CERIAS Tech Report 2004-72

A TRUST-BASED CONTEXT-AWARE CONTROL MODEL FOR WEB SERVICES

by R. Bhatti, E. Bertino, A.Ghafoor

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

A Trust-based Context-Aware Access Control Model for Web-Services

Rafae Bhatti
School of Electrical & Computer Engineering, Purdue University, West Lafayette, IN
rafae@purdue.edu

Elisa Bertino
CERIAS and Department of Computer Sciences, Purdue University, West Lafayette, IN
bertino@cs.purdue.edu

Arif Ghafoor
School of Electrical & Computer Engineering, Purdue University, West Lafayette, IN
ghafoor@purdue.edu

Abstract

A key challenge in Web services security is the design of effective access control schemes that can adequately meet the unique security challenges posed by the Web services paradigm. Despite the recent advances in Web based access control approaches applicable to Web services, there remain issues that impede the development of effective access control models for Web services environment. Amongst them are the lack of context-aware models for access control, and reliance on identity or capability-based access control schemes. In this paper, we motivate the design of an access control scheme that addresses these issues, and propose an extended, trust-enhanced version of our XML-based Role Based Access Control (X-RBAC) framework that incorporates context-based access control. We outline the configuration mechanism needed to apply our model to the Web services environment, and also describe the implementation architecture for the system.

Keywords: XML, Role-Based Access Control, Trust Management, Web-Services

1. Introduction

Security in Web services is critical to their wide-scale adoption and integration in Web-based enterprise systems and softwares. The present day Web is abound with examples of Web-based enterprise services, and there is an increasing trend amongst them to migrate to the Web services platform in order to enhance and diversify the online services provided to their customers. While shifting from the traditional client-server architecture to Web services technology is seen as an endorsement of the Internet community's faith in the promise of the Web services paradigm, the goals of interoperability and ubiquity as envisioned by the Web services technology can only reasonably be realized if the unique security challenges posed by this paradigm are appropriately addressed. Among these challenges is to develop models for effective access control in dynamic XML-based Web services. The uniqueness here comes from the fact that the Web-based enterprise resources being exposed via Web services are typically dynamic and distributed in nature, and hence require adaptive access control models that can capture the dynamically changing security requirements of the target enterprise.

The mechanisms required to effectively enforce access control across distributed, heterogeneous domains are becoming increasingly complex. This complexity arises not only because of the sheer size of the distributed clientele accessing online services but also because of the fact that access control system should capture security-relevant contextual information, such as time, location, or environmental state available at the time the access requests are made, and incorporate it in its access control decisions. These context parameters capture the dynamically changing access requirements in a Web-based enterprise, and hence are critical to the effectiveness of the resulting access control scheme. The context directly affects the level of trust associated with a user, and hence the authorizations granted to him/her. These parameters constitute what is generally termed as a "user profile". The access privileges of requestors to an online service provider could be based on certain thresholds as established by the System Security Officer (SSO) based on the requestor's access patterns. If at any time, a requestor appears to deviate from his/her usual profile, the thresholds (i.e. the trust level) would automatically be reduced as a precaution to prevent a

potential abuse of privileges. This is a real-time requirement, and is exceedingly important in dynamic Web services serving thousands of customers with diverse activity profiles. In order for the access control to be effectively exercised in such scenarios with context-sensitive access requirements, the traditional access control models must be extended to make them context-aware. To this end, we propose to employ the generalized temporal extension to our X-RBAC [1] model, the XML-based Generalized Temporal Role Based Access Control (X-GTRBAC) model [2]. X-GTRBAC was originally proposed as a solution to enterprise-wide access control, but due to its XML-based framework, it can also be configured to provide access control in Web services. In Section 3, we introduce the reader to the X-GTRBAC model and outline the mechanism to extend X-GTRBAC as a context-aware access control framework for Web services environment.

