-Health system portal. The system is designed to integrate existing medical systems, application, and services. We also illustrate the design and implementation of the access control engine and some medical services.

5.1 System Deployment Platform

The overall system architecture is described in chapter 3. Recall that in Figure 3.1, our prototype system contains two portal servers, that is, portal A and portal B. Portal A integrates services provided by hospital A and B, and portal B integrates services provided by hospital C. All hospitals provide the EPR service pointing to a centralized EPR system.

Our portal server utilizes the BEA Weblogic Portal 8.1, which provides enterprise portal infrastructure for streamlined portal development. This framework includes a graphical environment for developing portals, as well as browser-based assembly tools for business experts. It also simplifies the production and management of customized portals, allowing us to leverage a shared service environment to incorporate changes with minimal complexity and efforts [bea]. Figure 5.1 shows the homepage of the patient portal interface.

5.2 Medical Services

Our research group has designed and implemented ECG Telemonitoring service, Blood Pressure (BP) Monitoring service, Teleconference service, and EPR system. I was responsible

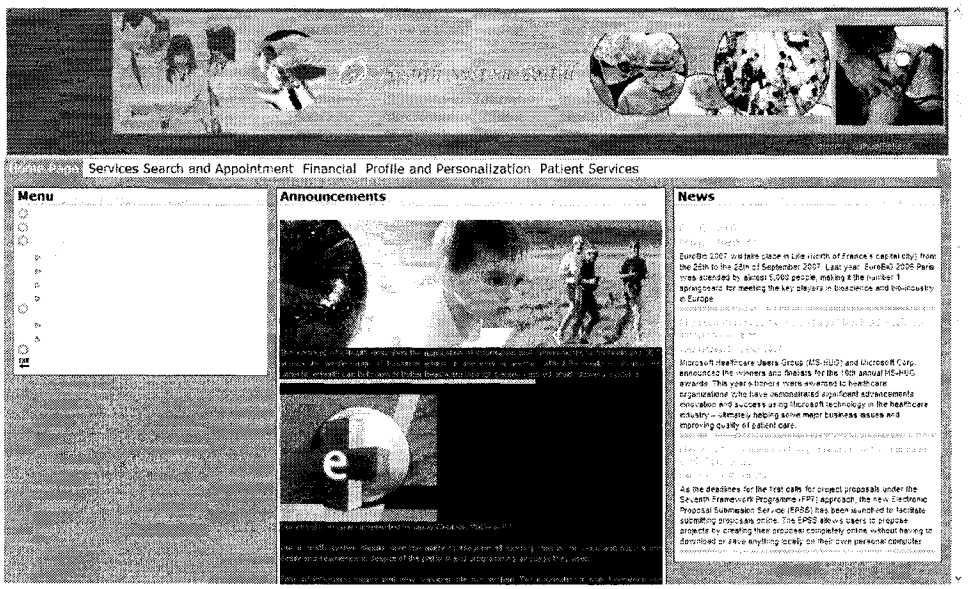

Figure 5.1: Patient Side ECG Monitoring Interface

for the implementation of the ECG Telemonitoring service and BP Monitoring service.

5.2.1 ECG Monitoring Service

In our implementation, the ECG monitoring service is in real-time mode. For the automated mode, sophisticated algorithms will be needed to analyze the ECG signals, which is out of the scope of this research. The ECG monitoring service contains three components: a patient-side applet, a medical staff-side applet, and an ECG monitoring server. This patient-side applet is based on the toolkit ECGSYN [ecg], which is a realistic ECG waveform generator in the PhysioNet toolkits. It can generate a synthesized ECG signal with user-settable mean heart rate, number of beats, etc. PhysioNet is an Internet resource for biomedical research and development sponsored by the NIH's National Center for Research Resources. In real world, the ECG signals should be gathered from the external medical devices which are wired (such as USB, RS-232 etc) or wireless (WiFi, Bluetooth, IR etc.), and are connected to a computer. For demonstration purposes, we use a ECG signal generator to simulate this procedure. We also simplify the implementation of the toolkit and add our own features, such as signal transmission, Kerberos functions, etc. The ECG monitoring server is a java-based application that acts as a repeater to forward signals to the medical staff side and

