Web Service Authorization Framework

Thomas Ziebermayr, Stefan Probst
Software Competence Center Hagenberg, Hauptstrasse 99, 4232 Hagenberg, Austria
thomas.ziebermayr@scch.at, stefan.probst@scch.at

Abstract

Web Services represent an important technology for distributed applications and will replace various other technologies for distributed application development soon. A lot of problems of the early days of Web Services are solved now. However, for authorization no appropriate solution is available and ready to use. We define requirements for authorization of Web Services and investigate existing authorization solutions concerning these requirements. Based on existing authorization solutions and the defined requirements, a Web Service Authorization framework is developed. We describe concepts and the design of the proposed framework and give an overview of selected implementation aspects (e.g. authorization data access, descriptive deployment). The framework emphasizes easy deployment of Web Service authorization and is ready to use. Practical experience using the framework concludes the paper.

Keywords: *Web Service, Authorization, Security, Service, Framework*

1 Introduction

Security is an absolute need in today's software applications. Since the trend is to use web-based software, new security issues arise. Software does not run anymore in a small and manageable environment but rather in an environment with many uncertainties concerning the users, the participating parties and systems. Thus, the application itself must address security and provide adequate mechanisms.

As illustrated in Figure 1, security consists of several layers. Low-level security addresses the secure transmission of data via an untrusted net, using cryptography and communication security mechanisms. High-level security aims to protect the application itself by defining a security model. A security model comprises several mechanisms to enforce the desired security policy, which are in particular [11]:

- Authentication: establishes the identity of one party to another. Thus, authentication needs to prove the identity of a certain user to the system.
- Access Control: determines whether a user (subject) is allowed to access an object or not. This decision is based on the authorization of the system wide security policy.
- Auditing: gathers data about activity in the system and analyzes it to discover security violations and diagnose their cause.

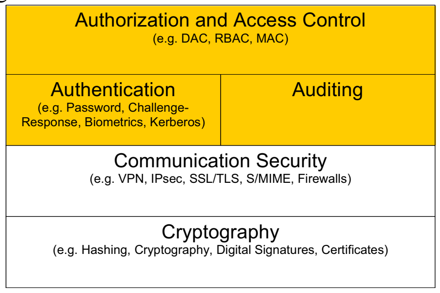
Authorization and Access Control
(e.g. DAC, RBAC, MAC)
Authentication
(e.g. Password, Challenge-Response, Biometrics, Kerberos)
Auditing
Communication Security
(e.g. VPN, IPsec, SSL/TLS, S/MIME, Firewalls)
Cryptography
(e.g. Hashing, Cryptography, Digital Signatures, Certificates)

Figure 1: Levels of security mechanisms

Reusable components for the development of security-aware applications are available especially for the lower levels of IT security, namely, cryptography and communication security. At higher levels, i.e. authorization, access control, authentication, and auditing, adequate components that can be easily reused and integrated into software applications are missing. This is especially true when looking for solutions addressing the Web Service platform. Existing development environments (e.g. Microsoft .NET, J2EE driven by Sun) already offer such security models and mechanisms. These mechanisms are mostly not expressive enough, respectively not sufficiently adaptable to complex application requirements [18]. Consequently, the enforcement of high-level security mechanisms at the application- and business-logic layers results in a practice of permanently re-inventing the wheel. This motivates to have closer look on Web Service authorization and its requirements.

The remainder of this paper is structured as follows: section 2 defines requirements for Web Service authorization. Section 3 presents a solution for Web Service authorization based on the results of previous sections. Section 4 introduces the implementation of the Web Service authorization framework and describes experiences with the implementation. Section 5 discusses open issues and future work concerning the Web Service authorization framework. Section 6 investigates related work concerning authorization and studies them compared to the defined requirements. Finally section 7 concludes the paper and provides remarks.

