

Design and Implementation of a Practical Secure Distributed Healthcare Application

Gan Zaobin^{1,2} and Varadharajan Vijay²

¹ College of Computer Sci. & Tech., Huazhong University of Sci. & Tech.
Wuhan 430074, PR China

² Department of Computing, Macquarie University,
Sydney, NSW 2109, Australia
{zgan, vijay}@ics.mq.edu.au

Abstract. Security plays a vital role in the design and practical deployment of distributed applications. All companies have to repeatedly spend considerable time, capital and effort on the implementation of the security mechanism for their applications, and the result is unsatisfactory as well. In this paper, we investigate an integrated security management tool - ManageSecure, present a formal description of the healthcare system requirements. and then describe how to implement the healthcare system security objectives by means of ManageSecure. The result shows that ManageSecure can greatly cut down the development schedule of applications and the cost, and the security of applications can be guaranteed as well.

1 Introduction

Today, Internet has become the forum for intra- and inter-organizational interactions. Most distributed applications are developed in the Internet environment. Securing access to such systems via Internet is vital and has become increasingly challenging. Therefore, security plays a vital role in the design and practical deployment of the distributed applications, and all companies have to repeatedly spend considerable time, capital and effort on the implementation of the security mechanism for their applications. On the one hand, the securing web-based distributed application could be complicated, time-consuming and costly. It could involve a whole host of security components and practices, such as managing user authorizations, implementing single sign-on, creating a web portal, and managing security resources in the security infrastructure. On the other hand, many organizations lack the resources or the expertise to integrate a complete security solution into their specialized products and often fail to recognize that they address only a subset of the overall web security challenge. Obviously, it is not economical and necessary for every company to design a specified security mechanism for each application. Nowadays, more and more integrated security management tools are emerging. If we can properly use these tools in our practical applications, the development schedule of applications will be greatly cut down, the cost will be reduced considerably and the security of the applications

can be guaranteed. In this paper, we present our attempt to secure the healthcare system using an integrated security management tool- ManageSecure [1].

The remainder of the paper is organized as follows. Section 2 presents a brief overview about ManageSecure. The access control requirements for a medical practice and hospital environment are described in Section 3. Section 4 discusses the implementation of the access control using ManageSecure. In Section 5, a brief discussion of the advantages of the proposed system is given. Finally, concluding remarks are presented in Section 6.

2 ManageSecure Overview

ManageSecure, developed by Business Networks International, is an integrated suite of security tools to implement a strong security perimeter around web-based applications [1]. This includes the following integrated functions 1) managing system security resources, 2) managing a public key infrastructure, 3) implementing single sign-on to web-applications enterprise-wide, 4) controlling access to web applications, and 5) monitoring web services and web servers for security. In addition, ManageSecure is designed to be highly scalable, configurable, interoperable and extendable.

As single sign-on and role-based access control have been used in our practical healthcare demo system, we briefly describe them first in the following sections.

2.1 Single sign-on

Single sign-on to web applications improves user experience, since the user does not need to remember and present multiple security credentials to different web servers and applications. When properly implemented, it can also enhance security, since a uniform security policy can be centrally defined and enforced across the enterprise. Standards such as Security Assertion Markup Language (SAML) are designed to support federated identities [2], such that users can traverse multiple access control domains without re-authenticating.

When a user logs onto a web-server that has a ManageSecure access control component (plug-in), this access control component communicates with a back-end access control server, authenticates the user and installs a SAML artifact on the browser session. Once a SAML artifact is installed as part of the browser session, subsequent accesses within the same session to other web-servers will use this artifact to retrieve authorization assertions from the original access control server that created the user session. Hence, the user will not need to authenticate multiple times to access multiple web-servers on the corporate network.

SAML artifacts can be used across domains and across multiple SAML compliant servers. Trusted access control servers across domains can consult each other on SAML authorizations, and permit access. The management interface can be used to define security policies for access to which the user has access. However, the software can be configured to use an external portal.

The access control filters can also be configured to delegate authentication functions to a remote server that acts as the SAML token issuing party. After the user is authenticated, the original web server can then consume the resulting SAML token as the relying party. This permits the centralization of authentication functions. An API (Application Programming Interface) is provided for use by applications to retrieve user identity and attributes based on the SAML tokens supplied to the applications.