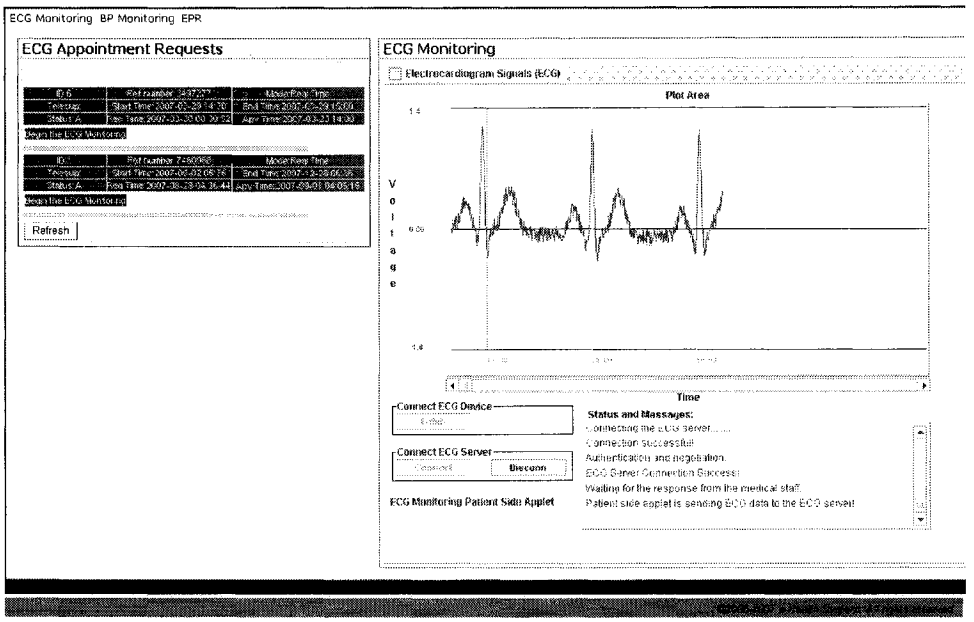

Figure 5.2: Patient Side ECG Monitoring Interface

stores the ECG signal data into a database. Figure 5.2 and Figure 5.3 show the snapshots of the patient-side and medical staff-side ECG monitoring interface respectively. When a patient wants to use the ECG monitoring service, s/he needs to simply click a desired appointment in the ECG appointment request panel, and the applet will be downloaded and run in the patient's local machine. The patient can then perform the monitoring via the applet interface.

5.2.2 BP Monitoring Service

The architecture of the BP monitoring service is shown in Figure 5.4. The BP monitoring service is implemented in Java. It exposes its functionality as Web services. In the portal server side, the BP monitoring portlet acts as a service requester on behalf of the user. Unlike traditional servlet or JSP, portlet is a container of servlet, JSP or Java Page Flow (JPF). In our implementation, the JPF, a special Java file that uses a JPF file extension, the nerve center of the portlet, is employed to separate the user interface code from navigational control and business logic. User interface codes are placed in a set of JSP files, which render users BP values input forms, diagnostic result, or any error information. Navigational control is

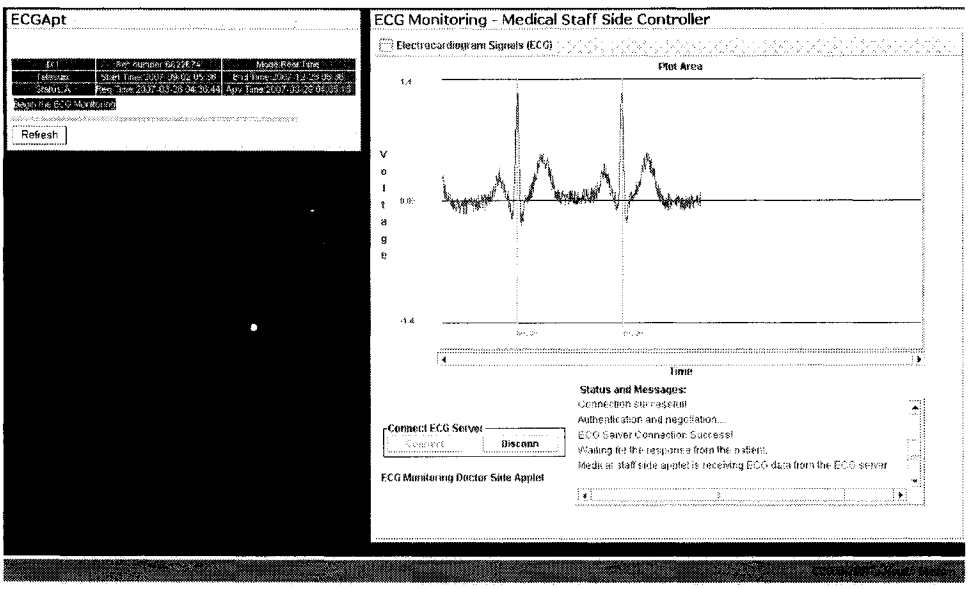

Figure 5.3: Medical Staff Side ECG Monitoring Interface

also implemented in a page flow's single controller file. It utilizes a special JSP tag to invoke an action with a hyperlink. When the link on a JSP file is clicked, the page flow runtime detects the action and runs the navigational action method and controls user navigation between those JSP pages. Business logic can be implemented either in the page controller file, Java controls, or proxies called from the JPF file. In the implementation, business logic is implemented as a stand alone service and is wrapped as a Web service. The service proxy located in the JPF helps the portlet to locate and communicate with this service. The service is also for demonstration purposes. The medical criteria data is collected according to the US National Library of Medicine [bp1] and American Medical Association Report [bp2].

