MEDINFO 2010
C. Safran et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-588-4-874

Healthcare System Evolution towards SOA: A Security Perspective

Vassiliki Koufi, Flora Malamateniou, George Vassilacopoulos, Despina Papakonstantinou

Department of Digital Systems, University of Piraeus, Piraeus, Greece

Abstract

Healthcare providers often face the challenge of integrating diverse and geographically disparate IT systems to respond to changing requirements and to exploit the capabilities of modern technologies. Hence, systems evolution, through modification and extension of the existing IT infrastructure, becomes a necessity. This paper assumes a healthcare systems evolution towards a service-oriented architecture (SOA) and places emphasis on the development of an appropriate authorization model and mechanism that ensures authorized access to integrated patient information through web service invocations.

Keywords:
Healthcare systems, SOA, Context-aware authorization.

Introduction

Healthcare delivery is undergoing radical change in an attempt to meet increasing demands in the face of rising costs. In this context, healthcare providers adopt new ways of delivering healthcare involving a major shift to information systems that enable authorized access to integrated patient information anytime, anywhere [7,9]. Thus, healthcare providers are faced with the challenge to safeguard their significant past investments in the development of large and complex information systems by modifying and extending them in response to changing requirements in addition to capitalizing on modern technologies [8]. To this end, a systems evolution process is often designed and implemented with the objective to achieve interoperability among diverse systems that may have been developed at different times and with different technologies. Such evolutionary processes are operating on legacy systems, are directed toward long-term user needs and are usually conducted in an iterative and incremental manner [7,10].

Contemporary system evolution processes are increasingly directed towards Service-Oriented Architectures (SOA) since such system architectures are thought to maximize IT support to business processes that are subject to constant change in response to a changing environment. In this respect, SOA allows users to rapidly build, reuse and reconfigure business processes as healthcare priorities, regulatory requirements or environmental conditions change [6,10]. Hence, it responds to the need of exchanging medical information between diverse systems on the web and can form an ideal architectural basis for evolving legacy healthcare systems [7,8].

In most cases, a SOA is not built from scratch but rather the functionality of legacy systems and their components are being wrapped to web service interfaces. Thus, as contemporary SOA is intrinsically reliant on web services, an evolution process towards developing a SOA often uses the transformation of legacy systems into web services as the first step. The SOA architecture assumed in this paper, as part of an evolution process, is based on an ESB/BPEL software infrastructure that enables legacy systems to be synthesized so that they serve as a unified whole. Hence, uniting legacy systems into healthcare processes involves exposing each system as a web service and, then, using BPEL to combine the web services into healthcare processes.

When the healthcare system envisaged as a result of an evolution process is SOA-based, developers are faced with the challenge not only to ensure that the evolved system supports the delivery of healthcare services within and across organizational boundaries but also to meet global security requirements that were not applicable when disparate systems were in place. Thus, developers are called upon to ensure that component sub-systems constituting a SOA can interact and exchange information subject to an appropriate level of privacy and disclosure regulations based on state of the art practices for access control [1].

This paper presents a security framework that addresses the authorization and access control issues arisen in an interoperable healthcare system that accrues from the evolution of legacy healthcare systems into a SOA-based system. The need to also evolve security has led to the development of a context-aware authorization model which is based on the role-based access control (RBAC) paradigm. This model is then used in a prototype interoperable, SOA-based healthcare system that has resulted from an evolution process to enable authorized access to integrated patient information during the execution of healthcare processes.

Motivating scenario

The basic motivation for this research stems from our involvement in a recent project concerned with defining a prototype SOA-based system architecture by evolving legacy healthcare applications. To illustrate the security aspects of the

adopted approach to systems evolution, a sample evolution project is described which is concerned with patient referrals among healthcare providers within the boundaries of a health district.

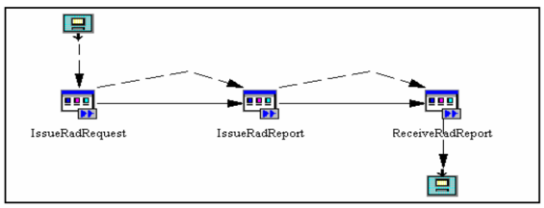

Figure 1- A high-level view of the healthcare process concerned with radiology orders.

