Schema-Driven Security Filter Generation For Distributed Data Integration

Hui Dong, Zhimin Wang, Robert A. Morris
Department of Computer Science
University of Massachusetts at Boston
Boston, MA 02125
Email: hdong1,wangzm,ram@cs.umb.edu

Douglas Sellers
CustomInk, LLC
7900 Westpark Dr. Ste T500
McLean, VA 22102
formerly at NatureServe

***Abstract*— With the prevalence of loosely coupled web service composition in the service oriented infrastructure, distributed security management has become important to offer flexible web security. In our project[1] affiliated with NatureServe [19], we developed a XACML [11] policy-oriented security system to provide trusted data service to 75 network partners in the Natural Heritage Program [24]. In our model, a security filter based on a data integration schema is generated dynamically and applied to the returned document to satisfy the security constraints defined both by each partner and by the NatureServe service mediator. Challenges of resolving the policy conflicts in the distributed environment are addressed.**

I. Introduction

We describe an access control system for a project directed by NatureServe [19] for the distributed service of biodiversity data served by the 75 partners of the NatureServe Natural Heritage Network members. Until this project, each provider has annually updated its data in a central repository. In the new arrangement, data will be served directly from the source in a distributed fashion to provide the most up to date possible to key decision makers. This distributed version will be rolled out incrementally with various providers coming on line and their data eliminated from the central store, which may, however, provide indexing and caching services in the model of the Global Biodiversity Information Facility (www.gbif.org). Each Natural Heritage partner will provide data conforming to two specific schemas. One describes the attributes of species. The other provides attributes of biodiversity populations – surveys of species and natural communities and ecosystems – in set places according to set monitoring protocols designed by NatureServe and the partners. It is important for many uses of this data that the exact location (latitude and longitude or GIS polygons) of the populations be provided, but for some rare or endangered species, this location must be revealed in detail only to authorized requestors in order to prevent collectors from harvesting them, or from persons with commercial or other interests from destroying them in order to avoid land restrictions due to laws protecting those species. Further, some observations are made on private land with the landowner promised a high degree of insulation from curious viewers who might visit their land without permission. In all cases, the requirement is not to hide the location completely, but rather to change the spatial resolution of it, e.g. to report only the county in which the population is found. This paper describes the access control system we have provided for the system. In many cases, we describe facilities that will be needed when the system is fully distributed operationally, and are not needed in the currently envisioned implementation. For example, the access control system we describe will support data providers delegating control on certain data to certain other providers to assist applications of regional collaborations with shared access to sensitive data, but this functionality will not be enabled in the initial rollout.

With the rapid development of the web and related technology, Web Service paradigms and standard protocols have become the key approach in distributed environments to interrelate services through well-defined contracts between different parties across the network. As demand increases for online automated data integration, XML has played an important role in data representation, exchange and information dissemination in Web Service composition. This interoperability infrastructure allows service providers to publish and aggregate new services more easily based on business flows, however, it also causes new security concerns to protect sensitive data against unauthorized users. In the distributed environment, different data providers have their access rules linked to their data items, and trusted data integration services (TDS) will fetch data sets from different providers and send back clients' response documents compliant to the TDS' own integration schema. The data mediator has to ensure the selective exposure of the sensitive information only to classes of users who satisfy the security characteristics and qualifications defined by both the original information provider and the integration service provider. As in other common distributed data integration scenarios, NatureServe TDSs are exposed as web services and published by using standard interface descriptions using the Web Service Description Language (WSDL)[25]. Query results are encapsulated in a document compliant to an XML integration schema and security policies are applied to the resulting document. The TDSs and related security services which it invokes, are written in Java and exposed with the Apache Axis web service framework [26], each supported by a well-defined Java API. One consequence of this is that various

[1]This work is supported in part by NSF grant 0345400 to NatureServe.

1-4244-0596-3

components may be deployed locally in flexible composites reflecting which resources can be most effectively be accessed from a single process without incurring web service overhead. In a fully distributed environment, this may vary with the participating organizations. In the NatureServe installation, the orchestration of this is controlled in part by the Spring dependency injection framework [27]