Another issue we highlight in the paper is trust-based role assignment to users. There are different (although related) notions of "trust" in the literature. The one that is relevant to our purposes is the level of confidence associated with a user based on certain certified attributes thereof. In our framework, this level of confidence is not quantitatively reported. Instead, we rely on the Trust Management (TM) approach of trusted third parties (such as any PKI CA[1]), and use the certification provided by them to assign roles to users. We derive our motivation for doing that from the review of traditional access control schemes that have adopted either an identity or capability-based approach to authorize users [1, 3-7]. Such mechanisms do not scale well to the distributed Web services architecture, and hence would cause a significant burden to be attached to the enforcement of the access control scheme. This is because each credential needs an explicit delegation act by the respective domain administrators. In order to overcome this limitation, we outline a mechanism to incorporate trust in X-GTRBAC model in Section 3. In particular, we would use TM credentials (i.e. certificates) to allow trust establishment amongst distributed domains.

The remainder of this paper is organized as follows. We begin by providing a compendium of related work in the area of Web services security, and discuss how our framework aligns with the existing security architectures. We also review the features provided by existing Web-based access control schemes, and their suitability to Web services. We next introduce our trust-based context-aware access control model, which is based on a temporal extension of X-RBAC with trust domains incorporated into it. The paper concludes with the discussion of implementation architecture of our model and an overview of future research goals.

[1] Public Key Encryption Certification Authority

2. Background and Related Work

We shall now provide a background and compendium of current state of the art in Web services security. A fair amount of related research in this area is due to the industry, with standards such as Security Assertion Markup Language (SAML) [8] and eXtensible Access Control Markup Language (XACML) [9] being recently adopted. SAML defines an XML framework for exchanging authentication and authorization information for securing Web services, and relies on third-party authorities for provision of "assertions" containing such information. XACML is an XML framework for specifying access control policies for Web-based resources, and with significant extensions can potentially be applied to secure Web services. The XACML specification supports identity-based access control and incorporates some contextual information, such as location and time, into access decisions, without any formal context-aware access control model. There also are other emerging specifications, most notable amongst them are the ones outlined in WS security roadmap [10]. The roadmap consists of a number of component specifications, the core amongst them are WS-Security, WS-Policy, and WS-Trust. WS-Security is similar in intent and purpose to SAML, only uses a different technology. WS-Policy is used to describe the security policies in terms of their characteristics and supported features (such as required "security tokens", encryption algorithms, privacy rules, etc.). WS-Trust defines a trust model that allows for exchange of such security tokens (using mechanisms provided by WS-Security) in order to enable the issuance and dissemination of credentials within different trust domains, and establish online trust relationships. The models proposed in the roadmap have been directed primarily at the authentication aspect of Web services security, with an emphasis on designing secure messaging protocols to communicate the security-relevant information, such as security tokens and characteristics of security policy. The specification leaves room for custom authorization models to be tied into the architecture at the appropriate (i.e. WS-Policy) level. In this paper, we present an access control model that is capable of doing exactly that; our XML-based framework allows easy integration into the existing XML-based

architectures for Web services security, while providing an effective authorization mechanism suitable for Web services environment.

There has been an effort in the research community to highlight the challenges in Web-based access control within the XML framework, including both the initial DTD-based solutions [3-6], and the more recent schema-based approaches [1, 7]. In [1], we have presented X-RBAC, an XML-based RBAC policy specification framework for enforcing access control in dynamic XML-based Web services. X-RBAC was designed to readily integrate within the XML framework, and emphasized simple, yet effective, administration through the use of RBAC. We also maintained that X-RBAC includes a comprehensive set of features that is comparable to the related access control schemes cited above, and is targeted for the Web services environment. Although X-RBAC and related schemes provide viable solutions, there remain issues that impede the development of effective access control models for Web services environment. Amongst them are the lack of context-aware models for access control, and reliance on identity or capability-based access control schemes. We next elaborate upon these issues, and propose an extended and trust-enhanced version of our X-RBAC model in an attempt to address them.