5.3 Two-Tier Access Control

The access control engine is a dedicated service that performs rule-based access control for users' requests. It contains three components, as shown in Figure 5.5. The core of the engine is an evaluator, which acts as a reasoning system to validate the rule set. This component is written in Prolog and runs in the Prolog server (as shown in Figure 5.6). Prolog is a simple but powerful programming language developed at the University of Marseille [Rou75]. It is

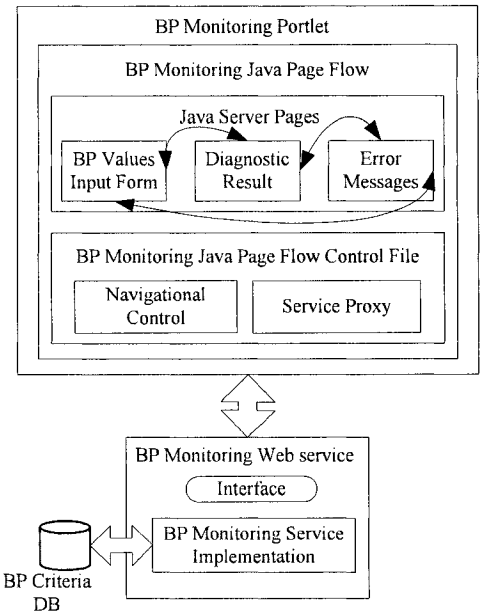

Figure 5.4: BP Monitoring Service Implementation

also a practical tool for logic programming. By using Prolog, we can write clear, readable, concise, and error-free programs. There are many Prolog products on the market, which includes open source (such as SWI-Prolog [swi], YAP-Prolog [yap]) and commercial ones (such as SICStus Prolog [sic]). As a leading product, SICStus Prolog is a state-of-the-art, ISO standard compliant Prolog development system [sic]. It is efficient and robust for handling large amounts of data and large applications. It supports bi-directional interfaces to C & C++, .NET and Java (in the current implementation, SICStus 4.0.1, the callback is not supported for .NET and Java). It also supports multiple platforms (such as Windows, Linux, Mac etc.). Thus, the SICStus Prolog is used to implement the core of the engine.

Another important component of the engine is the predicates API, which implements all required user-specific predicates used in our system. It is also responsible for collecting data, such as Current Server Time, Server Load, Appointment Information, User Profile, etc., to instantiate the variables in each predicate. This component is written in C++.

The third component is a Java application that sends request to and receives response from the Prolog server and exposes the validation interface as a Web service which can be invoked by access control proxies resided in applications. This component can also be

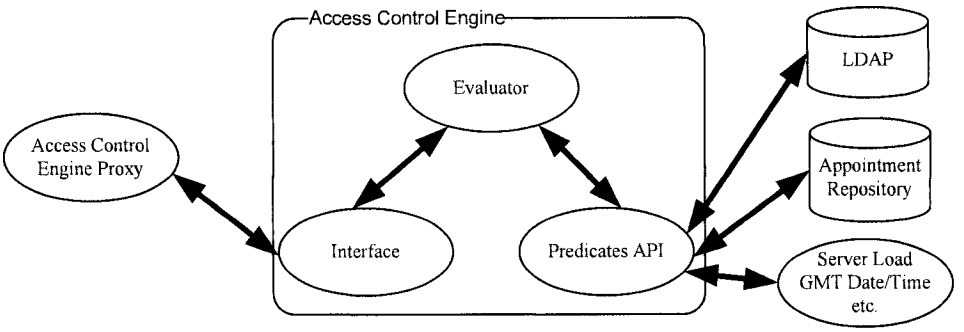

Figure 5.5: Components of the Access Control Engine

implemented using other languages supported by SICStus, such as .NET, C or C++.

Figure 5.7 is the interactions among the components involved in Rule BAC procedure, as described below.

1. The access control proxy sends the authorization request to the access control engine.

2. Upon receiving the request, the interface component forwards it to the validator (Prolog server).

3. According to the "action" tag in the request, corresponding rule set will be activated. Each rule within the rule set contains one or more predicates. The evaluator will invoke each predicate implementation via predicates API.

4-5. The predicates API may collect necessary data from databases or environment parameters to instantiate variables within each predicate.

6. The predicates API returns to the evaluator with "True" or "False" after executing the implementation of the predicate.

This procedure may occur multiple times until predicates API returns all predicates executing result (as shown in 7-10).

11. After collecting all the results from the predicates API, the evaluator will perform the inference procedure to obtain authorization decision and return the decision to the interface component.

12. The interface component returns this decision back to the access control proxy, and the proxy will enforce access control according to the decision.

Hereafter, we illustrate in details how the access control engine is configured, compiled