2 Web Service Authorization Requirements

Web Services [20] provide mechanisms to connect distributed components. They can be used in various application scenarios and only limit the application by the current absence of environmental services like security and transactions. This paper focuses on one security aspect, on Web Service authorization. In the following we list the common requirements of Web Service authorization:

- Independent from Web Service carrier protocol
- Conform to Web Service protocols
- Service, not object authorization
- Abstract solution deployable to various Web Service scenarios

These requirements underlines the necessity for platform independent authorization solutions, but they only address the common needs of Web Service authorization, they do not describe the authorization needs of “real world” Web Services. Therefore we describe a “real world” Web Service scenario and derive additional requirements from this scenario.

In this scenario Web Services are used for platform independent communication between the business logic at the server and the presentation tier. These Web Services provides access to sensitive data and are located on one server and are not distributed over the Web. The client is used by many users that lead to a large amount of requests to the Web Services. The presentation layer is responsible for authentication of the users but authorization is moved to the Web Service layer. Sensitive data delivered by the Web Services is queried based on service parameters. These parameters need to be included into authorization as they define the data set delivered to the user. The Web Service implementation is based on existing frameworks. Administration of authorization data needs to be done in parallel to the use of Web Services and the authorization used for these Web Services should be applicable to other similar Web Services as well. Based on this scenario we identified the following additional requirements:

- Fine grained authorization at parameter level
- Efficient access to authorization data
- Descriptive tailoring of authorization
- Easy usage and deployment
- Applicable to existing Web Service frameworks
- Multi user access to authorization data

In our opinion most of the currently implemented Web Service scenarios are similar to the previously described scenario, but no appropriate authorization framework exists. Especially the fine-grained level is missing. In the following section thus we provide a solution that meets these requirements.

3 Web Service Authorization Framework

Within this section the Web Service Authorization Framework (WSAF) is introduced. Previously it is necessary to investigate the basics of authorization as basis for the design of the framework.

Authorization is described by giving a subject the right to access an object (to authorize the subject). This decision is based on predefined rules that describes who is allowed to access an object. Authorization rules describe relations between subjects and objects that represent allowed access paths. So the basis for the rule description is the unique description of subject and object. This paper does not focus on authentication; therefore the definition of a subject is used as it is defined for authentication [11]. But it is necessary to introduce a notion for describing an object tailored to the intended application area: to services.

Literature concerning service authorization provides various proposals for the description of services. One that nearly fits our need is provided by [12]. This approach provides an abstract definition of Web Services. [12] defines a hierarchical structure, the Web Service object and its Web Service methods (see Figure 2).

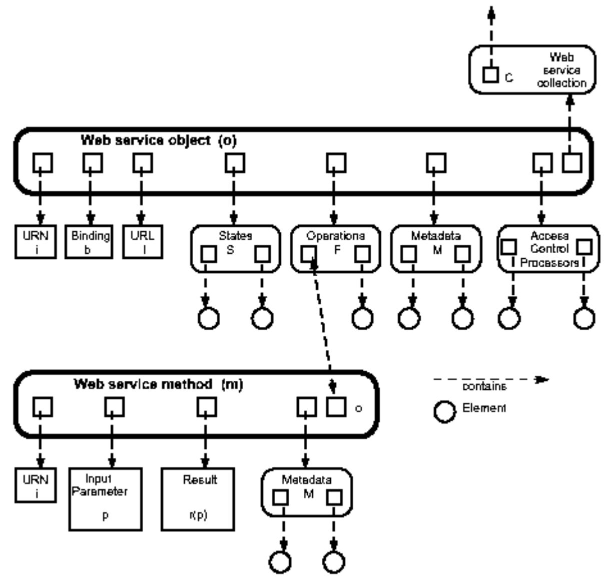

Figure 2: Web Service method [12]

This definition is rather complex and contains information that is not necessary for authorization of simple services. WSAF tailors this definition to its needs and only uses necessary attributes. It is neither necessary to consider the result of the Web Service method for authorization nor to consider the URN of a Web Service as WSAF does not consider distributed authorization. So minimizing the definition to the necessary data leads to the following definition of a service in the authorization context: A service description consists of a component name that offers services, a service name and the names of the parameters of the service (see Figure 3). This definition is simple but sufficient for the definition of the service and match an actual service call to the description in the rule base.