2.2 Role-base Access Control

Studies by the National Institute of Standards and Technology (NIST) show that Role Based Access Control (RBAC) represents a cost-benefit ratio of 109:1. As per NIST, RBAC maps to organizational-specific structures in a way that reduces direct and indirect administrative cost and improves security [3]. The RBAC model supports definition of user roles and mapping roles to privileges in the system. Object access is defined in terms of privileges. This provides maximum flexibility in managing users and resources, and changes to user roles or object sensitivity can be handled easily compared to purely identity based access control.

ManageSecure supports two levels of attribute mapping. Each user can be mapped to a set of roles within various administrative domains. Each role and domain attribute combination can in turn be mapped to a set of privileges.

When a user attempts access to a resource, her/his privileges are determined as follows - her/his identity is first mapped to roles in various domains, and these roles and domains combinations are then mapped to her/his corresponding privileges. Each privilege is essentially a fine-grained security policy definition, specifying the web resource (URL), the authentication level required to access this resource, the client IP addresses allowed, and the time ranges that are permitted for accessing this web resource.

3 Access Control Requirements for the Healthcare System

In this section, by means of a first order language [4], we introduce a formal description of the access control requirements for a medical practice and a hospital setting given by the specification in [5].

3.1 The language

Let L be a sorted first order language with equality, with disjoint sorts for *role*, *privilege*, and *object* respectively. Assume L has the following vocabulary:

- Sort *role*: with role set R , and role variables r_1, r_2, \dots
- Sort *privilege*: with privilege set P , and privilege variables p_1, p_2, \dots
- Sort *object*: with object set O , and object variables o_1, o_2, \dots

- A ternary predicate symbol $s\text{-holds}$ which takes arguments as *role*, *privilege* and *object* respectively.
- A binary predicate symbol \in which takes arguments as a variable and a set.
- A binary predicate symbol \subseteq whose both arguments are sets.
- Logical connectives and punctuations: as usual, including equality.

For instance, a fact that a role r has a privilege p for object o is represented using a ground atom $s\text{-hold}(r, p, o)$. The set membership is represented as follows: for example, a role r_1 is a member of the role set R is represented using the formula $r_1 \in R$.

In addition, we can represent constraints among roles authorizations. For example, the rule stating that for any role and object, if the role is the owner of this object, then the role should have read and write privileges for that object. This constraint can be defined as follows:

$$\forall r, o, s\text{-holds}(r, \text{Own}, o) \wedge s\text{-holds}(r, \text{Read}, o) \wedge s\text{-holds}(r, \text{Write}, o)$$

The following example shows how the traditional matrix can be represented by a set of ground atoms.

Example 1 Consider the access matrix in Table 1, where R, W, M represent the privileges of *Read*, *Write* and *Modify* respectively. This access matrix can be represented in our formalism as the following set of atoms.

$$\{ s\text{-holds}(S_1, \text{Read}, O_1), s\text{-holds}(S_1, \text{Write}, O_1), s\text{-holds}(S_1, \text{Modify}, O_2) \\ s\text{-holds}(S_2, \text{Write}, O_2), s\text{-holds}(S_2, \text{Modify}, O_2), s\text{-holds}(S_2, \text{Read}, O_3) \}$$

Table 1. An Access Matrix Table

	O_1	O_2	O_3
S_1	R, W	W, M	
S_2		W, M	R

3.2 Access Control Requirements for a Medical Practice

A medical practice, like a miniature hospital setting, is one where a few GPs (with the aid of nurses, pathology collectors and receptionists) work under the same roof and they all share a common intranet with access to a backend patients' database. Such a medical practice may also have a specialist section, where specialists like gynaecologists and cardiologists may come in and see patients referred by these GPs at certain times of the week. In a medical practice, the roles are *Patients*, *Receptionists cum cashiers*, *Pathology collectors*, *Nurses*, *Doctors*, *Practice manager and Medical director*. These roles constitute a comprehensive list of roles present in a real-world medical practice. Each of these roles has a specific set of privileges on access to an electronic patient record.

Here, to provide a comprehensive list of access control requirements for all these roles, the electronic patient record is split into four components: *Personal*

and contact details (*Essential*), *Personal and contact essential (Not that essential)*, *Personal details (clinic details)*, *health details* [5]. The way to read each of these component Sets is: each entry is called a field within a component, the entries after - are subfields within this field and these are the sub-fields where data are entered into for this field. Here, we will not introduce these sub-fields due to the length constraint of the paper. More formally, we give definitions in the light of ManageSecure.

