An Approach in Designing an Authorization Layer for Web Services using an Inference Engine

Weider D. Yu **Archana Mansukhani**

San Jose State University, San Jose CA 95192-0180

Abstract

Web services are a new way of thinking in distributed computing. They are an important step towards service-oriented architecture (SOA). Web services are used to obtain service in an open, platform independent way. Recent focus on web services has been in the area of security, which is an ongoing concern in many areas and is very pertinent to web services technology.

This paper describes the design of a reusable authorization layer for web services software. This layer resides separate from the web services themselves and uses a rule based inference engine for determining authorization and access rights. It also uses different types of access control to formulate feature-rich rules.

Keywords: Web Services, authorization layer, security, inference engine, reusability, Service Oriented Architecture (SOA), expert system rules.

1. Introduction

The industry has seen a rapid growth in web services security since the beginning of 2004. Many new standards have been established to deal with various aspects of security. OASIS (Organization for the Advancement of Structured Information Standards) has defined several standards for web services security. Two of the standards are relevant to authorization: XACML (XML Access Control Markup Language) [9] which is a language for specifying access control rules and policies, and SAML (Security Assertions Markup Language) [10], which is a protocol for specifying authentication and authorization. However, there are no standards exist, for defining an authorization framework for web services. The expected WS-Authorization standard is still under observation.

This paper proposes a reusable authorization layer, which is separate from the web services themselves. All web services requests pass through the authorization layer. An authorization decision is made at this layer and passed onto the web services [1], [2].

This paper proposes to use logic rules in determining access rights. A logic based inference engine is used in determining the outcome of each authorization decision. Rules are fired depending on the facts and conditions for an authorization request. Using this approach effectively establishes a separate authorization layer at which authorization decisions are made. Furthermore, the framework of the authorization layer can be reused in multiple web services in order to service various authorization needs.

2. Authorization Layer

The purpose of this research work is to design a new architectural framework to provide authorization functionality. Although authentication is an important step towards authorization, strict authentication is assumed to be available and it is outside of the scope of this paper.

Figure 1 shows an overview of the architectural framework of the system. It is assumed that web services requests are authenticated before being processed by the authorization layer. The main goal of this design is to separate all authorization related functionalities so that they can be designed and implemented independently from the rest of the web services implementation. This enables the reusability of authorization functionality.

The authorization layer parses the incoming web service request and produces various parameter data required for authorization. It then formats the input data into the appropriate type required by the inference engine. The inference engine uses the input and its authorization rules to determine the appropriate access rights for the request. A set of authorization rules is used to determine which user or service can access what component or service [3], [5].

2.1. Language syntax for specifying Authorization rules

The authorization rules in the approach are specified using a special functional language of predicate calculus type. Some common notations are used. ∀ is the universal qualifier and indicates *for all* or *for every* instance. ∃ is the existential qualifier and indicates that there exists *at least one* instance. The symbol ⇒ indicates that there is an *if-then* relationship.

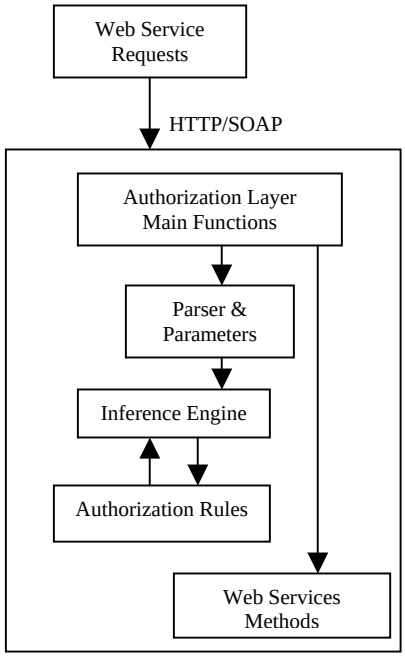

Fig. 1: System Architecture

If a company employee as a client user wants to connect to his/her company computer from outside the company environment, then the client's role is changed from *on_site employee* to *off_site employee*, and the client's access rights to certain resources are modified based on the policy of the company's off-site employee access rights. These policies are company dependent. The rules for specifying such access restriction are expressed as:

$\forall x$ [on_site_emp(x) $\Rightarrow$ *access*(E)]

$\forall x$ [off_site_emp(x) $\Rightarrow$ *access*(E')]

where on_site_emp(x) is a predicate for *x is an on_site employee*. *access*(E) is a relationship between the employee x and the component E that implies that the employee x is authorized to access the resource component E (and the resource component E' is a subset of E). In this case, the resources in the company are represented by components.