For authorization it is necessary to define connections between object and subject. These connections define who is allowed to use which service. These authorization rules are the basis of the authorization framework. Most of the current solutions allow defining rules on method level. But often this is not enough e.g. given a method to retrieve the details of an account `getAccountDetails(accountID)`. All people with an account are allowed to call this method, thus this restriction is not enough. It is necessary to involve the parameter value into the authorization rules.

WSAF allows to define who is allowed to call which service with which parameters. In case of the previous service the `accountID` for the caller is defined, preventing unallowed access to account data. Figure 3 shows how the authorization rule connects subject and object.

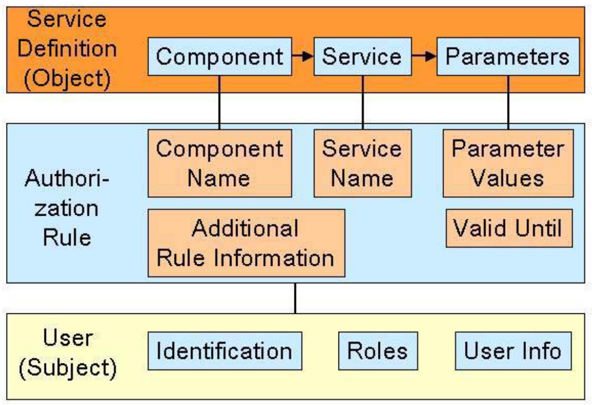

Figure 3: User service relation

An access control rule consists of a reference to a service definition, a reference to a user and additional rule information. Additionally for all parameters that should be considered in access control a value or value range is defined.

Applied to the previous example this means that the own `accountID` is defined as value for the parameter named `accountID` for the service `getAccountDetails`. All other values are not allowed, thus a service request containing another ID is rejected by the access control framework.

The ID of the subject defines the connection from the subject to the rule. Additionally the rule contains metadata for access control like a “valid until” date. This simple rule definition is enough to protect the service from unauthorized access.

To understand how WSAF works we will describe the function by an example. Given a subject (called user) who uses a service to retrieve data about the actual weather. This user is already registered at the service provider for using this service. The user does not pay for the worldwide weather service but for the European one. The access control therefore has to check whether the user is allowed to access the service and the region. Next the activities during access control are described following this example (refer to the illustration in Figure 4).

When the user sends a service request, the access control searches for the user and his rights before the request is forwarded to the service implementation. First the user must exist in the rule base; otherwise a system error occurred. This assumption is allowed, because during authentication the data entered by the user will be compared to stored data about the user, if the user does not exist, authentication does not succeed. The authorization service first searches for an exact match of the service request to the service rights of the user (component, service). When the user is allowed to use the service, the access control checks whether the service is protected by parameter restrictions. When the user has the rights to use the service without restrictions, the access control task is finished and access is allowed. If the user has parameter restrictions, the access control compares the parameter of the service request with the defined restrictions. The parameter restrictions can be defined by a single value but also by a range of values. If the parameter check succeeds, the service call is allowed.

If no rule can be found for the actual user that matches the service request at service level, the access control logic searches the rule base for matches at the component level. Rules at this level describe unrestricted access to all services of a component. It is not possible to define parameter restrictions within such access control rules. If no rule can be found for the actual user that matches the requested service at component level, access is not allowed. Without definition of rules for a user, a user is not allowed to access any service. When rules are defined, the granularity of the rule defines which services are allowed to access and if it is restricted by parameter definitions. The evaluation order of the rules follows the performance requirements of the secured application (see Figure 4).