3. Trust–Enhanced X-GTRBAC Model

This section begins with an introduction to the X-GTRBAC model. It then describes the mechanism to configure X-GTRBAC to provide context-aware trust-based access control in Web services.

3.1. X-GTRBAC- An Introduction

The X-GTRBAC framework is based on Generalized Temporal Role Based Access Control (GTRBAC) model [11]. X-GTRBAC augments GTRBAC with XML to allow for supporting the policy enforcement in an heterogeneous, distributed environment. GTRBAC extends the widely accepted Role Based Access Control (RBAC) model proposed in the NIST RBAC standard [12]. RBAC uses the concept of roles to embody a collection of permissions within an organizational setup. Permissions are associated with roles through a permission-to-role assignment, and the users are granted access to resources through a user-to-role assignment [13]. GTRBAC provides a generalized mechanism to express a diverse set of fine-grained temporal

Table 1. Salient Features of X-GTRBAC

Element Type	Element Name	Purpose
RBAC Element	*XML User Sheet (XUS)*	Declares the users and their authorization credentials
	XML Role Sheet (XRS)	Declares the roles, their attributes, role hierarchy, and any separation of duty and temporal constraints associated with roles
	XML Permission Sheet (XPS)	Declares the available permissions
RBAC Assignments	*XML User-to-Role Assignment Sheet (XURAS)*	Defines the rules for assignment of users to roles; these assignments may have associated temporal constraints
	XML Permission-to-Role Assignment Sheet (XPRAS)	Defines the rules for assignment of permissions to roles; these assignments may have associated temporal constraints
RBAC Constraints	*XML Separation Of Duty Definition Sheet (XSoDDef)*	Defines the separation of duty constraints on roles
GTRBAC Constraints	*XML Temporal Constraint Definition Sheet (XTempConstDef)*	Defines the temporal constraints on role enabling and activation; also defines temporal constraints for user-to-role and permission-to-role assignments
	XML Trigger Definition Sheet (XTrigDef)	Defines context-based triggers for invocation of periodic events subject to associated constraint evaluation
Authorization Credentials	*XML Credential Type Definition Sheet (XCredTypeDef)*	Defines the available credential types

constraints on user-to-role and permission-to-role assignments in order to meet the dynamic access control requirements of an enterprise. X-GTRBAC allows specification of all the elements of the GTRBAC model. These specifications are captured through a context-free grammar called X-Grammar, which follows the same notion of terminals and non-terminals as in BNF, but supports the tagging notation of XML that also allows expressing attributes within element tags. The detailed specification of these elements of X-GTRBAC framework can be found in [2]. Table 1 enlists the salient features of the model.

We now describe the mechanism to configure X-GTRBAC to provide context-aware trust-based access control in Web services. Toward that end, we need to outline a set of formal specifications to capture contextual information, and illustrate how it can be incorporated within the access control model. In addition, we would need to provide an interface to the system to accept TM credentials instead of its usual user credentials as the basis of privilege assignments.

3.2 Context-aware access control

This section defines the set of specifications needed to configure X-GTRBAC for context-aware access control in Web services environment. We base our set of specifications on a tuple language that can be readily mapped onto our existing XML-based framework. In the following, we provide the formal definition of *context*, and then use that to provide the definition for a *service_access_request*. In order to formalize the context, we introduce a type system to allow specifying domains of legal values for various context parameters. Our formal model relies on the components we define below:

Parameter Name Set: A set `PN` to denote the possible names of context parameters
Parameter Type Set: A set `PT` to denote the possible types of context parameters
Context Parameter: A context parameter is represented by a data structure `p`, having the following fields: `name ∈ PN`, `type ∈ PT`, and a function `getValue()`.
Roles Set: `RR` = $\{rr_1, \ldots, rr_k\}$, where $\mathbf{rr}_i$, $1 \le i \le k$ is a regular role[2] in GTRBAC
Operations Set: `RO` = $\{ro_1, \ldots, ro_k\}$, where $\mathbf{ro}_k$, $1 \le i \le k$ is a regular operation in GTRBAC

[2] We introduced the term regular role in [17] to differentiate it from an administrative role.