All Roles in a medical practice can be defined as a set R :

$R = \{ \textit{Patients, Receptionists cum Cashiers, Pathology Collectors, Nurses, Doctors, Practice Manager, Medical Director} \}$.

ALL Privileges on access to an patient record can be described as a set P :

$P = \{ \textit{Create, Read, Write, Modify} \}$.

The fields of the first component *Personal and contact details (Essential)* can be defined as a set E_1 :

$E_1 = \{ \textit{Name, Address, Date of Birth, Phone Number, Payment Method, Data and Time of this Visit} \}$.

Similarly, the fields of the second component *personal and contact essential (Not that essential)*, the fields of the third component *personal details (clinic details)*, the fields of the fourth component *health details*, can respectively be defined as a set E_2, E_3, E_4 .

$E_2 = \{ \textit>Title, Alias or preferred name, Separate Postal Address, Email address, Fax number, Occupation, Gender, Marital, Ethnicity, Country of birth, Next of kin, Employer, Family members} \}$.

$E_3 = \{ \textit{Status, Provider, Location of Provider, Procedures/treatment code, Pathology results (in/out), Radiology results (in/out), Visit history, Next appointment} \}$.

In addition, in order to describe the access control policies conveniently, we also define the two subsets of E_3 as follows:

- $E_{31} = \{ \textit{Status, Provider, Location of Provider, Pathology results (in/out), Radiology results (in/out), Next appointment} \}$.
- $E_{32} = \{ \textit{Status, Next appointment} \}$.

$E_4 = \{ \textit{Date and time of this consultation, Reasons for consultation, Consultation, History, Observations, Allergies and sensitivities, Immunisation record, Medication, Personal medical history, Pathology results, Radiology results} \}$.

In terms of the access control requirements from the real-world medical practice, we only define a few main access control policies here:

Policy 1: $\forall r \in R \wedge r = r_2 \subset s - \textit{holds}(r, \textit{Create}, E_1) \wedge s - \textit{holds}(r, \textit{write}, E_1) \wedge s - \textit{holds}(r, \textit{Modify}, E_1)$.

It shows that a receptionist can *Create* an empty patient record and have *Write, Modify* access to all fields of E_1 .

Policy 2: $\forall r \in R \wedge r = r_2 \subset s - \textit{holds}(r, \textit{Read}, E_2) \wedge s - \textit{holds}(r, \textit{write}, E_2) \wedge s - \textit{holds}(r, \textit{Modify}, E_2)$.

Policy 3: $\forall r \in R \wedge r = r_2 \subset s - \textit{holds}(r, \textit{Read}, E_3) \wedge s - \textit{holds}(r, \textit{Write}, E_{31}) \wedge s - \textit{holds}(r, \textit{Modify}, E_{32})$.

Policy 4: $\forall r \in R \wedge r = r_3 \subset s - \text{holds}(r, \text{Read}, E_1) \wedge s - \text{holds}(r, \text{Read}, E_2)$.

Policy 5: $\forall r \in R \wedge r = r_5 \subset s - \text{holds}(r, \text{Read}, E_1) \wedge s - \text{holds}(r, \text{Read}, E_2) \wedge s - \text{holds}(r, \text{Read}, E_4) \wedge s - \text{holds}(r, \text{Write}, E_4) \wedge s - \text{holds}(r, \text{Modify}, E_4)$.

3.3 Access Control Requirements for a Hospital Setting

Next, we present the access control requirements for a hospital setting. In a hospital, the roles that have direct contact with patients' records are *Patients*, *Administration officer*, *Administration head*, *Finance officer*, *Finance head*, *Registered nurse*, *Specialist nurse*, *Head nurse*, *Registrars*, *Senior Registrars*, *Consultants* and *Head of department*. In this case, because a medical practice resembles a mini hospital setting, the patient record defined for a medical practice is almost sufficient to be the patient record for a hospital, only 3 more fields in the patient record will be added. Here, we define them into the *Personal details (hospital related)* component.

All Roles in a hospital setting can be defined as a set R' :

$R' = \{ \textit{Patients}, \textit{Administration officer}, \textit{Administration head}, \textit{Finance officer}, \textit{Finance head}, \textit{Registered nurse}, \textit{Specialist nurse}, \textit{Head nurse}, \textit{Registrars}, \textit{Senior Registrars}, \textit{Consultants}, \textit{Head of department} \}$.

The fields of the *personal details (hospital related)* component can be defined as a set E_5 :

$E_5 = \{ \textit{Status}, \textit{Provider}, \textit{Location of Provider}, \textit{date of admission}, \textit{date of discharge}, \textit{Ward and bed number}, \textit{Pathology results (in/out)}, \textit{Radiology results (in/out)}, \textit{Visit history}, \textit{Next appointment} \}$.

Here, we also define the access control policies for a hospital setting as follows:

Policy 6: $\forall r \in R' \wedge r = r'_2 \subset s - \text{holds}(r, \text{Read}, E_1) \wedge s - \text{holds}(r, \text{Write}, E_1) \wedge s - \text{holds}(r, \text{Modify}, E_1)$.

$\forall r \in R' \wedge r = r'_2 \subset s - \text{holds}(r, \text{Read}, E_2) \wedge s - \text{holds}(r, \text{Write}, E_2) \wedge s - \text{holds}(r, \text{Modify}, E_2)$.

Policy 7: $\forall r \in R' \wedge r = r'_9 \subset s - \text{holds}(r, \text{Read}, E_1) \wedge s - \text{holds}(r, \text{Read}, E_2) \wedge s - \text{holds}(r, \text{Read}, E_5)$.

4 Implementation Aspects

We created the practical distributed healthcare system as a Web application, and mapped those roles into specific system users. We specified role, session, and user constraints and also a set of role-based access control policies using ManageSecure in our system. In this section, we outline the high-level architecture overview of our implementation, as shown in Figure 1, the access control server consults policy repositories to make centralized access control decisions in the practical distributed healthcare system. Access control node (ACN), installed on an IIS web server as a pluggable module, acts as a proxy to the practical distributed healthcare system. The ManageSecure plugin intercepts user access, communicates with the access control server, and handles the user identification and authentication. Once the user is logged in, a portal page is displayed. One

of the links on this portal page is the practical distributed healthcare system. When the user clicks on this link, the ManageSecure connector interacts with the practical distributed healthcare system. A ManageSecure plugin (Java class) is used from within the practical distributed healthcare system, to act as a callback into the single sign-on layer, to validate the user session against the user identity (this is to prevent spoofing attacks at the portal). ManageSecure comes with a ready to use connector and the callback Java plugin for the practical distributed healthcare system.

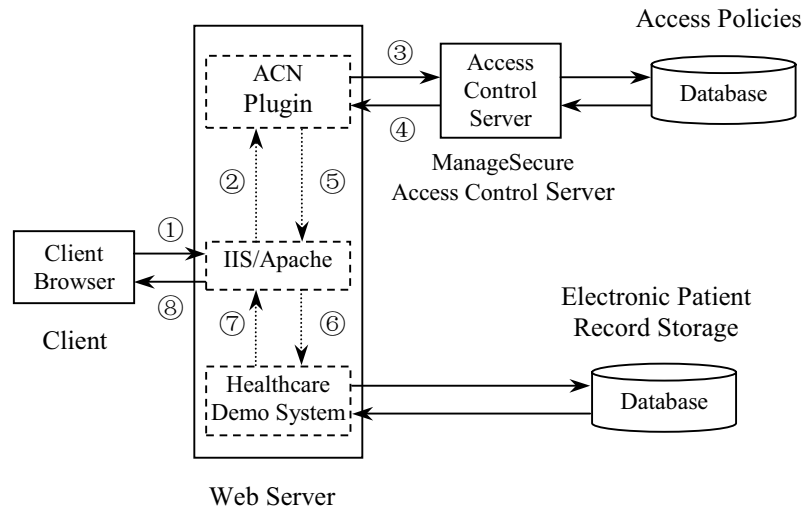


Fig. 1. An Architecture for a healthcare demo system with ManageSecure.

4.1 The Configuration of ManageSecure

In ManageSecure, there is a super Administrator (*admin*). This role is specific to ManageSecure administration, such as definition of administrative domains, domain administrators, and configuring server component properties. So we first log on Manager Console of ManageSecure as *admin*. Two domains (*Domain-M* and *Domain-H*) can be defined in the practical distributed healthcare system and their respective administrative roles can be configured as well, as shown in Figure 2.

Definition 1 Medical Practice Domain (*Domain-M*)

The Medical Practice Domain can be regarded as a virtual organization related to a real-world medical practice. It has four administrative roles that can perform specific administration functions. This allows fine-grained separation of responsibilities.

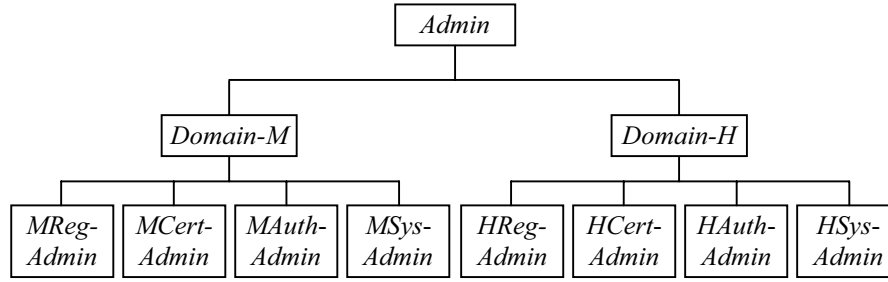


Fig. 2. Domain Structure.

In the *Medical Practice Domain*, we define four administration roles: *MCertAdmin*, *MRegAdmin*, *MAuthAdmin* and *MSysAdmin*. *MCertAdmin* is certificate authority and manages and revokes user certificates. *MRegAdmin* is web user registration authority and can add a new web user record into the system. *MAuthAdmin* is Web user authorization authority and authorizes web users to access applications. *MSysAdmin* is a system administrator and performs security administration functions that are not handled by the other roles, such as setting up server, scanning jobs.

Definition 2 *Hospital Setting Domain (Domain-H)*

The Hospital Setting Domain can be regarded as a virtual organization related to a real-world hospital setting. It also has four administrative roles that can perform specific administration functions. This allows fine-grained separation of responsibilities.

In the *Hospital Setting Domain*, we also define four administration roles: *HCertAdmin*, *HRegAdmin*, *HAuthAdmin* and *HSysAdmin*. They have the same functions as the four administration roles in the *Medical Practice Domain*

Next, we log on the Manager Console of ManageSecure as *MAuthAdmin*. And *Web Roles*, *Web Servers* and *Web Privilege* can be configured, web users' authorization requests are approved, denied or reassigned as well, as shown in Figure 3. Here, we outline several configuration examples for the *Medical practice Domain*:

(1) *Web Servers*

Web Servers that are protected using ManageSecure Access Control Nodes can be added to the policy database. In the configure of *Web Servers*, the Domain Name of the web application server is defined in detail, e.g. *www.healthcare.com.au*.

(2) *Web Privileges*

Web Privileges are actually policy definitions, as shown in Figure 4. *Web Privileges* to access various web resources can be defined based on the following criteria:

1) Server: To designate the web server's address. Here, the web server's address is *www.healthcare.com.au*.

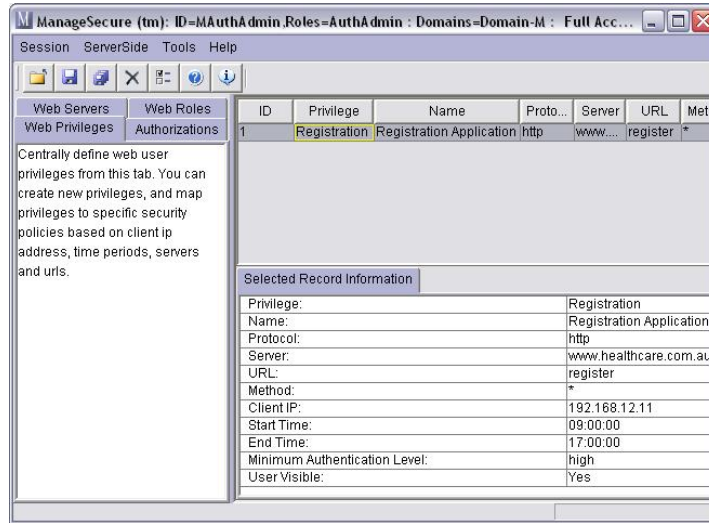


Fig. 3. Web Roles, Web Servers and Web Privilege Definition.

2) Relative URL: To designate the application's Uniform Resource Locator. It may be a web page file, or a directory.