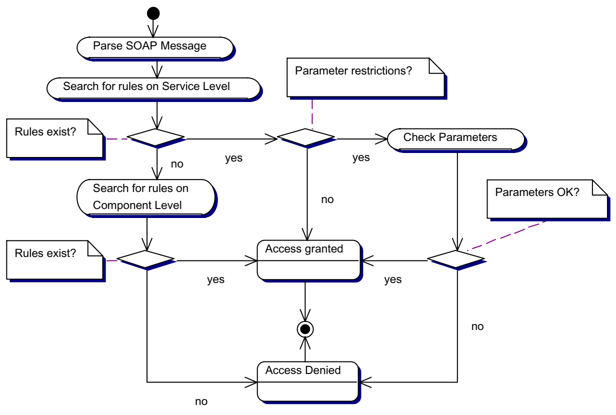

Figure 4: Authorization activities

In the most cases rules will be defined at service level, therefore it is faster to start searching for rule definitions at this level. Authorization rules at component level are high-level rules that deliver a lot of rights to the user. Therefore such rules should be used carefully. Rules at component level can be avoided when defining all rights at service level. If a component consists of four services, it will be necessary to define four service rights as equivalent to one component rule.

4 Implementation and Experiences

The implementation of the framework is tailored to the use for Web Services. Therefore Web Service enabling technologies are used. The framework follows the n-tier architecture [6] (refer to Figure 5).

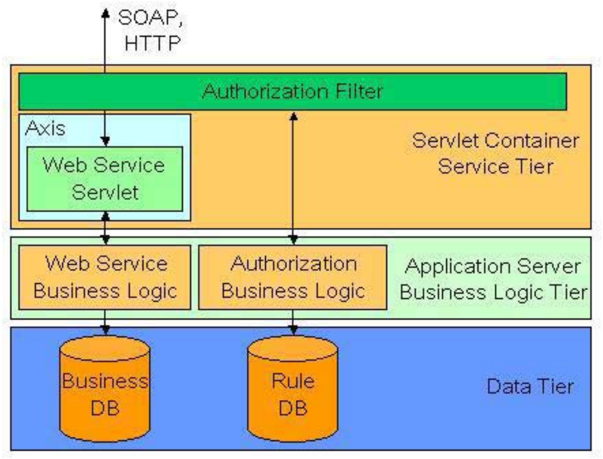
SOAP, HTTP
Authorization Filter
Axis
Web Service Servlet
Servlet Container Service Tier
Web Service Business Logic
Authorization Business Logic
Application Server Business Logic Tier
Business DB
Rule DB
Data Tier

Figure 5: Architecture overview

The framework data (service definition, user definition, access rules) is stored in a relational database (database layer). In the current implementation PostgreSQL [9] is used as database.

The logic for interpretation of the authorization rules is implemented in a separate middleware layer (logic layer). This logic is used to decide whether someone is allowed to call the service or not. The communication between authorization filter and authorization logic is not Web Service based as there is no need for Web Service technologies. Web Services provides a lot of advantages but also disadvantages like decreased performance, therefore they should only be used when necessary.

A Servlet filter [17], [16] implements the authorization client. This filter is positioned in front of the Web Service. The Servlet filter parses the Web Service request and uses the authorization components to decide whether access is allowed or not. If the user is not authorized, the request is rejected; otherwise the call is forwarded to the Web Service. The filter is tailored to the SOAP [19] protocol for Web Service access. The information contained in SOAP requests is used for access control decisions. The filter parses the SOAP request and the relevant data (service name and parameters) are used. The SOAP request does not contain information about the component, which is needed for authorization. We decided to group related Web Services through the address (URL) of the Web Services. A group of related Web Services can be accessed at the same address. Every address has its own filter, the component name is therefore configured for every filter and is not gained through the service request dynamically. It is necessary to align the component name configured for the filter and the Web Services names with the service and component names used for access control rules.

The Web Service itself is published using the Apache Axis framework [2]. This framework is based on Java Web Technology (Servlet). The integration of WSAF does not influence the Web Service implementation. As the access control entry point is implemented as Servlet filter, it is only necessary to integrate the filter into the Web application that provides the Web Service.