The factual data required to support the predicate must be analyzed and the employee status is determined during the authentication stage. This information is passed on to the authorization layer or it must be included in the SOAP (Simple Object Access Protocol) request received at the authorization layer.

In another example, the rule below is for some off_site employees who has unrestricted access rights to the resource component E:

$\exists y$ [off_site_emp(y) $\Rightarrow$ *access*(E)]

2.2. Data for Authorization

Certain client or user related data are required from a web service request to make authorization decisions. Such data may be user or service id, role used for accessing the service, team to which the user belongs, task being worked on by the user, and certain other data related to the particular service.

The data required depends on the authorization types supported. This kind of data may be either directly obtained from the requesting client or obtained via an XML schema for the client's web service SOAP request. The security of the data must be enforced against malicious intruders.

3. Major Development Benefits

Several major benefits can be achieved with the approach:

- Using an expert systems shell allows greater flexibility in defining rules. Authorization designers can use rule-oriented facilities provided by the shell to define and manage authorization rules. Without this, it is a cumbersome task to manage rules using a database and no tools.
- Access control using rules fits naturally into the conceptual framework of the expert system shells whereby rules are "fired" depending on existing facts. These facts can relate to roles, location, user, requesting service, task, context, team, time and so on [6], [7], [8].
- Authorization rules become portable. They can easily be ported and used in other web services for similar types of access. Separating the rule definitions as well as execution from the underlying services effectively separates the authorization layer thus rendering it reusable and extendable.

4. Architectural Layer Design and Its Implementation

The web services example described in section 2.1 was implemented using the proposed architectural design. A short description of the web services example is given below.

4.1. Resource Components Used in a Case Study

A company, AMS Inc., wants to use the Service Oriented Architecture (SOA) concept to deploy certain internal intra-company services as well as its external commercial services via web services. The company also wants to make part of its ERP (Enterprise Resource Planning) system available in the web services.

Figures 2 and 3 show detailed resource components for some subsystems of the ERP system that must be exposed as web services. The resource components are typically structured to form various resource component hierarchies. As the scope of an ERP system is large, only the following subsystems are shown in the paper: Project Management (PM) and Human Resource Management (HRM).

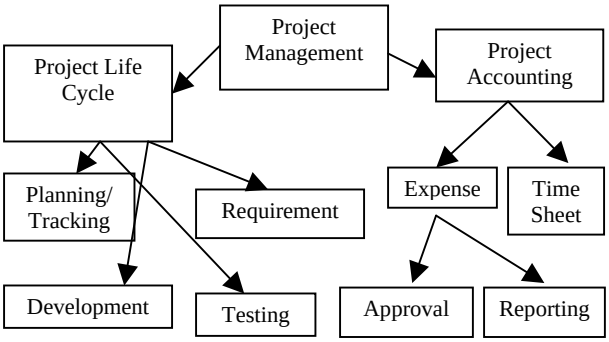

Fig. 2: Components of Project Management

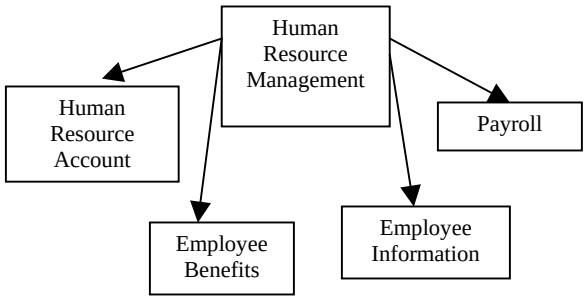

Fig. 3: Components of Human Resource Management

4.2. Resource Component Access Requirements Analysis

A special requirement analysis is conducted to provide a detailed requirement specification document regarding various authorized access rights to the resource components. Some of the requirements defined for the resource components include:

- A product engineer in a team is allowed to access resource components managed by that product team only.
- A quality assurance engineer has access rights to all the resource components for which he/she is responsible, which may span multiple teams.
- All employees can access the resource components, which are directly related to themselves, such as company policy and employee benefits.
- A manager can access any resource components for all the teams managed by him/her.

4.3. Inference Engine Used

CLIPS, [4], a well-known rule based expert system tool, is used as the inference engine in the implementation of the authorization layer. Rule-based knowledge representation scheme is used to represent the specific knowledge of resource access rights. The system performs reasoning capabilities based on pattern matching.

The inference engine and the rules constructed by using CLIPS and other program components for the authorization layer can be easily integrated in C/C++ programming environment. This is a major advantage achieved by using CLIPS as the inference engine tool.

4.4. Authorization Rules Specifications Using ARS

Authorization Rule Specifier (ARS) is a new special purpose language, designed to be used for specifying authorization rules in the system.

It is required during the requirement analysis stage to define a set of meaningful shorthand notations for expressing the resource components to be used. Some of the examples are shown:

- PM stands for Project Management. And PM-PA-TS means that Time Sheet (TS), which is a part of the Project Accounting (PA), which is in turn a part of Project Management (PM).
- PM-PLC-[DEV | PT | RM | T] means that [Development | Planning & Tracking | Requirement | Testing] component, which is a part of Project Life Cycle (PLC), which is a part of Project Management (PM).
- HRM stands for Human Resource Management, and HRM-[EI | EB] means that [Employee Information | Employee Benefits] is a part of Human Resource Management (HRM).

The authorized access right granted at a certain level of an organizational resource component hierarchy does not imply the access rights to the resource components at its lower level. It simply means that the user can be considered for giving access rights to the components at the lower level.

All the required authorization rules are specified using the language ARS. Other than the quantifiers, logical connectives are also available in ARS, such as $\wedge$ (AND), $\vee$ (OR), and $\neg$ (NOT), which can be used to specify logical expressions in authorization rules.

Some authorization rules for the example are shown below:

1. $(\forall x)$[loc-secure(x) $\wedge$ context-teamtask(x) $\Rightarrow$ *access*(PM)]
2. $(\exists x)$[access-PM(x) $\wedge \neg$ (role-user(x) $\vee$ role-customer(x)) $\Rightarrow$ *access*(PM-PA-TS)]
3. $(\exists x)$[access-PM(x) $\wedge$ team-A(x) $\wedge$ role-engineer(x) $\wedge$ task-dev(x) $\Rightarrow$ *access*(PM-PLC-DEV)]
4. $(\forall x)$[access-HRM(x) $\wedge$ role-employee(x) $\wedge$ time-office_hours(x) $\wedge \neg$ (role-user(x) $\vee$ role-customer(x)) $\Rightarrow$ *access*(HRM-EI) $\wedge$ *access*(HRM-EB)]
5. $(\forall x)$ $(\exists y)$[access-PM(x) $\wedge$ role-manager(y) $\wedge$ task-dev(x) $\wedge$ role-direct_support(y, x) $\Rightarrow$ *access*(PM-PLC-PT) $\wedge$ *access*(PM-PLC-RM) $\wedge$ *access*(PM-PLC-DEV)]

4.5. Semantic Interpretations of the Rules

The semantic interpretations of the above rules are stated below

1. Any client user (x) who would like to be granted with access rights to resource components from a secure location, and the client user (x) is in a context environment involving team tasks, then the client user (x) can access those resource components of Project Management (PM).
2. There exists someone (x) who is specially authorized to access the resource components of Project Management (for x's team) and the person's (x's) role is neither a customer nor a user, then the person (x) can access the Time Sheets (TS) component which is a part of Project Accounting (PA) of PM.
3. There exists someone (x) who is in a team A and has access to resource components of Project Management for the team. If the person (x) is an engineer and involved in a development task, then the person (x) can access the development component portion in Project Management-Project Life Cycle (PM-PLC) for the team A.
4. Any employee (x) (not an user or a customer) who has access rights for the resource components of Human Resource Management (HRM) can access their information and benefits during office hours by accessing Employee Benefits (EB) and Employee Information (EI) components in HRM. A person who is not an employee cannot access this information.
5. For any person (x) who has access right for the resource components of Project Management and whose task involves development, if there exists a person (y) who is a manager and the person (x) is directly supported by the person (y), then the person (x) can access the resource components: planning and tracking (PM-PLC-PT), requirement (PM-PLC-RM) and development (PM-PLC-DEV) for the team's project.

4.6. Implementation Architecture

The authorization layer of the web service was implemented in .NET using managed C++ extensions. A CLIPS wrapper was compiled as a separate module and linked with the web service. This wrapper contains the authorization layer, as shown in Figure 1.

The data required for authorization may be obtained in various ways. For example, location of a user may be obtained by modifying the SOAP header to retrieve the IP address of the user, which is used for performing location based access control. The authentication module, based on the user account, may be used to specify roles.