A number of research efforts have schemes for document centric access controls. Bertino et al. [1] have studied issues like access granularity and role hierarchies. Damiani et al. [2, 3] have proposed to use XPath queries to express policies and a quite complicated way to materialize and maintain views. Fan et al. [4] came up with a DTD-based control model and view-based query rewriting. Generally, a document view-based query control mechanism sets limits on query ability and only allows a user to receive data meeting her corresponding trusted query level. For example, in the NatureServe project most data is not sensitive, and there is a requirement that unauthenticated users may see this data, but never get back data containing sensitive elements. In this scenario, both privileged and unprivileged users must be allowed the same queries, but not the same results. Moreover, in the distributed environment, a user may be registered with different data providers within the NatureServe Natural Heritage Program and assigned different roles for each. In such a case, exposing different view schemas may lead to extra information leakage and increase the security breach. Hoda et al. [5] and Cho et al. [6] assumed that access permission annotation is explicitly included in the data elements, which are hard to maintain for massive collections. Bhatti et al. [7] and Feng et al. [8] proposed a role-based architecture to address authorization administration. Miklau [9] proposed a secure XML document delivery using encryption. Kudo et al. [10] presented an extension to access control authorization by introducing provisional actions with XML document access. Many of these approaches are adequate in a data warehouse scenario, but do not meet our distributed data requirements. For example, the same sensitive information may be tagged with different granularity by different providers. Even when a single return schema is in place, as with the current deployment (but not required by our architecture) the protection requirements for the same data items might be different at different partners and a single enforcement mechanism such as encryption does not satisfy the requirements in these complicated integration scenarios. Furthermore, data access control delegations between partners needed to be addressed to facilitate regional collaborations and data sharing.

The goal of our work is to develop a security model that can accommodate to fast changing data integration scenarios. Data providers can easily change access rules for different data based on their own concerns, and integration service providers can modify the integration schema and published service interfaces and must be able to do so without affecting each other. The model should protect data at varying granularity, and also make decisions and enforce protection based on content as well as structural transformation requirements. The decision process should distinguish policies from different data providers and address possible policy conflicts. The overall authentication system should fulfill the requirements of a distributed computing environment and scale well. To achieve these goals, we use XACML [11] as our policy language to specify the access control mechanism over resources. Based on policies and the integration schema, a security filter is generated at run time. The generated filter is in XSLT [12] format and performs required structure transformation to deliver the return document satisfying all security requirements.

The remainder of this paper is organized as follows. Section 2 reviews XACML. Our security model is defined in section 3. In Section 4 we present the filter generation algorithm and discussion. In Section 5 we discuss how we handle possible policy conflicts in a distributed web service environment. Section 6 concludes the paper.

II. The eXtensible Access Control Markup Language

XACML is an OASIS standard which describes a general-purpose policy language and a protocol that defines the syntax of XACML compliant request and response. The policy language is used to describe general access permissions over classified information in a syntax described in a core schema [14]. XACML defines sets of standard datatypes, functions and combination algorithms, and it also supports extension mechanisms to allow users to integrate customized security control operations. The common practice of the XACML model is that some service wants to authorize an action on a resource. The original request will be sent to a module called a Policy Enforcement Point (PEP). The PEP will fetch all the related information (the role, the resource, the action, etc.) and create a XACML request message and forward it to a Policy Decision Point (PDP). Once the PDP receives the request, it will retrieve respective policies and environment attributes from a policy repository and attribute authority (which together comprise a Policy Administration Point (PAP)) to make a decision on what kind of access may be granted, together with optional obligations to be imposed on the enforcer. This answer will be sent back to the PEP for enforcement of the decision.

In a Web Service environment, every data provider can define its own rules to control access.. Using a standard language, such as XACML, insures compatibility and interoperability during policy exchange, combination and service aggregation. In Section 3 we describe how we describe and show how we instantiate XACML's abstract representation of policies. The core modules of the XACML model, the PDP and the PEP can be services that are accessed in a distributed way. Different combinations of those modules (central PDP/distributed PEPs, or distributed PDPs and PEPs, etc) can satisfy complicated scenarios in a distributed computing environment and can make service aggregation easier and more seamless. Furthermore, policies about a single resource class can be distributed over the network and different groups can manage the sub-pieces of the rules by themselves. For example different state agencies

may have variant access definitions for a protected species, hence they may wish to manage their own rule sets as needed. The PDP will fetch references to all those sub-pieces at query time and provide a decision for the PEP to enforce the required security constraints. The XACML extension mechanism is a powerful feature to allow the security system to support more generic usage other than this specific case. A role-based hierarchy and standard profiles using SAML [22] and LDAP [23] are proposed on the XACML site [28]. Several systems like Cardea [15], PRIMA[16] and Shibboleth[17] have been developed to provide web based authentication and authorization systems over different contexts.