Service: A service is an abstraction of the operations provided by the system on its resources. Formally, a service is a subset of the data set `RO`, and is designated by the service name **`srv`** that is defined according to the `wsdl:service` element of the WebServices Description Language (WSDL) document
Services Set: `SRVS` = $\{srv_1, \ldots, srv_k\}$, where $\mathbf{srv}_i$, $1 \le i \le k$ is a service.

Definition 1: (*Context*): A context set `C` consists of `n` context parameters $\{p_1, \ldots, p_n\}$, $n \ge 0$, s.t. for any p_i, p_j, with $i \ne j$ and $1 \le i, j \le n$, we have that p_i`.name` $\ne$ p_j`.name` (i.e. the parameter names must be distinct).

We mention here that `PN` and `PT` constitute a set of pre-specified parameter names and types determined by the SSO. For example, the set `PN` may be defined as: `PN ={time_of_day, location, duration, system_load}`, with the corresponding set `PT` defined as: `PT ={Time, String, Long, Integer}`. The `p.getValue()` function is used to dynamically compute the value of the parameter, and its implementation is system-dependent. For built-in system parameters, such as `time_of_day`, it might just serve as a wrapper around system functions such as getCurrentTime(). The dynamic mechanism to compute parameter values especially helps in the case of mobile users accessing Web-based services, because in such environments the parameter values are constantly changing and may need to be re-evaluated at certain intervals. Additionally, for dynamic access constraints, such as duration, `getValue()` would be called periodically to ensure that the constraint is always satisfied. We also note from the preceding definition that the context may be an empty set.

Definition 2: (*Service_access_request*): A service access request is defined as a triple `<role, srv, context>` where `role ∈ RR`, `srv ∈ SRVS`, and `context` is defined according to Definition 1 and captured dynamically at the time of the access request.

Based on the *service_access_request*, the system determines the applicable *access policy* for the requested service. This policy will be based on a set of constraints on the role and service name, and evaluated in conjunction with the available contextual information to enforce fine grained access control. An access policy consists of a collection of access conditions. In order to formulate an access condition, we refer to the notion of parameterized roles of [2].

Parameterized roles are roles supplemented with *role_attributes*. The attributes set A of a role contains a collection of contextual attributes (such as time, location or system load) that may be used to define context-based conditions on roles. The values of these attributes are specified by the SSO, and these values are compared with the values of the supplied context parameters in order to evaluate an access request. The set of contextual attributes of a role is hence a subset of the set C of context parameters, and follows the same type system. We formally define the *access_policy* below. We assume the existence of a function getAttributesSet(role), which returns the set A for a given role.

Definition 3: (*Access Policy*): Let $r \in$ RR be a role, $srv \in$ SRVS be a service name denoting a service. The access policy AP for a (*r*, *srv*) pair is a set of clauses, where each clause is a Boolean combination of expressions. An expression is of the form <attr □ val> where attr is a role attribute s.t. attr ∈ getAttributesSet(*r*), val is the value of the parameter as specified by the SSO in order for role *r* to access the service *srv*, and □ is any relational comparison operator.

It may be mentioned that we have intentionally kept our model generic enough, as it is unlikely for any one model to capture all types of contextual information and associated conditions that might arise in practice. But for most practical purposes, the sets PN, PT and *role_attributes* may be extended according to the system requirements in order to define access conditions based on appropriate context parameters.

We now give the following set of algorithms to evaluate a *service_access_request*.

The ComputeAccess algorithm works as follows. In Step 1, the clauses corresponding to the (role,srv) pair are retrieved from the AP into a dynamic array CL. Step 2 retrieves the attributes of the role into a dynamic array A. In Step 3, the algorithm loops over the array CL and calls the routine getDecision() for each of the clauses. Each clause has potentially multiple expressions, and so each expression is evaluated using the evaluateExpr() routine. For each expression, this routine retrieves the attribute from the attribute array A and then calls the routine checkCondition() to evaluate the conditions corresponding to this role attribute. This routine loops over the set C of supplied context parameters and finds the matching context parameter for this role attribute by calling the match() routine, which internally compares the name and type of the two entities. Since the set A is a subset of set C, this search always results in a match. When a match is found, it compares their values according to the operator specified in the AP. If the condition is satisfied, a value of true is returned to

```
Algorithm: ComputeAccess
Input: role, srv, C     //C is context array
Output: decision d , d ∈ {YES, NO, PENDING, N/A}

1: CL[] = getClauses(role,srv)
2. A[] = getAttributesSet(role)
3. FOR i = 1 to length(CL) DO
     clause = CL[i]
     access = getDecision(clause,A,C)
     IF access = false
        return NO
4: IF access = true
        return YES
```

```
Algorithm: getDecision
Input: clause,A,C
Output: result          //boolean

1. initizalize(result[])
2. FOR i = 1 to size(clause) DO
     expr = clause.getExpr(i)
     result[i]=evaluateExpr(expr,A,C)
3. return computeResult(result[])
```

```
Algorithm: EvaluateExpression
Input: expr, A, C
Output: result                  //boolean

1. name = expr.getAttrName()
2. attr = getAttribute(A,name)
3. result = checkCondition(attr, C)
4. return result
```

```
Algorithm: checkCondition
Input: attr, C
Output: result              //boolean

1. p=match(C,attr)
   IF p.getValue() □ val
      result = true
   ELSE
      result = false
2.: return result
```

getDecision. After the result for all access conditions within the clause has been computed, the getDecision routine then computes the overall result for the clause and returns it to ComputeAccess. If any of the clauses evaluates to **false**, a NO is returned as the output of the ComputeAccess algorithm, because the overall access decision is a conjunction of all individual clauses. Otherwise, after the loop terminates successfully over all the clauses, a YES is returned. Other decisions such as PENDING or N/A may also be returned by incorporating system-specific logic into the algorithm.

As an illustration, consider the example of a recently launched initiative of a German insurance company [14]. The company leverages Web services technology to introduce online visitors to its services, and allows them to purchase insurance coverage through an entirely digital process. The evaluation of an online coverage request requires several kinds of personal information to be made available, and the same needs to persist in the company's database for a subsequent evaluation of an insurance claim. At that point, however, the access to the customer's resources should only be granted after establishing the fact that the requestor indeed is "the" genuine customer. For instance, assume that the following *service_access_request* is submitted for evaluation to the system:

<**role**=priv_cust,
service_name="review_claim",
context={p_1{time,12PM},
p_2{location,"WashDC"},p_3{duration,
0},p_4{system_load,"low"}>.

This request says that a user belonging to the priv_cust (privileged customer) role has requested to review an online insurance claim through the Web-based review_claim service offered by the company. The context recorded at the time of access request is provided to the system as part of the request. Note that duration is initialized to 0 because the access has not yet started. Now, assume that the following AP is applicable to the given (role, srv) pair:

{< CL1> , < CL2 >, < CL3> , < CL4 >}

s.t.

CL1: {time > 9AM} AND {time < 5PM)
CL2: {location = "WashDC"} OR {location = "NewYork"}
CL3: {system_load != "high">
CL4: {duration ≤ 600s}

Based on this information, the system would return an authorization decision for this *service_access_request*. The available contextual information indicates that the access conditions are satisfied. In addition, due to the duration constraint specified for the requested service and enforced by the dynamic temporal constraint mechanism of GTRBAC, the access duration of the user is continuously monitored, and any violation thereof is detected on a per-user basis by the GTRBAC Processor (see Table 2). The mechanism to deal with the violation is system-specific, but GTRBAC allows a trigger mechanism to take immediate actions in such situations (such as de-activating the role for the given user). Detailed discussion on such mechanisms can be found in [2].

3.3 Incorporating trust domains

In this section, we briefly describe a mechanism to incorporate trust domains in X-GTRBAC to enable effective access control in a distributed environment, where user identities are not known a-priori. Since X-GTRBAC makes the access decisions based on the eligible roles for known users, we can use TM credentials to assign roles to users. While it is sometimes viewed as appropriate in TM to adopt a direct authorization model, i.e. to combine authentication and access control into one authorization step [15], we would like to motivate here that the indirection through roles helps scalability and flexibility in the case of large scale open systems, especially Web services. Hence, a significant advantage that accompanies the role-based approach adopted in our framework is that of simplified authorization administration [13]. An earlier approach that merged features from TM and RBAC, called the Role based Trust management framework (RT), was reported in [16]. However, our primary goals are different from RT. The latter is primarily a TM credential exchange and distribution mechanism to assist authorizations in a distributed environment; it does not support an elaborate access control scheme beyond the basic permission-to-role assignment mechanism in RBAC. We focus on providing a context-aware access control model for the Web services environment, and rely on TM credentials for determining the trust level (i.e. role) associated with a user. As mentioned in the introduction, the trust level can be subsequently adjusted based on the user's activity profile. Such a profile can be maintained by logging the contextual information associated with the invocation and acceptance of a *service_access_request*.

Table 2. Description of X-GTRBAC system modules

Module Name	Description
Document Composition Module (DCM)	Used to compose the policy documents; contains the XML-Policy Base that serves as the document repository; is an external component of the model
Policy Loader	The interface of the system to the DCM; used to load the XML policy files into the system
Policy Validation Module	Validates the XML policy files for existence checking and type conformance according to policy rules; XML syntax validation is also implicitly done
XML processor	Contains an XML parser that generates the DOM tree representation of the XML policy files
GTRBAC Processor	Contains a GTRBAC Module that translates the DOM tree representation to internal RBAC data structures representing the system state at any time; maintains logs of sessions and updates the system data structures to allow contextual information to be incorporated in access decisions

The use of TM credentials to establish role memberships of users requires the X-GTRBAC model to be adapted to accept distributed TM credentials. We touch upon the mechanisms needed to do this in the next section, but leave an elaborate treatment of the same for some future work, as it is not the focus of our current paper. It may be noted here that the trust-based approach to verifying user credentials effectively adds authentication support to our existing authorization model.

4. Implementation Architecture

There is an on-going effort underway on extending our implementation prototype, first reported in [1]. The major components of X-GTRBAC system architecture are summarized in Table 2. The existing prototype incorporates the temporal constraint enforcement mechanism as per the GTRBAC model. The generalization of the contextual information to include parameters other than time as described in the paper is being incorporated into the system. We are also working toward a set of specifications that would allow us to substitute the existing credential evaluation mechanism with that involving TM credentials. Because of the modular design of X-GTRBAC, this task can be accomplished with only slight modifications in the overall architecture. The components affected would be (i) the XML Policy Base, since it would now need to store a different XCredTypeDef sheet based on TM credentials, and (ii) the XML Processor, since it would now employ a different evaluation logic for processing credential declarations. Our set of specifications would be XML-based, and hence can be expected to integrate well with the existing framework.