The sample process considered here is concerned with patient referrals to a radiology department of a hospital. Suppose a healthcare delivery situation where a physician situated within a health district issues a radiological request for one of his/her patients. The request is submitted to a radiology department of a hospital which schedules the radiological procedure requested and notifies the requesting physician on the date and time scheduled. After performing the radiological procedure requested, the radiologist consults the relevant portion of the patient's record and writes a radiological report, incorporating both the radiological images and the associated text, which is sent to the requesting physician. Figure 1 shows a high-level view of the radiological process considered. This process consists of three composite tasks: *IssueRadRequest*, *IssueRadReport* and *ReceiveRadReport*.

A SOA-based evolution

The sample process described above involves two separate legacy healthcare systems which, in general, are based on different technologies:

- *Radiology Information System (RIS)*: A system that manages radiological requests and produces radiological reports.
- *Electronic Medical Record (EMR)*: A system where patient data is stored and managed.

To protect healthcare providers' investments, these legacy systems can become component applications of a loosely-coupled interoperable system (i.e. the system resulted from the evolution process). With regard to the sample healthcare process of Figure 1, all or parts of these legacy systems can be expressed as orchestrated web services (consisted of interconnected tasks) and the whole system can be a SOA implementation on an ESB/BPEL software infrastructure [4]. For example, the EMR system above may be converted into a number of web services and each web service may consist of a number of interconnected tasks. A detailed description of the system evolution approach adopted (including technical and operational requirements) is presented elsewhere.

To benefit from the advantages of SOA, the composition of two or more web services is implemented using BPEL which requires sophisticated solutions for securing web services [6]. Such an implementation may pose security problems not encountered in traditional, not SOA-based, systems. In particular, the authorization to execute a web service task built-in a SOA needs to be externalized and this authorization may be subject to certain contextual constraints that are extracted and evaluated from every request and for every application of the system [5].

An authorization perspective

Typically, role-based authorizations with regard to web service tasks and related data accesses are specified when the SOA is designed and the exact user-to-role and role-to-permission assignment relationships are decided. Context-aware authorizations are intended to provide more flexibility by taking context into account when deciding on the permission(s) that should be granted to users at run time [2,7]. Hence, these authorizations are bound to specific web service invocations and incorporate such constraints as those based on the data content, the user identity, the valid time and the location of attempted web service task accesses.

Table 1 - Extract of authorization requirements for the healthcare process of Figure 1

1	PHs may issue requests for radiological procedures on their patients. (IssueRadRequest)
2	RDs may issue radiological reports for patients on request by PHs. (IssueRadReport)
3	PHs may receive patient radiological reports issued by RDs only if requested by them. (ReceiveRadReport)

From a role-based authorization perspective, the healthcare process of Figure 1 involves two roles: the role of the physician (PH) and the role of the radiologist (RD). An extract of the authorization requirements regarding web service task execution and related data access privileges assigned to these roles is shown in Table 1. It is seen that the tasks "IssueRadRequest" and "ReceiveRadReport" can be executed by a user holding the role PH and the task "IssueRadReport" can be executed by a user holding the role RD. Also, during the execution of these tasks the users holding the roles PH and RD can have certain access rights on relevant data objects. In addition, the healthcare process of Figure 1 surfaces some additional requirements with regard web service invocations and associated task execution involved in data accesses. These requirements include the following:

- **Data content** – Some role holders should be allowed to exercise a set of permissions on web service invocations and task executions that result in accessing certain data objects only. For example, a physician is allowed to invoke the relevant web service and execute the "IssueRadRequest" task for reading patient records and issuing (write, edit and send) radiological requests only for his/her patients.