The filter can be used for different Web Services in parallel, the authorization logic handles parallel access to the rule base. The usage of the framework underlines the easy use and shows that the implementation is able to handle access control to Web Services and it fulfills the requirements defined within this paper.

The design and the implementation of WSAF bases on the requirements defined previously and provides a simple answer to these requirements as described in the following:

- Independent from Web Service carrier protocol
  The client component of WSAF is carrier protocol dependent, but it is easy to provide additional authorization clients for other protocols.
- Conform to Web Service protocols
  WSAF uses information contained in SOAP messages for authorization.
- Service, not object authorization
  WSAF describes services as subject.
- Abstract solution deployable to various Web Service scenarios
  WSAF uses abstract service and user definition, therefore it is applicable to other similar Web Service scenarios.
- Fine grained authorization at parameter level
  WSAF allows to define parameter restriction for services.
- Efficient authorization data access and administration
  WSAF bases on proved database technology that allows efficient data access and that ensures transactional security.
- Descriptive tailoring
  Deployment of WSAF for other Web Services means to deploy the filter for Web Services and to define rules in the database. Touching the code is not necessary.
- Easy usage and deployment
  WSAF is simple and does not solve every Web Service authorization problem, but it is easy to use and to deploy.
- Applicable to existing Web Service frameworks
  The implementation of WSAF shows the applicability.
- Multi user access to authorization data
  The data layer allows multi user access and provides transactional security.

WSAF provides a solution to these requirements, but some work need to be done in the future to improve the framework and to solve open issues.

5 Open Issues and Future Work

The presented Web Service authorization framework implements the authorization requirements for Web Service based applications defined within this paper. This enables to use this framework for various scenarios that base on Web Service technology. Nevertheless, the implementation only addresses Web Services based on the Java platform. The implementation of the framework for Web Services based on other technologies e.g. .NET is an open issue.

The design of this framework is not tailored to Web Services, but can also be applied for authorization of service-based applications. To use the framework for services, it will be necessary to implement authorization for other protocols than SOAP. This enforces new approaches for the extraction of access control information from the service request.

Another open issue is the administration of the rules. It is necessary to provide tool support for administration due to the large amount of data and due to error prone rule definition. This tool should support rule search and the definition of services and users.

The implementation of WSAF aims to handle a large amount of users in the rule base. It is an open issue to implement performance tests of authorization to measure the time needed for authorization and identify improvements of this implementation of the framework.

6 Related Work

The WSAF is not the first proposal to solve Web Service authorization, but the only one that meets the requirements defined in Section 2.

In the following we present related work and technologies that provides support for Web Service authorization.

6.1 XACML

The eXtensible Access Control Markup Language (XACML) [8] is an OASIS standard that describes a policy language and an access control decision service interface. The policy is used to describe general access control requirements, and is extensible. It is possible to define new functions, data types, combining logic, etc. The request/response language that is used for the access control service allows forming a query to ask whether or not a given action should be allowed. The response always includes an answer about whether the request should be allowed using one of four values: Permit, Deny, Indeterminate (an error occurred or some required values was missing, so a decision cannot be made) or Not Applicable (the request can not be answered by this service). XACML is currently available in version 1.1 and is not only an OASIS standard, but implementations are also available from Sun [13].

Although XACML does provide a standardized way to describe access policies and a standardized interface to a decision service, it does not fit to the requirements for authorization defined in this document. Two of these requirements are efficient and concurrent access to rule data. XML-based data does not fulfill this requirement. It is necessary to provide additional environment that supports transactional access from multiple users and that provides performant access to the rule data. Such an environment is an XML-Database that is rather expensive and rarely available [1].

One important feature of XACML is the possibility to distribute the access control decision services. The results of all the services involved in an access control decision is combined following the rules defined by XACML, so that it is possible to come to a decision. The definition of the access control decision service as Web Service leads to the question how to secure this service. It will be also necessary to provide access control to the access control decision service. Solving access control by providing an access control Web Service introduces a new point of attack for denial of service attacks.

The previously defined requirements for access control do not include the need for distributed access control services, so one of the strengths of XACML are not utilized for these common scenario. XACML tries to provide a general solution to Web Service access control; this leads to a rather complex solution. Furthermore, having the inadequate rule base for a large amount of users and the additional access control service, XACML is not the solution to the previously defined requirements.

6.2 Distributed Access Control Processor for Network Services

[12] describes in his work the design of a distributed access control processor for network services on the web. It uses a general definition of Web Services for the definition of access rules to these services. This seems to be the solution to some of the previously defined requirements. It is necessary to be able to define Web Services in a general way, so that the access rules can be defined for different Web Service applications. The disadvantage of this approach is that the description of the Web Service is needlessly complex. Although abstract solutions are applicable to more usage scenarios, it is necessary to minimize the complexity to avoid performance problems and sources of errors. Additionally, this approach does not provide a solution how to store access rules; it only addresses the theory.

The approach of [12] aims on distributed access control. Distributed access control is not a requirement for our solution; therefore the biggest advantage of this solution is not utilized. Although the whole concept is not applicable for our needs, the approach has interesting ideas that can help us to solve access control problems.

6.3 Java Security Mechanisms

Java provides various libraries addressing security issues. We will focus on solutions for access control of distributed applications, which is the application domain we address.

Java provides support for server applications by the Java Enterprise Edition (J2EE) [14]. This Java server platform provides support for declarative security. Declarative security means that security issues are not addressed directly in the application code by writing special security code. The behavior concerning security is defined in descriptors outside the code whereas the environment enforces this externally defined security policy. In fact, the runtime environment for J2EE applications, the so-called application servers, interprets these descriptors. Although this concept is a good approach for providing security to applications, J2EE security does not provide appropriate mechanisms to define fine granular access control. For example, J2EE only allows the assignment of access rules to a role on method level. Therefore access restrictions of a single user at parameter level cannot be defined with J2EE security mechanisms.

Lai et al. [3] present an extension to the standard Java platform, namely the Java Authentication and Authorization Services (JAAS). JAAS offers mechanisms for authentication and authorization, thus enabling the development of high-level security models in Java. JAAS maps users into principals. Access control evaluates, if a certain principal has appropriate permissions on an object. Principal is a common class for subjects, thus a principal can be a user, a role or any other subject-type.

The authorization itself must be done programmatic, meaning that the code contains security-relevant statements that check if a certain principal is authorized to execute a code-piece or not. This leads to the fact that the code must be touched every time the security policy changes. Furthermore, programmatic security has negative impact on reusability, since the code contains security-statements that address specific, security-relevant domain-requirements.

JAAS itself supports a role-based access control model. Other models can be implemented, using the offered mechanisms provided in JAAS. However, JAAS only offer mechanisms for building security models. Thus the implementation of security models is up to developer, resulting in an error-prone process and often customized, not standard-conform security models.

Other approaches built in into application servers or Web Servers only provide access control on service level. This means that it is only possible to define who is allowed to use which service. But it is not possible to define additional restrictions to a service access right that allows fine granular access control. In most cases this access control is done in a role-based manner [6]. Such access control mechanisms are provided by Enterprise Java Beans (EJB) [14],[4] application servers or by the .NET framework [5].

6.4 .NET Security Mechanisms

Microsoft's .NET platform provides a wide variety of security features, which enable the implementation, and integration of high-level security models into applications [5]. Unfortunately, .NET does not provide declarative security, which means that security must be addressed directly into the code by providing appropriate statements. The .NET framework uses a role-based access control model, allowing to state that specific code pieces can only be executed by certain roles. However, this model can be extended to provide a finer granularity in access control. This of course would require some additional effort from the programmer, resulting in the implementation of customized access control models. Since this can be an enormous task resulting in code pieces that are not reusable and specific to the aimed target application domain, the mechanisms are not appropriate enough in order to meet the authorization requirements defined in section 2 when providing small service based applications.

7 Conclusion

A variety of technologies, techniques and implementations are currently available that address the increasing need of security for distributed applications. But no solution supports authorization of Web Services in an easy way and no solution is available ready to deploy for simple Web Service based applications. The framework presented within this paper fills this gap. It provides a simple solution for current Web Service applications and is ready to use. This framework does not address all existing Web Service scenarios, but the most common. In our opinion it is not necessary to find a solution that solves every possible authorization problem. Providing a solution that solves the most common Web Service authorization problems without unnecessary complexity provides real value to the industry. The simplicity of Web Services was one of the main arguments for using them instead of other technologies for distributed computing. Introducing complex environment destroys this advantage.

8 Acknowledgment

The authors acknowledge support of the Kplus Competence Center Program which is funded by the Austrian Government, the Province of Upper Austria, and the Johannes Kepler University Linz.

9 References

[1] A. Chaudhri, A. Rashid, R. Zicari, XML Data Management: Native XML and XML-Enabled Database Systems, ISBN 0-201-84452-4,

Addison Wesley, 2003

[2] Apache, Web Services – AXIS, http://ws.apache.org/axis/, December 2003

[3] C. Lai, L. Gong, L. Koved, A. Nadalin, R. Schemers (1999) User Authentication and Authorization in the Java Platform. In Proc. 15th Annual Computer Security Applications Conference, Phoenix, AZ, USA, 1999

[4] E. Jendrock, S. Bodoff, D. Green, K. Haase, M. Pawlan, B. Stearns (2002) The J2EE Tutorial, ISBN 0-201-79168-4, Addison Wesley, 2002

[5] Foundstone Inc., CORE Security Technologies: Security in the Microsoft ® .NET Framework, http://www.foundstone.com/pdf/dotnet-security-framework.pdf.

[6] H. Steiert, Towards a Component-based n-Tier C/S-Architecture, ISAW3 Orlando Florida USA, 1998

[7] Jiffy Software, XACML Implementation, http://www.jiffysoftware.com/ , 2003

[8] OASIS, Extensible Access Control Markup Language (XACML) Version 1.1, Committee specification August 2003, http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf, January 2003

[9] PostgreSQL.org, PostgreSQL, http://www.postgresql.org/, January 2004

[10] R. Sandhu, D. Ferraiolo, R. Kuhn, The NIST Model for Role-Based Access Control: Towards A Unified Standard. Proc. 5th ACM Workshop on Role-Based Access Control, July 2000.

[11] R. Sandhu, P. Samarati; Authentication, Access Control, and Audit. ACM Computing Surveys, Vol. 28, No. 1, March 1996

[12] Reiner Kraft; Designing a Distributed Access Control Processor for Network Services on the Web, ACM Workshop on XML Security, November 2002

[13] Sun; XACML Implementation; http://sunxacml.sourceforge.net/ ; January 2004

[14] Sun, Enterprise Java Beans Technologie, http://java.sun.com/products/ejb/, January 2004

[15] Sun, Java 2 Platform, Enterprise Edition (J2EE), http://java.sun.com/j2ee/index.jsp, January 2004

[16] Sun, Java Servlet Technologie, http://java.sun.com/products/servlet/index.jsp, January 2004

[17] Sun, The Essentials of Filters, http://java.sun.com/products/servlet/Filters.html, January 2004

[18] W. Essmayr, S. Probst, E. Weippl; Role-Based Access Controls: Status, Dissemination, and Prospects for Generic Security Mechanisms. Electronic Commerce Research Vol 4(1), pp. 127-156. Jan. 2004

[19] W3C, SOAP Version 1.2 Part 1, http://www.w3.org/TR/SOAP/, June 2003

[20] W3C, Web Service Architecture Requirements, http://www.w3.org/TR/2002/WD-wsa-reqs-20021114#id2604831, January 2004