III. Access control model with Security Filters

In this section, we present our filter-based security model for integrated web services. We define objects and relationships for our access specification and illustrate the architecture to address challenges in web service environments.

A. Security Model

Our XACML-based security specification is mainly composed of a set of security policies defined by a set of different data providers. Each XACML policy can be abstracted as a quintuple (subject, object, action, decision, obligation). In our fully distributed model, the "subject" is selected from a set of roles defined within the network. Roles are divided into local roles and network roles, and are maintained in a hierarchical structure. For example the "New England scientist" might be a subrole of the network role "HeritageGroup". Typically a user is registered with some data providers, and corresponding roles are assigned. The association between the users and roles is many to many, and a decision for query enforcement will be based only on the roles included in the policy descriptions. A policy "object" refers to the set of data items needed to be classified. In our model, every object is a resource identified by an XPath expression valid for a specified XML schema. Other entities and relations include a set of actions(such as "read" or "write") operable on data, a set of permissions(such as "deny" or "permit") used in policies and a set of obligations(such as "truncate" or "substitute") linked with decisions to protect sensitive information in the way a provider wants.

In our specification, we don't directly connect the role and permission together as typical role-based access control (RBAC) model [18] does, because a decision actually considers the role together with a specific resource and action as well as other related background information and conditions. Our RBAC-aware XACML access control model offers protection for different information aspects and is flexible to fit complicated requirements in a distributed environment. First, all the target resources for security manipulation are defined at the schema level; no direct data is involved. An XPath-based node selection scheme enables access rules to refer to various kinds of information items in the document, which can be either an XML element or an attribute. This resource indexing scheme is flexible enough to support a spectrum of protection granularity levels. The separation of permission assignments from instance data items also protects the integration service from instance data scheme mismatch issues. Generally every resource expression in the integration schema reflects a set of combined data from different partners. Those data items are modeled by different schemas at their own location and need not be equivalent from provider to provider. Thus policies are easy to maintain with integration schemata. As we described earlier, each data provider will maintain policies on their own data, and the integrated service can maintain a local view of schema mapping to generate policies only applied to the corresponding integration schema. On the other hand, the trusted data integration service can also allocate the integration schema to every partner and allow policies to be specified in the view of its global service schema. This can increase the performance by saving policy translation time. In the current rollout of NatureServe services, there is in fact a single global integration schema, but a fully distributed mechanism is required to support fully distributed data provision in subsequent instances of the system. Another advantage of separating policy permissions from instance data is that sensitive items may appear at different elements of the returned document and at different depths of the document tree if recursive definition is provided, and may be referenced indirectly by XML IDrefs or KeyRefs. All those cases can be well addressed using an XML Schema model with XPath expressions to define resources.

Our model also supports context-aware protection and allows the PEP to enforce different operations to satisfy the security requirements. In particular, different service providers may have different data. Access rules for users from different security realms. For example, a provider in Massachusetts may allow only researchers from New England to fetch the exact geographic location of rare species populations, but allow all users to extract the location of invasive species. In our model, context-aware PDPs correlate conditions with policy issuers to make the right decision. Typically a decision to deny access means no information is released to users. However, by supporting obligation extensions, our PEP allows various manipulations of the original data sets and transforms the document in different ways to meet application needs. For example suppose that in the original integrated document, a species distribution in New Mexico (NM) includes a county Rio Arriba and an exact location with detailed geographic coordinates given as a polygon of coordinate points (Figure 1.a). A "deny/elimination" rule will simply truncate the sensitive node (Figure 1.b); A deny/encryption rule will encrypt the sensitive information and only a user or downstream service having correct keys can decrypt and recover the data back (Figure 1.c); A "deny/substitution" rule meets the case where the client has the county level data access privilege to all the integrated data and converts the polygon to a county-based element (Figure 1.d) by appeal to external GIS software and digital gazetteers. Different users and consumer services get a different sanitized view of the original integrated data.

```
<SpeciesDistribution>
    <State code ="NM">
    <County name ="Rio Arriba" zipcode ="35039"/>
    <Polygonid="p23" lat= "35.81N" long ="106.61 W" />
    </ State>
</SpeciesDistribution>
```
(a)

```
<Statecode ="NM">
    <County name ="Rio Arriba" zipcode ="35039"/>
</State>
```
(b)

```
<State code ="NM">
    <County name ="Rio Arriba" zipcode ="35039"/>
    <Polygon id="p23" lat= "bAf32t" long ="Ct47erf" />
</State>
```
(c)

```
< State code="NM">
    <Countyname="Rio Arrba" zipcode = "35039"/>
    <Countyname="Sandoval" zipcode = "35043"/>
</ State>
```
(d)