- **Permission propagation** – Some role holders should receive additional permissions on web service invocations and task executions that result in accessing certain data objects in order to effectively execute a task but these permissions should cease holding upon successful execution of the task. For example, for an effective execution of the "IssueRadReport" task with regard to a patient, in response to a request submitted by a physician, a radiologist should receive the permission to access the relevant web service and read a relevant portion of the patient's record but he/she should not be allowed to retain this permission after successful task execution. Thus, the permission for reading the patient's record, held by the patient's physician, is passed on to the radiologist who performs the radiological procedure requested for as long as is required to complete the execution of the relevant task.
- **Restricted task execution** – In certain circumstances the candidates for a web service invocation and associated task executions should be dynamically determined and be either a sub-group of the authorized users or only one, specific authorized user. For example, if the radiological procedure requested by the patient's physician is an MRI, then the radiologist who is allowed to perform it, and the one who can execute the "IssueRadReport" task, is among the radiologists holding the relevant sub-specialty. Also, the radiological report issued by the radiologist can only be read by the requesting physician who is allowed to execute the "ReceiveRadReport" task.

The authorization requirements of Table 1 suggest that permissions on web service invocations and task executions that result in certain data accesses depend on the process context. In particular, contextual information available at access time, like temporal, location or user/patient relationship or user/medical specialty relationship, can influence the authorization decision that allows a user to invoke a web service and perform a task.

Authorization model

Within a SOA resulting from an evolution of legacy systems there is a need to address authorization globally, while incorporating component system authorization policies in order to enforce the least privilege principle [2]. In turn, this requires ensuring that access permissions with regard to web services and associated tasks are only awarded dynamically, thus allowing users to assume the absolute minimum role required for web service invocations and task executions. Hence, there is a need to enforce context-aware authorization constraints regarding web service invocations and associated task executions that result in accesses to patient information.

The context

From a SOA authorization perspective, context can be defined as any information which is available at run time and is considered relevant to web service invocations and associated task executions that result in data accesses [5]. Thus, every SOA-based system may be assumed to be associated with a context which is defined as an evaluation of a set of pre-specified, domain-dependent and domain-independent context types. Domain-dependent context types are those related to the subjects and objects involved in the particular healthcare process under study as well as the relationships between subjects and objects (e.g. user and patient as well as the "proximity" relationship between them) while domain-independent context types are those related to the environment (e.g. time and location).

In a typical RBAC environment, roles are often defined as named collections of capabilities and privileges intended to perform healthcare functions [2]. In our context, roles are divided into two main classes: strong and weak. Strong roles correspond to existing organizational structures and define the division of work and the lines of authority based on job functions and seniority (e.g. "physician" and "radiologist"). On the other hand, weak roles are derived from strong roles and are subject to domain-dependent contextual constraints (e.g. "attending physician" and "attending radiologist"). For example, a radiologist can only assume the weak role "attending radiologist" when a physician has issued a request for performing a radiological procedure on one of his/her patients and the particular radiologist is assigned or chooses to respond to the request.

To alleviate users from the burden to change roles at run time, as required by access needs and the current work context, and to reduce the administrative overhead imposed, a rule-based approach should be adopted that enables automatic role changes (e.g. from strong to weak and vice versa) on the occurrences of specific events which are termed "role change events". These events occur when a web service invocation is initiated (initiation events) and result in granting the appropriate weak role to the current user; they terminate when the web service invocation is terminated (termination events) and result in revoking the current user's weak role. To achieve role changes, event occurrences automatically trigger the processing of relevant rules defined in the event-condition-action (ECA) format [3]. Hence, role change rules can be described as follows:

Definition (Role change). A role change is a 4-tuple (u, r_i, r_j, e_k) stating that a user u holding the role r_i receives the role r_j on the occurrence of the event e_k.

Authorization rules

To reduce the complexity of the authorization model, only positive authorizations are considered, so no explicit authorization conflicts occur. The dynamic authorization model proposed here extends the classic role-based access control (RBAC) for SOA-based authorization, while retaining its advantages, in that it addresses the issue of dynamically changing user roles at run time based on event occurrences. Hence, web service invocation and task execution authorization rules can be described as follows:

Definition (web service invocation). A role-based authorization for web service invocation is a 4-tuple (r, "invoke", WS,

$\{p_k\}$) stating that a user holding the role r is allowed to invoke web service WS subject to contextual constraints $\{p_k\}$.