3) Client IP Address or Sub-net: To limit access from a specific IP address, we can specify the entire IP address (e.g., 192.168.1.24). To limit access to a specific subnet, we can specify partial IP address (e.g., 192.168.*). To allow access from any IP address, we can specify this as just *.

4) Access times: By default, if we do not modify the values, then access time constraints are not enforced. However, if a start and end time is defined, e.g. between 9am and 5pm, then the resource is protected by this privilege, access to the resource is permitted only between 9am and 5pm.

5) Minimum Assurance: Minimum login assurance level required to access this resource has 6 levels (highest, higher, high, medium, low, none).

6) Visible to User: If Yes, this resource needs to be explicitly displayed to the user as part of the users welcome page (i.e., the page that is displayed after logging in successfully). Otherwise, the resource is invisible to the user.

(3) Web Roles

Web Roles are actually organizational roles, and roles are in-turn mapped to privileges. So, a user can be mapped to an organizational role. The user will receive access to operations based on the privileges mapped by all her/his role. we can create a new role, for example, receptionist, and then redefine the privileges associated with the role. In addition, a role can be mapped to more than one privilege. Similarly, we can create the other roles described in the role set R .

(4) Authorizations



Fig. 4. Web Privilege Configuration.

When users submit self-registration requests through the web browser, the user registration records are submitted to the access control database, where it is available for view by the domains registration authority (*MAuthAdmin*). Once *MAuthAdmin* approves the users, they can start using the web resources using their registered identity. In the meantime, *MAuthAdmin* can deny or reassign users' authorizations in terms of the organizational requirements as well. In addition, *MAuthAdmin* can directly authorize for a designated user.

Then, we can log on the Manager Console of ManageSecure as *MRegAdmin*, add Web user, disable and enable web user. Log on the Manager Console of ManageSecure as *MSysAdmin*, we centrally define server names and authentication types, view an enterprise web access event logs, and also generate reports from these logs, and so on.

In addition, to implement a single sign-on experience and access control using ManageSecure, we need to install another component - access control node. We first stop IIS web server on the healthcare demo system, and install an instance of the access control filter. Then IIS web server is restarted again.

4.2 The Customization of Application

After the ManageSecure configuration is successfully finished, we customize the healthcare system using ManageSecure. Here, we define 4 steps in the following:

- (1) Define users for the healthcare demo system.

On the one hand, we can log on the Manager Console of ManageSecure as *MRegAdmin*, and add users, e.g., *zgan*. On the other hand, end-users can register themselves as web users using User SelfRegistration Component of ManageSecure.

(2) In terms of the access control policies depicted in Section 3, we customize all web pages corresponding to the policies.

For example, the policy 1 in Section 3.2, receptionists have *Create, Write, Modify* access to all fields of E_1 in the medical practice, we design *Registration application* (*register.html*).

(3) Define Privilege.

according to the configuration method in Section 4.1, we define *Privilege ID* as *Registration*, *Application* as *Registration application*, *Relative URL* as */web/register*, as shown in Figure 3 and Figure 4. Accesses are only allowed from the Client IP address 192.168.12.11. Access to this resource is only permitted when the receptionist is on duty between 9am and 5pm, and he/she must use the computer whose IP address is 192.168.12.11.

(4) Authorization.

Once the user *zgan* successfully logs on *www.healthcare.com.au* from a client, he can request authorization (access to *Registration Application*) from the domains registration authority. After *MAuthAdmin* approve his request, he can gain access to the application, as shown in Figure 5. Otherwise, he can not see the application hypelink, that is, his access the *Registration Application* is denied.

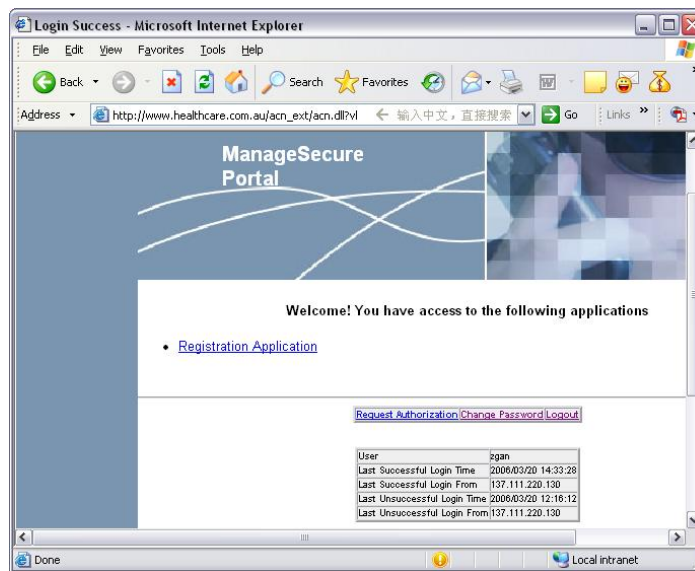


Fig. 5. Registration Application.