Fig. 1. different access manipulation with obligations

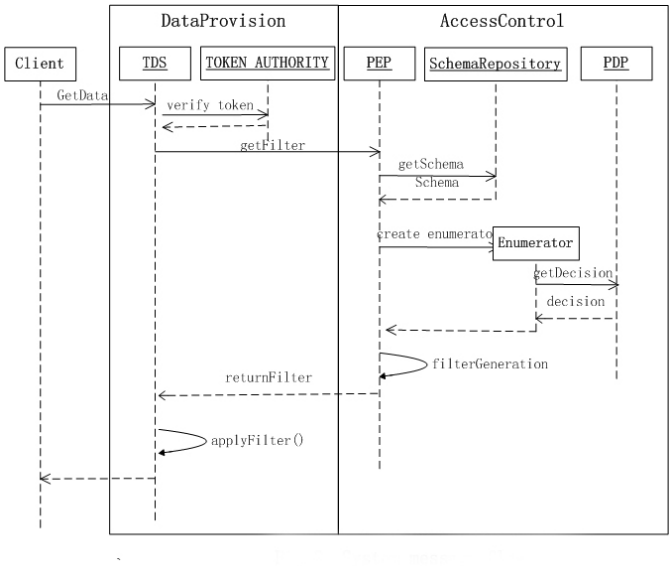

Fig. 2. System message flow

B. Implementation Architecture

As indicated in our Figure 2, we implemented a filter-based Trusted Data Service (TDS) in the NatureServe project [19]. In this illustration of our architecture, a client query with a SAML [13] token will be sent to a TDS with a request to provide data on the location of a population of a given species. After verifying the security token and acquiring an authorized role for the token, which is provided in a verifiable certificate issued by a trusted certificate authority, the TDS will send a message to a possibly distributed PEP service to acquire a security filter based on the data integration schema and the user credentials (See Section 4). The PEP uses a schema-driven algorithm to recursively generate the mediation script using XSLT with the coordination of PDP decisions, and returns the transformation script back to the TDS, which then applies it to return a sanitized version of data back to respective clients.

IV. Filter Generation

We dynamically generate the security filter solely at service invocation time based on the data integration schema (We do however, provide a filter cache for performance enhancements), Filter generation happens at the policy enforcement point. The PEP gets user credentials, usually sets of active roles, and an XML schema from the Trusted Data Service. Then it parses the XML schema to generate a schema tree. We recursively generate XPath expressions based on the schema tree and query the PDP for decisions about whether the client has access rights to the XPath selected resources. We also provide for hashing and caching the XPaths, also enhancing the performance when the schema is not very volatile and is common to the providers, as is usually the case in our implementation. If a "permit" decision is issued, the PEP enforces the decision by copying the content of the respective nodes. If a "deny/obligation" is returned, the PEP generates an XSLT script to provide for enforcement.. An example is shown in Figure 3.

```
<xsl:script implements-prefix="ns-security"
    language="java"src="java:test"/>
<xsl:template match="//Polygon">
    <xsl:text disable-output-escaping="yes">
     &lt;Polygon&gt;
    </xsl:text>
    <xsl:value-ofselect="ns-security:encrypt(text())"/>
    <xsl:textdisable-output-escaping="yes">
     &lt;/Polygon&gt;
    </xsl:text>
</xsl:template>
```

Fig. 3. Encryption obligation enforcement script

As we described, the filter generation is based on the XML schema, but there are some issues we need to consider. Compared to other approaches, our algorithm doesn't need involvement of instance data and solely depends on the schema itself. For non-recursive schemas, the enumeration will stop according to schema complexity, however it may loop infinitely on a recursive schema. One approach to deal with this problem is using relative XPaths instead of absolute paths when a recursive element is met. If we use this approach, we have to deal with XPath equivalence checks and path expression containment checks. On one hand, the algorithm to do generic XPath comparison is exponential [20]. On the other hand, an XPath containment result may allow or deny an access to a concrete data items under different circumstances, and the PDP will generate an intermediate result that will throw exceptions and prohibit sound and complete filter generation. In practice, we fix a given depth for enumeration because our schemas are simple and shallow. We use 10 as our default depth for our trusted data integration service. Another issue is a common problem of role-based access

control in distributed environments. Based on role hierarchy we defined in our model, a user can have network roles and local roles. Policies are submitted by different parties and may be enforced on different local roles. When a user invokes the TDS with security tokens, there are usually authorized roles from a local authentication service in the form of SAML assertion statements. However, policies fetched from a policy store by the PDP may be specified as local roles from different authentication authorities. The role mismatch requires the PDP to get extra information to make the decision, otherwise a 'Not Applicable' ruling will be generated. Dealing with role mismatch is beyond the scope of this paper. Figure 4 shows the filter generation algorithm