Definition (web service task execution). Given an authorization for invoking web service WS by a user holding the role r, a role-based authorization for web service task execution is a 5-tuple (r, "execute", WS, T, $\{p_k\}$) stating that a user holding the role r is allowed to execute task T of web service WS subject to contextual constraints $\{p_k\}$.

Such authorization rules contain both domain-dependent and domain-independent contextual constraints. For example, on an attempt to invoke a web service, the contextual constraints are evaluated and applied in order to restrain accordingly the permissible actions of a user.

SOA authorization system

In a SOA environment, a typical authorization architecture involves a subject (e.g. a user) which wants to access an object (e.g. a service). The authorization architecture takes care of requesting authorization decisions and enforces them. To this end, it has to intercept user requests asking whether the user is authorized based on an evaluation of the applicable policies and building the authorization decision (deny/permit) upon that.

A SOA-based system accrued from rendering legacy systems interoperable should not violate any security enforced in these systems but it can itself enforce additional security. Hence, an implementation of a centralized authorization architecture can be made with due regard to the security policies that have been incorporated into the legacy systems.

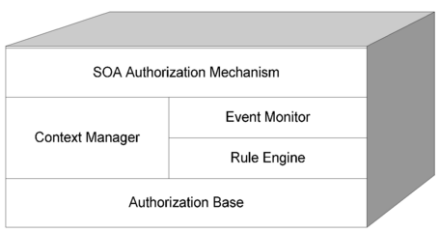

Figure 2- Authorization system architecture

Based on the authorization model defined above, an experimental SOA authorization system has been designed and implemented. This system enforces role-based authorizations that determine who (in terms of role) can invoke which web services, and can execute which web service tasks, and under what conditions. As shown in Figure 2, the architecture of the prototype authorization system consists of the following main components: the authorization base, the event monitor, the rule engine, the context manager and the authorization mechanism. The authorization base records all the elements of the model. The event monitor manages generated events and keeps track of the events that have already occurred in a healthcare process. The rule engine ensures that the rules set about invoking web services and executing web service tasks are enforced. The context manager collects the context information used in making access control decisions. The SOA authorization mechanism maintains user information and intercepts user requests (web service invocation or task execution) asking whether the user is authorized based on an evaluation of the applicable policies and building the authorization decision (deny/permit) upon that subject to contextual constraints.

A. Authorization base

The authorization base records users, roles (strong and weak), web services and permissions as well as user-to-role and role-to-permission assignments. Role-based authorizations are stored in the form of authorization rules with regard to web service invocations and web service task executions, most of which have been converted from legacy system authorizations to retain alignment between old and new system authorizations. Hence, source system security is conveyed. In addition, the authorization base contains role change event occurrences as well as the contextual constraints defined.

B. Event monitor

The event monitor records event occurrences into the authorization base and passes the event occurrence data to the rule engine that triggers the appropriate rule (e.g. for weak role granting/revocation). The event monitor is informed on event occurrences when a user attempts to initiate or terminate a web service invocation.

C. Rule engine

The rule engine stores weak role granting and revocation rules in ECA format, determines the appropriate rule when information on the occurrence of an event is received from the event monitor and triggers the weak role granting (revocation) action when the condition specified in the rule is satisfied. Weak role granting (revocation) amounts to recording (deleting) an entry in the user-to-weak role relationship table of the authorization base. Thus, the rule engine maintains the authorization base information up-to-date regarding dynamic granting/revocation of weak roles to strong role holders.

D. Context manager

The context manager, on occurrence of an event, determines the current work context and communicates the collected information to the rule engine which uses it when triggering the relevant ECA rule. The context manager is realized by a number of context agents which use middleware context collection services to monitor context and interact with the rule engine.

E. Authorization mechanism

The authorization mechanism accesses the permission relationships of the authorization base to enforce access control on users holding a role at the time of an attempted web service invocation or task execution. Thus, on an attempted web service invocation or task execution the authorization mechanism mediates between role holders and web services or tasks to determine whether the requested action should be permitted or denied.

Implementation issues

To illustrate the functionality of the proposed security model, a prototype SOA-based system was implemented that draws on the business rules and authorization requirements of the healthcare process depicted in Figure 1 and on the two legacy systems mentioned above.