5. Conclusion

In this paper, we have outlined a mechanism to develop a trust-based, context-aware access control model for Web services based on the X-GTRBAC framework. X-GTRBAC is a temporal extension of the earlier X-RBAC model for access control in Web services. The mechanism presented in the paper extends X-GTRBAC to support context-aware access control based on both temporal and non-temporal contextual conditions. In addition, we outline a mechanism to incorporate trust domains into X-GTRABC by the use of distributed TM credentials for unknown users. Such an approach effectively adds authentication support to our system. We have discussed the configuration of X-GTRBAC for its application in Web services environments, and also proposed extensions to our current implementation architecture for the purposes outlined in this paper. We intend to report the detailed results of our on-going implementation efforts in some future work. We also plan to explore the interplay of contextual conditions in the presence of separation of duty constraints and role hierarchies. In these situations, it is critical to ensure that the access to services based on inherited permissions do not violate any separation of duty constraints. Another future direction of research would be to investigate the suitability of the proposed administration model for X-GTRBAC [17] to Web services. We expect to see our framework evolve with time, as Web services standards are continually being

enhanced, and would likely incorporate additional security mechanisms such as secure messaging and transaction support into our system. Along related lines, it would be desirable to design a framework to evaluate security properties of a Web service based on the existing and emerging Web services specifications.

Acknowledgements

Portions of this work have been supported by the sponsors of the Center for Education and Research in Information Assurance and Security (CERIAS) at Purdue University, and the National Science Foundation under NSF Grant# IIS-0242419.

6. References

[1] R. Bhatti, J. B. D. Joshi, E. Bertino, A. Ghafoor, "Access Control in Dynamic XML-based Web-Services with X-RBAC", In proceedings of The First International Conference on Web Services, Las Vegas, June 23-26, 2003.

[2] R. Bhatti, "X-GTRBAC: An XML-based Policy Specification Framework and Architecture for Enterprise-Wide Access Control", Masters thesis, Purdue University, May 2003. Available as CERIAS technical report 2003-27.

[3] E. Bertino, S. Castano, E. Ferrari, "Securing XML Documents with Author X", IEEE Internet Computing May-June 2001.

[4] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, "A Fine Grained Access Control System for XML Documents", ACM Transactions on Information and System Security, Volume 5, Issue 2, May 2002.

[5] N. N. Vuong, G. S. Smith, Y. Deng, "Managing Security Policies in a Distributed Environment Using eXtensible Markup Language (XML)", Symposium on Applied Computing, March 2001

[6] S. Hada, M. Kudo, "XML Access Control Language: Provisional Authorization for XML Documents", October 16, 2000, Tokyo Research Laboratory, IBM Research.

[7] X. Zhang, J. Park and R. Sandhu, Schema based XML Security: RBAC Approach, IFIP WG 11.3 2003.

[8] OASIS, Security Services TC http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[9] XACML 1.0 Specification http://xml.coverpages.org/ni2003-02-11-a.html

[10] Security in a Web Services World: A Proposed Architecture and Roadmap http://www106.ibm.com/developerworks/securiy/library/ws-secmap/

[11] J. B. D. Joshi, Elisa Bertino, Usman Latif, Arif Ghafoor, "Generalized Temporal Role Based Access Control Model (GTRBAC) (Part I) - Specification and Modeling", Submitted to IEEE Transaction on Knowledge and Data Engineering. Available as CERIAS technical report 2001-47.

[12] D. F. Ferraiolo , R. Sandhu , S. Gavrila , D. R. Kuhn , R. Chandramouli, "Proposed NIST standard for role-based access control", ACM Transactions on Information and System Security (TISSEC), Volume 4, Issue 3, August 2001.

[13] R. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, "Role Based Access Control Models", IEEE Computer Vol. 29, No 2, February 1996.

[14] Accenture Web Services Case Study http://www.accenture.com/xd/xd.asp?it=enweb&xd=services\microsoft\case\micr_ergo.xml

[15] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote Trust Management System, version 2. IETF RFC 2704, September 1999.

[16] N. Li, J..C. Mitchell, and W. H. Winsborough. Design of a role-based trust management framework. In Proceedings of the 2002 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, May 2002.

[17] R. Bhatti, J. B. D. Joshi, E. Bertino, A. Ghafoor, "X GTRBAC Admin: A Decentralized Administration Model for Enterprise Wide Access Control", In proceedings of 9th ACM Symposium on Access Control Models and Technologies, New York, June 2-4, 2004.