5 Discussion

Typical web applications use login/password based authentication and RBAC [6][7]. Relying on applications to perform authentication and authorization leads to networks with multiple disparate authentication and authorization policy enforcements based on application implementation. Also, when there are large number of applications, users have to remember and manage multiple accounts and passwords, and these applications do not provide the enterprise a central point of control over authentication policies and authorization mechanisms, for example, when an employee is terminated, her/his access should be cancelled to all applications on the Intranet. These lead to many inconvenience for them and threaten the security of these systems, for example, if a password is stolen or forgotten. A dismissed employee can still use certain applications, and so on.

However, ManageSecure has a common, consistent, standard-based security layer between the user and enterprise applications, which allows us to centrally control and monitor the access, and reduces the security burden on the end-user. A user is authenticated once to the network and can access all applications without repeated authentications.

In addition, Passwords are vulnerable to guessing, sniffing, keystroke loggin, and social engineering attacks. Hence, for some sensitive information, ManageSecure can also provide the two-factor authentication function implemented using a combination of client certificate and users passwords. The solution can greatly simplify implementation and strengthen security.

Because the security control function is separated from the development of application systems, the design of application systems has to adapt to the access control requirements and ManageSecure. Here, the number of web application pages is dependent on the access control policies. Having more access control policies means there will be more web application pages.

6 Concluding remarks

Security plays a vital role in the design and practical deployment of distributed applications, and the implementation of the security mechanism is also a time-consuming and money-consuming work. In this paper, we investigate access control requirements for the practical distributed healthcare system and describe the use of ManageSecure in securing the healthcare system. Although this work is an experimental practice, it has shown that the use of the integrated security management tool (ManageSecure) in the practice application is economical and feasible.

However, in practice, the RBAC model of ManageSecure is insufficient in itself to meet the needs of the healthcare application, in which individual patients may express additional conditions restricting access to their records. In ManageSecure, we can define the access privilege to a URL resource based on role@domain attribute so that it passes the individual user's identity and role@domain to the application at that URL, the application can perform additional fine grained authorization (in addition to the RBAC). In addition, we

will try to extend the security function and enhance its flexibility with the API provided by ManagesSecure.

Acknowledgements

The work was sponsored by China Scholarship Fund. We thank Dr. Raja Kailar and technical support engineers of Business Networks International for their valuable help in implementing the practical demonstration application.

References

1. Kailar R.: ManageSecure C An Integrated Enterprise Web Security System. <http://www.bnetal.com/raja/papers/ManageSecurePaper20050409.pdf>
2. Scott Cantor, John Kemp and Rob Philpott, et al.: Security Assertion Markup Language (SAML) v2.0. <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>
3. Economic Impact Assessment of NISTs Role-Based Access Control (RBAC) Program. <http://csrc.nist.gov/rbac/rbac-impact-summary.doc>
4. Bai Y., Varadharajan V.: A Logic for State Transformations in Authorization Policies. In: Proceedings of the 10th Computer Security Foundations Workshop. Rockpor, USA. June 10-12, 1997. IEEE Computer Society Press, 1997, 173-182
5. Sim P.: Access Control Requirements for a Medical Practice and Hospital Environment and a Secure Access Control Architecture, Masters Thesis, Macquarie University (2002)
6. Joon S. Park, Ravi Sandhu and Gail-Joon Ahn: Role-Based Access Control on the Web. ACM Transactions on Information and System Security, Vol. 4, No. 1, (2001), 37-71
7. David F. Ferraiolo, John F. Barkley, and D. Richard Kuhn: A Role-Based Access Control Model and Reference Implementation Within a Corporate Intranet. ACM Transactions on Information and System Security, Vol. 2, No. 1, (1999), 34-64