Referring to the healthcare process of Figure 1, two web services have been developed: The one is a version of RIS concerned with radiological requests ("RIS_RadRequest" web service) and the other is a version of EMR concerned with medical record activities performed by either the physician or the radiologist ("EMR_RadPortion" web service). Thus, basically, the tasks depicted in Figure 1 belong to the "RIS_RadRequest" web service although these tasks include sub-tasks that are concerned with invoking the "EMR_RadPortion" web service to access medical record data.

Let a physician wishing to order a radiological procedure for one of his/her patients. Then, the following actions are performed with regard to security: On attempting to invoke the RIS_RadRequest web service, a web service invocation event occurs which is identified by the event monitor and is recorded into the authorization base while event occurrence data is passed to the context manager and to the rule engine. Then, the context manager evaluates the contextual parameters specified to determine the current context that constraints authorization rights (e.g. the patients that are currently attended by the physician) and passes this information to the rule engine. The rule engine identifies and triggers the relevant ECA rule to grant the weak role "attending physician" to the user, by inserting an appropriate entry into the authorization base, and passes the weak role to the authorization mechanism. The authorization mechanism consults the authorization base and grants to the user the restrained permissions for invoking the web service, performing the relevant tasks ("IssueRadRequest" and "ReceiveRadReport") and accessing the relevant patient's data only in order to issue a radiological request for one of his/her patients. Due to lack of space, a detailed description of the SOA authorization system implementation will be presented elsewhere.

Conclusion

Healthcare organizations are faced with the challenge to improve healthcare quality and reducing healthcare costs. To these ends, healthcare organizations often attempt to evolve their legacy systems by using innovative information technologies. However, for such a system to reach its full potential in supporting healthcare activities, authorization mechanisms must be in place that can conveniently and cost effectively regulate user access to information while providing confidence that security policies are faithfully and consistently enforced.

This paper presents an authorization model and system for enforcing authorization when migrating existing systems into a SOA. Based on the well known RBAC paradigm, a context-aware authorization model which is focused to the upper layers of SOA (i.e. the integration layer and the process layer) has been introduced and its practicability has been demonstrated using a prototype system based on a sample healthcare process.

References

[1] Bhatti R, Samuel A, Eltabakh MY, Amjad H and Ghafoor A. Engineering a Policy-Based System for Federated Healthcare Databases. IEEE T Knowl Data En 2007: 19 (9): 1288-1304.

[2] Casati F, Castano S and Fugini M. Managing Workflow Authorization Constraints through Active Database Technology. Information Systems Frontiers, 2001: 3 (3): 319-338.

[3] Casati F and Sham MC. Event-Based Interaction Management for Composite E-Services in eFlow. Inform Syst Front 2002: 4(1): 19-31.

[4] Davidsen L. Building an ESB without limits. IBM Software Workgroup, 2007.

[5] Emig C, Schandua H and Abeck S. SOA-aware Authorization Control. In: International Conference on Software Engineering Advances, 2006.

[6] Erl T. Service-Oriented Architecture (SOA): Concepts, Technology, and Design. The Prentice Hall Service-Oriented Computing Series from Thomas Erl., 2005.

[7] Ferrara FM. The CEN healthcare information systems architecture standard and the DHE middleware – A practical support to the integration and evolution of healthcare systems. Med Inform, 1998: 48: 173-182.

[8] Kart F, Moser L, and Melliar-Smith M. (2008). Building a Distributed E-Healthcare System Using SOA. IT Professional, 2008: 10 (2): 24-30.

[9] Lenz R and Kuhn K.A. Towards a continuous evolution and adaptation of information systems in healthcare. Med Inform, 2004: 73: 75-89.

[10] Pasley J. How BPEL and SOA are changing web services development. IEEE Internet Computing, 2005: 9 (3): 60-67.

Address for correspondence

Ms Vassiliki Koufi, Department of Digital Systems, University of Piraeus, 80 Karaoli & Dimitriou str. 18534 Piraeus, Greece. Email: vassok@unipi.gr