Once information about the user such as role, location, context, task, team etc. is obtained, the authorization layer uses that information and contacts the inference engine. Depending on the facts asserted about the user, certain rules will fire. As each rule fires, it leaves a fact in the working memory of the inference engine. After all possible rules have fired; the facts remaining in the working memory determine what access rights the user should be granted.

5. Comparison with Related Work

There is no standard that exists for specifying how to implement authorization for a web service. However OASIS has defined two standards related to authorization and access control. XACML [9] is a language for specifying access control policies and

rules. SAML [10] is a framework for communicating user authentication and authorization information in the form of assertions. The .NET framework also provides a way of authorizing web services requests. A comparison with these three authorization approaches is presented in this section.

Flexibility: In the inference engine based authorization approach, different types of authorization may be used depending on the application. The case used has demonstrated the use of role based, team based, task based, user based, time based, and context based among others. Other forms of access control, not covered here, may be added.

.NET based authorization approach only allows user or role based access control. This limits the application flexibility and requires defining everything in terms of roles and users. In addition, .NET based authorization does not provide fine-grained control. The XACML based approach does not prescribe any particular type and is as flexible as the inference engine based authorization. SAML supports attribute based authorization, and is typically used to support authentication in a limited way.

Extensibility: The inference engine based authorization can be extended with ease. It is not limited to role based or location based access control only. As new access control models are introduced in the future, the new rules provided can be integrated into the system without requiring too much effort.

In the .NET approach, this advantage is harder to achieve. XACML and SAML are both extensible as a new XML schema may be easily defined to support a new authorization type.

Maintainability: In the inference engine based authorization approach, there is only one authorization layer at the organization level and the access control rules are stored at one location, thus the maintenance tasks become easier and simpler.

In the other approaches, authorization must be specified for each web service and implemented along with the web services.

Testability: The inference engine based authorization approach provides an interactive specification and testing environment to test the access control rules before their deployment. CLIPS provides a command line tool and the web services administrator can use the tool to enter rules, assert facts, define functions, define global variables and thus test the rules.

In .NET SAML and XACML based authorizations, there is no such capability.

Reusability: The access control rules used in an inference engine based authorization can be easily reused for other web services using a similar access control policy. The rules can be either ported to or shared with other web services.

In all the other approaches, the access control mechanism is embedded in each web service individually.

6. Conclusions

This paper presents an approach to designing a reusable authorization layer that may be used for web services software authorization. Access rights may easily be defined by a set of rules. Authorization may be granted or denied depending on these rules defined for the application. Rules can be specified in a language of predicate calculus type.

The design is based on using multiple authorization methods to construct rules. Rules may also be constructed hierarchically. The access control of authorization types can be role-based, location-based, context-based, task-based, or team-based.

The design meets all the goals originally envisioned. Separating the authorization into a layer that resides above the normal web services makes the system reusable. It can be easily plugged into another system.

Adding an expert system adds several advantages; rules become easily maintainable and reusable. Flexible and rich rules may now be composed and the result is obtained by executing the authorization rules in the inference engine. The abilities of composing and testing rules are available before applying them in the system.

7. References

[1] F.T. Alotaiby, J.X. Chen, "A Model for Team-based Access Control (TMAC2004)," *Proc. Of IEEE Information Technology: Coding and Computing*, pp. 450-454, 2004.

[2] S. Chaari, et al., "An authorization and access control model for workflow," *Proc.Of IEEE Control, Comm. and Signal Processing*, pp. 141-148, Mar. 2004.

[3] R. Chandramouli, "A framework for multiple authorization types in a healthcare Application system," *Proc. Of Computer Security Applications Conference*, pp. 137-148, Dec. 2001.

[4] http://www.ghg.net/clips/CLIPS.html.

[5] S. Indrakanti, et al., "Authorization service for web services and its Implementation," *Proc. Of*

IEEE International Web Services Conference, 2004. pp. 774 – 777, Jul. 2004.

[6] A.K. Mattas, et al., "Towards dynamically administered role-based access control," *Proc. Of 14th International Workshop in Database and Expert Sys. Applications*, pp. 494-498, Sept. 2003.

[7] G.H. Motta, S.S. Furuie, "A contextual role-based control authorization model for electronic patient record," *IEEE Trans. on Inform. Technology in Biomedicine*, vol. 7, no. 3, 202 – 207, 2003.

[8] M.J. Moyer, M. Abamad, M., "Generalized role-based access control," *Proc. Of 21st International Conference on Distributed Computing Systems*, pp. 391-398, Apr. 2001.

[9] http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf.

[10] http://www.oasis- open.org/committees/download.php/11902/saml-2.0-os.zip