Input: Schema (S), Annotation (A)
Output: Filter Script(f)
Method:
1) Generate tree (T) representation of the XML schema S
2) SchemaEnumerator recursively generate set S' of all xpath expressions with depth lower than default depth from T
3) If a xpath expression exp not belongs to A
4) Delete exp from, S'
5) Endof
6) Generating filter heeders
7) Foreach expression exp in S'
8) Asking PDP for access decision, (this step may involve role federation)
9) Generate enforcement script based on the PDP decision
10) Endfor
11) Return generated script f

Fig. 4. Filter Generation Algorithm

V. Policy Conflict

The semantics of policy combination can be considered as arbitrary tree processing. Every tree represents a target, and leaves of the tree are different rules or policies. Intermediate nodes are combination operators such as "FirstPermit" or "DenyOverrides", etc. and can be customized by external functionality. In a distributed environment, policies may be defined by different entities and held at different places. Hence it is not practical to integrate all policies in advance and policy conflict can occur. A policy conflict is an inconsistency that causes indeterminacy in the PDP's query interpretation. We divide such conflicts into resource-centric policy conflicts, operation-centric policy conflicts and resource-semantic conflicts. Examples are below.

A resource centric policy usually regards ambiguous situations defined against a single resource. Subject, action or decision conflicts belong to this category. Predefined XACML combination algorithms for a policy set can be used to solve this kind of conflict effectively, such as the denyOverride given in Example 5.1 below.

Example 5.1: (decision conflict)
Policy 1 { "public user", "//Eos/county", "read", "Permit", "copy"}
Policy 2 { "public user", "//Eos/county", "read", "Deny", "elimination"}
Solution: A "DenyOverrides" combination algorithm defined for the policy set with target "//Ecos/county" – meaning that a "Deny" always supersedes a "Permit" – resolves the ambiguity.

In Example 5.2, a particular user is assigned multiple roles. A XACML combination algorithm extension allows us to assign some priority between subjects by building role hierarchies.

Example 5.2: (subject conflict)
Policy 1 { "researcher", "//Eos/county", "read", "Permit", "copy"}
Policy 2 { "public user", "//Eos/county", "read", "Deny", "elimination"}
Solution: A high priority role "researcher" overrides low priority role "public user", and a "Permit" will be issued. As both examples show, resource centric policy conflict usually applies to the same resource and can be solved totally in the PDP, which is the location of policy combination algorithms, whether those from XACML or those we extend, and which require no external computation (cf. Example 5.4).

An operation-centric policy conflict is generally an obligation conflict. The conflict in Example 5.3 usually happens at policy enforcement time and is resolved by the PEP based on stated resolution rules or results in an exception raised by the PEP.

Example 5.3: ("elimination" obligation conflict)
Policy 1 { "public user", "//Eos/county", "read", "Deny", "elimination"}, "truncate whole tree"
Policy 2 { "public user", "//Eos/county", "read", "Deny", "elimination"}, "truncate a subtree"
Solution: The PEP will generate a filter based on the association between the policy issuer and their supplied data. So in the returned document, if the data is from issuer1, the whole subtree will be truncated, if it is from issuer2, only a subtree will be truncated.

Resource semantic conflict often applies to different resources and it is data dependent. Consider the two policies in Example 5.4, and an implicit knowledge restriction such as: if a user is not allowed to access a bigger resolution, he is not allowed to access a smaller resolution either. With the raw data record in the example we provoke a conflict based on this knowledge illustrated thus:

Example 5.4:
Policy 1 { "usergroup A", "//Eos/county", "read", "Permit", "copy"}
Policy 2 { "usergroup A", "//Eos/polygon", "read", "Deny", "elimination"}
Data:
$< SpeciesDistribution >$
$< Countyid =$ "$county\_x$"$/ >$
$< Polyconid =$ "$Polygon\_y$"$/ >$
$< /SpeciesDistribution >$

If the size of county x is bigger than polygon y, we will simply filter out the polygon, and the user in usergroup A will get a county level Species Distribution to which he is entitled. However, if the size of county x is smaller than polygon y, then

based on the logical restriction, both records will be filtered out and the user is not allowed to access either entry. In these kinds of cases, the PEP can only tell the difference at enforcement time and the corresponding external computation needs to be invoked by conditional code inserted into the filter when it is constructed, solely on the basis of the integration schema and the policies.