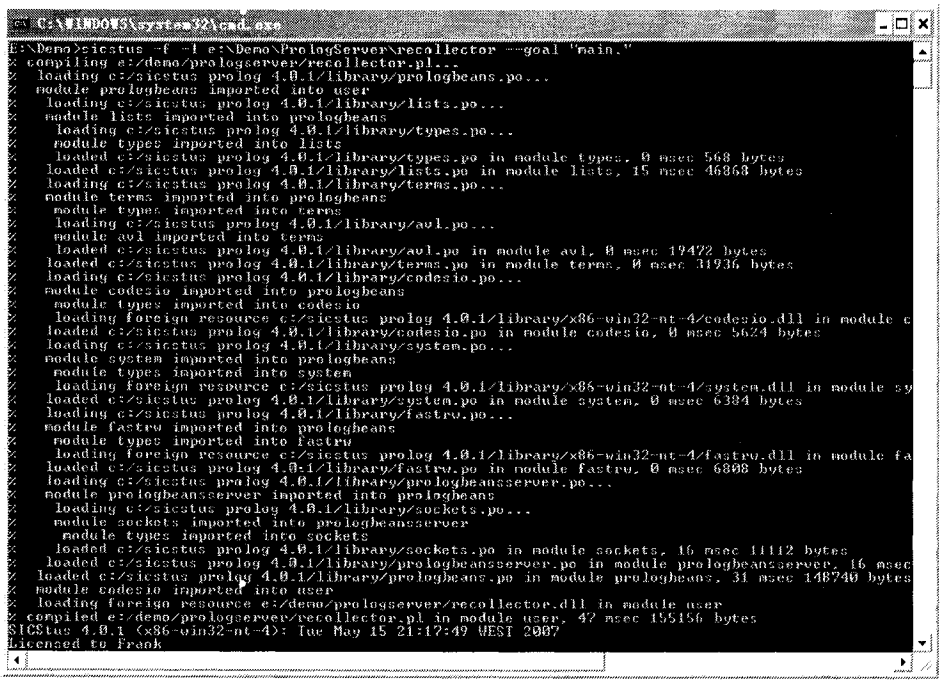

Figure 5.6: Snapshot of the Prolog Server

and linked under Windows XP SP2.

1. Software and System requirements: JDK 1.5, Microsoft Visual Studio 2005 Team Suite with SP1, SICStus Prolog 4.0.1, MySQL 5.0.

2. MySQL Server in its original configuration. In the current implementation, the MySQL database is used to store all necessary information, such as Appointment Information, User Profile, etc.

3. Under the Windows environment, when compiling the database (MySQL 5.0) operation interface "*.cpp" file, we must include the following "*.h" files and must obey the order:

 # include "myglobal.h"

 # include "mysql.h"

Moreover, we must also include "*_glue.h" to "*.cpp" file where "*" is identical to the "*.cpp" prefix. This glue file will be generated by the Prolog compiler when compiling the prolog program.

4. Visual Studio C++ 6.0 is not supported by SICStus 4.0.1 anymore since its compiler

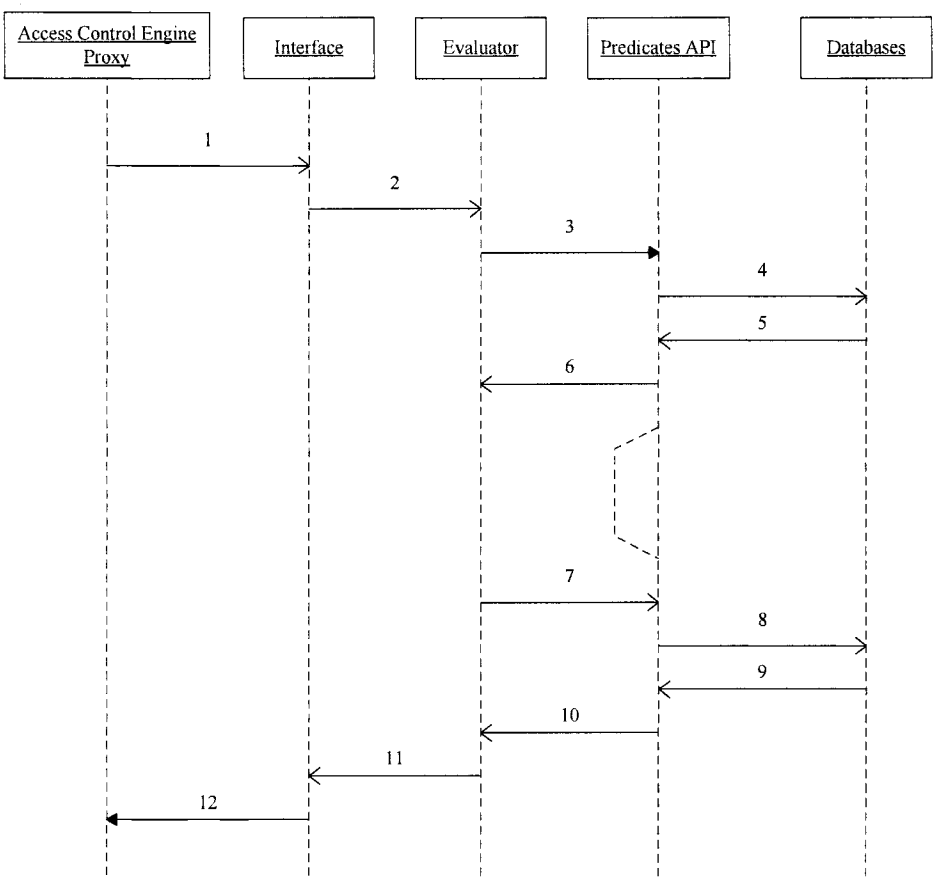

Figure 5.7: Sequence Diagram for Access Control Engine

does not support some options that SICStus needed. Thus, as recommended, we choose the Visual Studio C++ 2005 (VSC2005) as the default C++ program compiler. In order for the SICStus prolog compiler and linker to locate the "cl.exe" of the VSC2005, the Windows system environment variables must be set properly. Here is an example setting:

PATH = C:\Program Files\Microsoft Visual Studio 8\VC\bin

5. Configuration of SICStus 4.0.1. The default configuration file of SICStus is "spconfig-4.0.1". Since the default setting for the VSC2005 linker sets the flag "SAFESEH" on, some libraries used by MySQL is incompatible with the "SAFESEH" linker flag and this flag is completely optional. Thus, we turn "SAFESEH" off by modifying the configuration file by simply removing all occurrences of the "SAFESEH" flag. The following are two occurrences of such flag:

WIN32_SPLD_CC_SPECIAL=/link /NXCOMPAT /SAFESEH

SPLFR_SHLDFLAGS=-nologo -dll /INCREMENTAL:NO /SAFESEH /NXCOMPAT

...

Moreover, we should also set the path of the MySQL library as (where the path of the library may be different according to the installation location):

SPLD_EXE_LIBS=D:\MySQL\lib\opt\libmysql.lib kernel32.lib user32.lib...

6. Setting the environment variables of JDK 1.5.

7. Compiling the project. In order to run the program in an environment that does not have VSC2005, we must set the Manifest file for the project. Copy all "*.dll" and manifest files from the "...Microsoft Visual Studio 8\VC\redist\x86 \Microsoft.VC80.CRT" folder to the project folder where contains the "*.cpp" and "*.pl" files. Then rename the original "Microsoft.VC80.CRT.manifest" to "*.dll.manifest" where "*" should be identical to the name of the "cpp" file in step 3. Then execute the following command to compile the "cpp" and "pl" file:

splfr -verbose -keep recollector.pl recollector.cpp

Then a "*.dll" file will be generated. The foreign resource will be imported by the Prolog program automatically when the Prolog program is running.

8. Running the access control engine. Executing the following command to start the Prolog server:

%sicstus -f -l recollector -goal "main."

After that, a Java application will communicate with the Prolog server and the interface of this Java application will be exposed as Web service, and can be invoked by any access control engine proxy deployed in any applications.

Notice that in this access control engine, many application-specific predicates are defined. These predicates are implemented using C++ language (Predicates API). The return type of these predicates is boolean. Thus, if a predicate is approved (or not), true (or false) will be returned to the Prolog program. In Prolog, a rule is constructed by a rule head and a body and the body contains one or more predicates and each predicate has a prefix operator (positive or negative). If the logical operation of all predicates with their prefix operators are approved then the head, i.e the goal will be approved. Otherwise, the goal cannot be achieved. Each business policy can be represented as a rule set, which contains multiple rules and meta-policies. The Java applicaiton hides the rule validation details and only exposes one interface with few parameters. When the access control proxy wants to validate user's request, it only needs to provide the "object", the "subject" and the "action" values.

5.4 Use Cases

In this section, two use cases will be given to demonstrate how access control works together with e-Health services. The first is a Web service-based BP monitoring scenario, and the second is an applet-based ECG monitoring scenario.

5.4.1 Web Service-based BP Monitoring Scenario

After the patient successfully logs in to the system, the portal engine will redirect the user to the patient portal interface. By clicking the navigation bar on the portal interface, the interface of the BP monitoring service will be rendered to the patient. This interface contains two components, i.e. BP monitoring request history panel and the service panel. When the

patient clicks the desired request hyperlink in the request history panel, the service panel will be activated to allow the patient to input the systolic and diastolic values (In reality, these data should also be gathered directly from the BP measurement device installed on patients' body). Figure 5.8 is the sequence diagram for BP monitoring service. The procedure of how the messages interact among components within the portal and backend services is shown as follows.

1. After the patient fills in the systolic and diastolic values and clicks the "Submit BP Values" button, this request will be forwarded to the BP Monitoring Portlet.

2. Upon receiving this request, the portlet will get the necessary parameters from the request, assemble a new authorization request, and send it to the access control engine.

3. According to the rule set defined for the "Submit BP Values" action, the access control engine will evaluate this request, decide whether to allow the user to access, and then return the decision back to the BP monitoring portlet.

4. If the authorization is negative, the portlet will show an error message to the patient. Otherwise, it will contact the user profile database and fetch the gender and age information for future usage.

5. The user profile database returns the gender and age information of the patient to the BP monitoring portlet.

6. The BP Monitoring portlet assembles the BP values with gender and age information as parameters, and invokes the service via SOAP.

7. The BP monitoring service will process this request and return the result to the portlet.

8. Return the result to the patient.

5.4.2 Applet-based ECG Monitoring Scenario

Like the BP monitoring service, the ECG monitoring service user interface also contains two panels, i.e. a request history panel and a service panel. When the patient clicks the desired request hyperlink, the applet will be downloaded into the service panel, and all the parameters used by this applet will be initialized. After that, the user can use the service

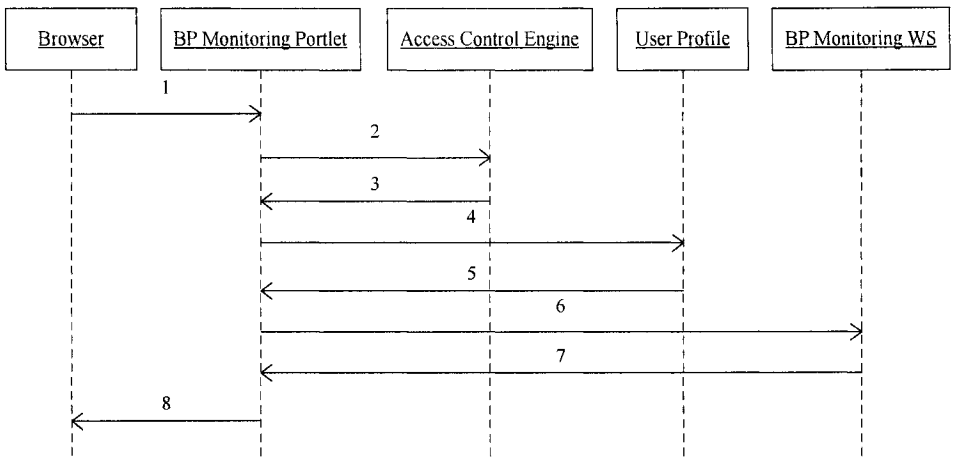

Figure 5.8: Sequence Diagram for BP Monitoring Service

via the applet. Figure 5.9 shows the sequence diagram on behalf of the patient.

1. The ECG applet authenticates itself to the Kerberos server in term of the patient.

2. The Kerberos server returns the client-to-server ticket to the ECG applet. In step 1 and 2, we ignore multiple messages exchanged between the ECG applet and the Kerberos server (for more details for how to get the client-to-server ticket, refer to chapter 2).

3. ECG applet uses the client-to-server ticket to authenticate itself and sends a request to establish the connection for ECG signal transmission.

4. If the authentication is successful, the ECG monitoring server will send an authorization request to the access control engine to confirm that this user is allowed to use this service.

5. The access control engine returns the authorization decision to the ECG monitoring server.

6. If the authorization is positive, the ECG monitoring server will accept the connection to the applet. At the same time, the ECG monitoring server will check whether the desired medical staff has connected to the server. If yes, the server will inform the patient-side applet to send data.

7. Upon receiving the instruction, the patient-side applet will transmit ECG signal to the ECG monitoring server.

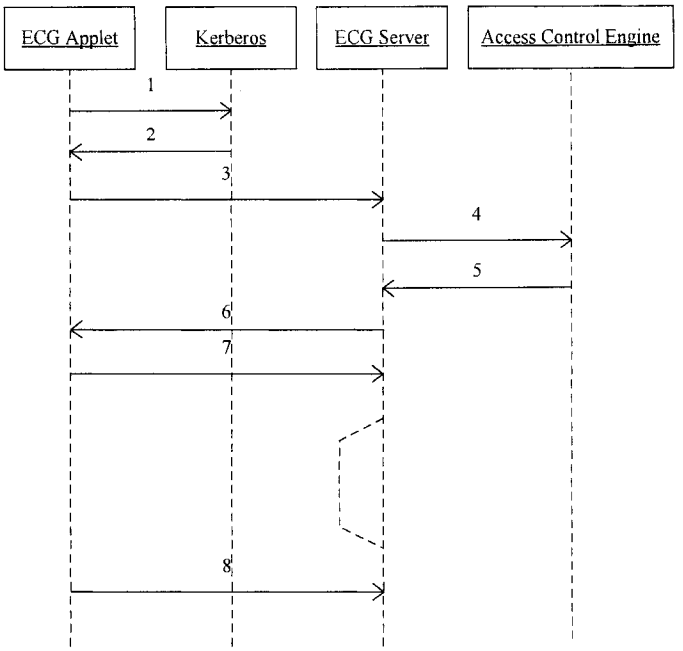

Figure 5.9: Sequence Diagram for Real Time ECG Monitoring Service on Behalf of Patients

Figure 5.10 shows the sequence diagram in term of the medical staff.

1. The medical staff-side ECG applet authenticates itself to the Kerberos server on behalf of the medical staff.

2. The Kerberos server returns the client-to-server ticket to the ECG applet.

3. ECG applet uses the client-to-server ticket to authenticate itself and sends a request to establish the connection for ECG signal transmission.

4. If the authentication is successful, the ECG monitoring server will send an authorization request to the access control engine to confirm that the medical staff is allowed to use this service.

5. The access control engine returns the authorization decision to the ECG monitoring server.

6. If the authorization is positive, the ECG monitoring server will accept the connection to the applet. At the same time, the ECG monitoring server will check whether the desired patient has connected to the server. If yes, the server will inform the medical staff-side applet to prepare receiving data.

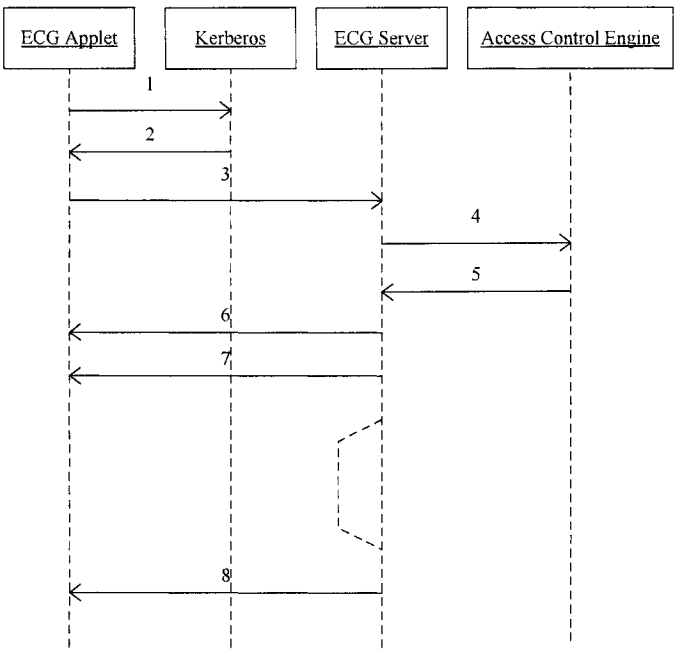

Figure 5.10: Sequence Diagram for Real Time ECG Monitoring Service on Behalf of Medical Staffs

7. Upon receiving the order, the medical staff-side applet will prepare receiving ECG signal from the ECG monitoring server in sequence and display the data to the medical staff.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Traditional designs of software systems failed to meet the requirements of our system. We thus based our design upon a service-oriented architecture that can satisfy the stated functional requirements. Our e-Health portals can integrate different medical services and applications. In our prototype system, real-time ECG monitoring service, BP monitoring service, EPR system and Teleconsultation service have been implemented and integrated into our e-Health portal.

We also pointed out limitations found in the access control module of many off-the-shelf software components. Our solution was based on a two-tier access control architecture that integrated existing RBAC modules with a rule-based access control extension. This design inherited the advantages of both models and was cost-efficient. We also indicated and proposed solutions for the inconsistency issue within the two-tier model, the predicate attribute naming inconsistent issue, and the restricted attribute issue in applying our model to a federation environment.

6.2 Future Work

The proposed architecture and access control mechanism have been formulated in the thesis. Future researches are listed below:

1. Extend rule-based access control engine to enforce authorization of workflow-based services.

2. Quality of Services (QoS) is an important factor to be considered. QoS includes the scalability and performance of Web service-based services, the performance of the access control engine, the delay, jitter, response time etc., of real-time services etc. In the future work, we will investigate those QoS issues.

3. The WSRP standard does not address any security standard currently. Thus, for the authentication of consumer, the authentication of end user to the producer, message integrity and confidentiality should be enforced in a standard way.

Bibliography

[bea] http://edocs.bea.com/wlp/docs81/index.html.

[Bis05] G. Bisson. e-Health Portal and SNOMED for a More Personalized Integrated EHR. *Proc. UM2005 Workshop on Personalization for e-Health*, 2005.

[bp1] http://www.nlm.nih.gov/medlineplus/ency/article/003398.htm.

[bp2] http://www.ama-assn.org/ama1/pub/upload/mm/38/a-06csaph.pdf.

[CG04] Yuechun Chu and Aura Ganz. A mobile teletrauma system using 3G networks. *IEEE Transactions on Information Technology in Biomedicine*, 8(4), 2004.

[CRCM97] W.J. Chimiak, R.O. Rainer, J.M. Chimiak, and R. Martinez. An Architecture for Naval Telemedicine. *IEEE Transactions On Information Technology In Biomedicine*, 1(1), 1997.

[DBSL02] Nicodemos Damianou, Arosha K. Bandara, Morris Sloman, and Emil C. Lupu. A Survey of Policy Specification Approaches. 2002.

[E05] Jovanov E. Wireless Technology and System Integration in Body Area Networks for m-Health Applications. *Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference Shanghai, China*, 2005.

[ecg] http://www.physionet.org/physiotools/ecgsyn/java/ecgsyn-java.html.

[Eys01] G. Eysenbach. What Is e-Health? *Journal of Medical Internet Research*, 3(2):e20, 2001.

[FGT+03] J. Fayn, C. Ghedira, D. Telisson, H. Atoui, J. Placide, L. Simon-Chautemps, P. Chevalier, and P. Rubel. Towards new integrated information and communication infrastructures in e-health: Examples from cardiology. *Computers in Cardiology*, 2003.

[FH03] E. Friedman-Hill. Jess in action. rule-based systems in java. *Manning Publications, Greenwich (USA)*, 2003.

[FKM01] Beltrame F., Boddy K., and P. Maryni. Adopting telemedicine services in the airline framework. *IEEE Transactions on Information Technology in Biomedicine*, 5(2):171C174, 2001.

[GKS02] Aniruddha Gokhale, Bharat Kumar, and Arnaud Sahuguet. Reinventing the wheel? corba vs. web services. *The 11th International World Wide Web Conference*, 2002.

[HLL+07] Yuan Hong, Shuo Lu, Qian Liu, Lingyu Wang, and Rachida Dssouli. A hierarchical approach to the specification of privacy preferences. *Proc. 4th International Conference on Innovations in Information Technology (Innovations 2007), IEEE*, 2007.

[IJZ04] R.S.H. Istepanian, E. Jovanov, and Y.T. Zhang. Guest Editorial Introduction to the Special Section on M-Health: Beyond Seamless Mobility and Global Wireless Health-Care Connectivity. *IEEE Transactions On Information Technology In Biomedicine*, 8(4), 2004.

[jsr] http://jcp.org/en/jsr/detail?id=168.

[JSSS01] S. Jajodia, P. Samarati, M. Sapino, and V. Subrahmanian. Flexible Support for Multiple Access Control Policie. *ACM Transactions on Database Systems (TODS)*, 26(2):214–260, 2001.

[KG06] G. Kaur and N. Gupta. E-Health: A New Perspective on Global Health. *Journal of Evolution and Technology*, 15(1):23–35, 2006.

[Koc05] C. Koch. A new blueprint for the enterprise. *CIO Magazine*, 2005.

[KPK+01] E. Kyriacou, S. Pavlopoulos, D. Koutsouris, A. S. Andreou, and C. Pattichis. Multipurpose health care telemedicine system, Engineering in Medicine and Biology Society. *Proceedings of the 23rd Annual International Conference of the IEEE*, 4:3544–3547, 2001.

[LHL+07] Shuo Lu, Yuan Hong, Qian Liu, Lingyu Wang, and Rachida Dssouli. Access control for e-health system portal. *Proc. 4th International Conference on Innovations in Information Technology (Innovations 2007), IEEE*, 2007.

[LLH+08] Qian Liu, Shuo Lu, Yuan Hong, Lingyu Wang, and Rachida Dssouli. Securing telehealth applications in a web-based e-health portal. *Proc. 3rd International Conference on Availability, Reliability and Security (ARES 2008), IEEE*, 2008.

[MA06] Omar W. M. and Bendiab A.T. E-Health Support Services Based on Service-Oriented Architecture. *IEEE Computer Society*, 2006.

[mit] http://web.mit.edu/kerberos/.

[MM04] Brennan D. M. and Barker L. M. An interactive telemedicine system for remote speech-language pathology treatment. *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, 2004.

[MS04] Evered M. and Bogeholz S. A Case Study in Access Control Requirements for a Health Information System. *Australasian Information Security Workshop 2004 (AISW 2004), Dunedin, New Zealand. Conferences in Research and Practice in Information Technology*, 32, 2004.

[NZ] D. Wijesekera N. Zannone, S. Jajodia. Creating objects in the flexible authorization.

[oas] http://www.oasis-open.org/.

[PKV+02] C.S. Pattichis, E. Kyriacou, S. Voskarides, M.S. Pattichis, R. Istepanian, and C.N. Schizas. Wireless Telemedicine Systems: An Overview. *IEEE Antenna's and Propagation Magazine*, 44(2), 2002.

[plu] http://portals.apache.org/pluto/.

[RCW+98] C.J. Riedel, T.F. Choudhri, D. Wilson, N. Khanafer, A. Alaoui, W. Tohme, and S.K. Mun. Telemedicine in neurosurgery: peri-operative management. *Proceedings of Medical Technology Symposium, IEEE*, pages 80–82, 1998.

[RE06] L. Rostad and O Edsberg. A Study of Access Control Requirements for Healthcare Systems Based on Audit Trails from Access Logs. *Computer Security Applications Conference, ACSAC '06, IEEE*, 2006.

[Rou75] P. Roussel. Prolog, manuel de rfrence et d'utilisation. *Groupe Intelligence Artificielle, Facult des Sciences de Luminy, Universit Aix-Marseille II*, 1975.

[RS99] Scherrer J. R. and Spahni S. Healthcare Information System Architecture (HISA) and its Middleware Models. 1999.

[sam] SAMTA Project, http://samta.offis.de/.

[San95] R.S. Sandhu. Issues in RBAC. *ACM Workshop on Role-Based Access Control*, 1995.

[SCFY96] R. S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.

[sic] www.sics.se/sicstus/.

[SRK+04] S. Shin, C. Ryu, J. Kang, S. Nam, Y. Song, T. Lim, J. Lee, D. Park, S. Kim, and Y. Kim. Realization of an e-Health System to Perceive Emergency Situations. *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, 2004.

[swi] http://www.swi-prolog.org/.

[tsi] The Telemedicine System Interoperability Architecture, http://telemedicine.sandia.gov/.

[VPI+01] S. Voskarides., C.S. Pattichis., R. Istepanian., E. Kyriacou., M.S. Pattichis., and C.N. Schizas. Mobile health systems: A brief overview. 2001.

[W.71] Lampson B. W. Protection. *Proceedings of the 5th Princeton Conference on Information Sciences and Systems*, page 437, 1971.

[w3c] http://www.w3c.org/.

[wet98] World Wide Emergency Telemedicine Service (WETS). *Telemat. Application Programme Project HC-4025*, 1998.

[XGL03] Yang Xiang, Qiwei Gu, and Zhengxiang Li. Computer-Based Medical Systems. *Proc. 16th IEEE Symposium*, page 108 C 113, 2003.

[yap] http://www.ncc.up.pt/ vsc/yap/.