VI. Summary and Future work

We have proposed a new paradigm to maintain security constraints during Web Service oriented data integration. A role-based and context-aware XACML module is used to define access control policies general enough for every distributed service provider. A sanitized view is based on security filter generation at run time. This view generation process isolates the original data and schema from the client to prohibit the information leakage. Different enforcements are instantiated by the filter according to decisions from a PDP. Policy conflicts are addressed in our architecture.

We have begun to extend our research to a systematic approach to policy conflicts not addressable by XACML extensions[21]. Issues about policy and operation delegations, bringing ontology to make policy combination more expressive and improve the data integration process by materializing filters in advance, are worthy of more consideration in future work.

References

[1] E. Beritino, S.Castano, E.Ferrai, and M.Mesiti, Controlled Access and Dissemination of XML documents, In Proc. 2nd ACM Workshop on Web Information and Data Management, 1999.

[2] E.Damiani, S.D.C.Vimercati, S.Paraboschi, and P.Samarati, Securing XML Documents, in EDBT, 2000.

[3] E.Damiani, S.D.C.Vimercati, S.Paraboschi, and P.Samarati, A Fine-grained Access Control System for XML Documents, ACM Trans. Information and System Sec., Vol.5, No.2, May 2002.

[4] W Fan, CY Chan, M Garofalakis, Secure XML Querying with Security Views, Proceedings of the 2004 ACM SIGMOD international conference, 2004 Data Management, 1999.

[5] S. Hada and M. Kudo. XML access control language: Provisional authorization for XML documents. http://www.trl.ibm.com/

[6] S. Cho, S. Amer-Yahia, L. Lakshmanan, and D. Srivastava. Optimizing the secure evaluation of twig queries. In VLDB, 2002.

[7] R Bhatti, E Bertino, A Ghafoor, JBD Joshi,XML-based specification for web services document security, Computer, 2004

[8] X Feng, L Guoyuan, H Hao, X Li,Role-based access control system for web services,Computer and Information Technology, 2004

[9] G. Miklau and D. Suciu. Controlling access to published data using cryptography. In VLDB, 2003.

[10] M.Kudo, S.Hada, XML Document Security based on Provisional Authorization, In Proc. of ACM

[11] OASIS eXtensible Access Control Markup Language (XACML) TC, XACML V2.0, http://www.oasisopen.org/ committees /tc_home.php ? wg_abbrev = xacml

[12] W3C, XSL Transformations (XSLT) Version 1.1;W3C Working Draft 24 August 2001, http://www.w3.org/TR/2001/WD-xslt-20010824/, August 2001. Data Management, 1999.

[13] OASIS Security Services (SAML) TC, "SAML V2.0", http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf

[14] OASIS XACML policy schema

[15] http://www.ipg.nasa.gov, visited 2003-09-29

[16] Markus Lorch, David Adams, Dennis Kafura, Madhu Koneni, Anand Rathi, Sumit Shah "The PRIMA System for Privilege Management, Authorization and Enforcement inGrid Environments", communicated to the 4th Ind. Workshop on Grid Computing - Grid 2003

[17] Marlena Erdos and Scott Cantor, "Shibboleth Architecture v5", Internet2/MACE, May 2002

[18] Joon S.Park, R.Sandhu, and Gail-Joon Ahn, Role-Based Access Control on the Web, ACM Trans.Information and System Sec., Vol.4, No.1, Feb. 2002.

[19] NatureServe http://www.natureserve.org/

[20] G Miklau, D Suciu,Containment and Equivalence for an XPath Fragment, PODS, 2002

[21] H Dong, Z. Wang, R. Morris, Rule based Security Policy management for Web Service Integration, submit for publication.

[22] SAML 2.0 profile of XACML v2.0, available at http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf.

[23] LDAP profile for distribution of XACML policies, available at http://lists.oasis-open.org/ archives/xacml/200310/msg00059.html

[24] NatureServe Natural Heritage Program http://www.natureserve.org/ visitLocal/

[25] Web Services Description Language (WSDL) 1.1, available at http://www.w3.org/TR/wsdl

[26] Apache Axis web service framework, http://ws.apache.org/axis/

[27] Spring application framework , http://www.springframework.org/

[28] Core and hierarchical role-based access control (RBAC) profile of XACML v2.0, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf