A Credential-Based Approach for Facilitating Automatic Resource Sharing Among Ad-Hoc Dynamic Coalitions*

Janice Warner[1], Vijayalakshmi Atluri[1], and Ravi Mukkamala[2]

[1] Rutgers University, Newark NJ 07012, USA
{janice, atluri}@cimic.rutgers.edu
[2] Old Dominion University, Norfolk, VA 23529, USA
mukka@cs.odu.edu

Abstract. Today, there is an increasing need for dynamic, efficient and secure sharing of resources among organizations. In a dynamic coalition environment, participants (including users and systems) of an organization may need to gain access quickly to resources of other organizations in an unplanned manner to accomplish the task at hand. Typically, when entities agree to share their information resources, the access control policies are agreed upon at the coalition level. These coalition level agreements are not at the level of fine-grained policies, in the sense that they do not specify which specific users can access which data object. In this paper, we propose a *dynamic coalition-based access control* (DCBAC) model that allows automatic access to resources of one coalition entity by users from another coalition entity. To make the model applicable to true ad-hoc dynamic coalitions, we employ a *coalition service registry*, where coalition entities publicize their coalition level access policies. Any coalition entity wishing to access a specific resource of another coalition entity can obtain a *ticket* by submitting its entity credentials which are subsequently evaluated by the coalition service registry. DCBAC employs a policy mapper layer that computes the exact credentials required by remote users that are comparable to those required by local users. We demonstrate how the coalition and resource level access policies can be specified in XML-based languages and evaluated.

1 Introduction

Today, there is an increasing need for dynamic, efficient and secure sharing of resources among organizations. This is driven by a number of applications including emergency and disaster management, peace keeping, humanitarian operations, or simply virtual enterprises. Typically, resource sharing is done by establishing alliances and collaborations, also known as *coalitions*. Due to the nature of the applications, the coalitions are often *dynamic* where entities may join or leave the coalition in an ad-hoc manner.

In a dynamic coalition environment, participants (e.g. users, systems) of an organization may need to gain access to resources (both data and services) of other organizations to accomplish the task at hand. As an example, in a natural disaster scenario, such as the earth quake in Turkey on May 1, 2003 and the Tsunami in Asia on December 26, 2004, government agencies (e.g., FEMA, local police and fire departments),

* The work of Warner and Atluri is supported in part by the NSF under grant IIS-0306838.

non-government organizations (e.g., Red Cross) and private organizations (e.g., Doctors without Borders, suppliers of emergency provisions) needed to share information about victims, supplies and logistics[10]. Similar examples include homeland security applications where sharing of information across different organizations is needed for identifying criminal and terrorist behavior, illegal shipments, and the like. In a commercial setting, organizations may share resources and information in order to provide comprehensive services drawing from unique skills of diverse participating entities.

Typically, when coalition entities agree to share their information resources, the access control policies are agreed upon at the coalition level. For example, an agreement between government entities A and B might be that they will share resources to aid in a smuggling investigation. These coalition level agreements are not at the level of fine-grained policies, in the sense that they do not specify which subjects are allowed to access which specific resources.

The security policy needed for allowing access would be "user Alice of entity A can access the *immigration* file of entity B." Enforcing the coalition-level security policies requires transforming the high-level policies to implementation level.

Current approaches to facilitate resource sharing resort to one of the following methods: (i) Users from one coalition entity are explicitly given permission to access resources from another coalition entity. This approach is administratively time consuming and requires explicit revocation after the coalition is disbanded or when a user is no longer affiliated with the coalition entity. (ii) A single access id is provided to all of the users of the coalition entity. While this simplifies administrative effort, it makes fine-grained access control impossible. (iii) The resources are copied to the coalition entity that requires access to them. Updates are difficult and may result in uncontrolled sharing. In addition, all the above approaches are not suitable for dynamic and ad-hoc coalitions, and are only feasible among entities that have pre-established partnerships.

Access control research in the area of dynamic coalitions is relatively new. Philips et al. [10] have described the dynamic coalition problem by providing several motivating scenarios in a defense and disaster recovery settings. They have developed a prototype that controls access to APIs and software artifacts [9]. Cohen et al. [3] have proposed a model that captures the entities involved in coalition resource sharing and identifies the interrelationships among them. In [2,5], the researchers have addressed the issue of automating the negotiation of policy between coalition members in a dynamic coalition. Finally, in [13], Yu et.al propose automated mechanisms for trust building between entities using digital credentials Our research complements these works by addressing the issue of automatic translation of coalition level policies to the implementation level policies, and vice versa.

Our approach is a coalition-based access control (CBAC) model that allows automatic access to resources of one coalition entity by users from another coalition entity[1] by employing three layers (coalition, role and user-object). A user's request for a specific remote object is first translated into a role level request, and then into a coalition level request before being sent to the remote coalition entity. At the remote coalition entity, the coalition level request is trickled down through the three layers and translated into an user-object access request. The information appended at each layer at the requesting coalition entity is understood and dealt with by the corresponding layer at the other coalition entity, much like the TCP/IP network protocol.

In this paper, we propose *dynamic coalition-based access control* (DCBAC) model that is specified based on the credentials possessed by coalitions as well as subjects. DCBAC extends CBAC in several directions in order to eliminate its inherent limitations. First, CBAC assumes that every pair of coalition entities have to agree on the coalition policies in advance in order to allow access to data from one entity to subjects[1] of the other entity. As a result, this model cannot entertain a "truly dynamic coalition", where entities of the coalition join or leave in an ad-hoc manner. To cater to true ad-hoc dynamic coalitions, we employ a *coalition service registry* (CSR) where coalition entities publicize their coalition level access policies. Any coalition entity wishing to access a specific resource of another coalition entity can obtain a *ticket* by submitting its entity credentials which are subsequently evaluated by the CSR. Second, CBAC computes the credentials required by a user wishing to access a remote object as a union of all the credentials possessed by all users playing a role, which has privileges to access that object. This is a very conservative approach, and requires a large number of unneeded credentials from a requesting user. Our DCBAC employs a *mapper layer* that accurately computes the credentials required by a user to access a resource. Third, we demonstrate how the access policies, at the coalition level and resource level, can be specified in XML and be evaluated.

Our DCBAC system comprises of four layers – (i) *coalition level*, which interacts with other coalition entities and is responsible for ensuring the authenticity of the coalition entity requesting access to its resources, (ii) *credential filter*, which is responsible for examining incoming credentials and attaching appropriate credentials to outgoing requests, (iii) *credential* $\Longleftrightarrow$ *local access control mapper*, which converts local access control rules to policies concerning credentials for outgoing requests and vice versa for incoming requests, and (iv) *local access control* layer, responsible for uniformly serving the both local and external access requests.

While facilitating automatic access, we ensure that the following requirements are met: (i) The existing access control mechanisms within each coalition entity remain intact. Our approach does not require any changes to the existing local access control mechanisms. (ii) Access is granted to subjects only if they belong to an organization recognized by the coalition, adhering to the coalition level access policies. (iii) Subjects of a coalition entity must have credentials with attribute values comparable to the values of those of the local subjects.

For example, in an emergency management scenario, the International Red Cross decides to make available its Emergency Response Information System to other relief organizations. However, it has three access requirements: (i) individuals who wish to access these resources must belong to an organization recognized by the International Red Cross as a reliable relief organization; (ii) they are allowed to only access information related to the emergency site in which they are currently operating; and (iii) they must possess credentials with attribute values comparable to the values of internal users of the resources. As an example, Dr. Roberts, a member of Doctors Without Borders, wishes to access data on infectious diseases in the area of an earthquake in Turkey,

[1] In this paper, we use subjects and users alternatively.

an emergency scenario that he is currently working on. Clearly, he meets the first two requirements as long as he can present the appropriate organizational credentials and proof that he is operating in Turkey. Whether or not he meets the third requirement, depends upon the credentials determined to be needed and the credentials he presents.

This paper is organized as follows. Section 2 presents required preliminaries to introduce our DCBAC system. Section 3 presents our DCBAC system. Section 4 demonstrates how the coalition and resource level policies can be specified in XACML specification language. Section 5 shows how access requests can be specified and evaluated. Finally, Section 6 summarizes our conclusions and outlines future research in this area.

2 The Preliminaries

We briefly present the necessary formalism required to describe our DCBAC model. Specifically, we review the formalism for resources and credentials.

Each organizational entity maintains a set of resources, RES, that can be shared with other organizational entities within a coalition. Resources may include data objects as well as services offered by the coalition entity. Each resource belongs to a resource-type, organized as a resource-type hierarchy.

Definition 1. [Resource-type] An resource-type rt is a pair (rt_id, RA), where $rt\_id \in RT$ is a unique resource-type identifier; and RA is the set of attributes associated with rt_id. Each $ra_i \in RA$ is denoted by an attribute name.

Definition 2. [Resource] A resource res is a triple $(rt\_id, res\_id, res\_attr\_values)$, where $rt\_id \in RT$, $res\_id \in RES$, $res\_attr\_values = (ra : v_1, \ldots, ra : v_n)$, where $\{ra_1, \ldots, ra_n\} \subseteq RA(rt)$. $RA(rt)$ denotes the set of attributes associated with rt.

We use $res(res\_id), res(res\_id)$ and $res(res\_attr\_values)$ to denote the resource id, the resource-type id, and the set of attribute values of the resource res, respectively. The set of resource attributes describe the resource such as keywords or concepts.

We assume that each subject is associated with one or more *credentials*. Credentials are assigned when a subject is created and are updated according to the profile of the subject. To make the task of credential specifications easier, credentials with similar structures are grouped into *credential-types*. Credential-types are typically organized as *credential-type hierarchy*. We denote the set of credential-type identifiers with CT, the set of credential identifiers with CI, and the set of subject identifiers with U. A credential-type can be formally defined as follows.

Definition 3. [Credential-type] A credential-type ct is a pair (ct_id, A), where $ct\_id \in CT$ is a unique identifier and A is the set of attributes belonging to ct_id. Each $a_i \in A$ has an attribute name and $A(ct)$ is the set of attributes belonging to ct.

Definition 4. [Credential] A credential c, an instance of a credential-type ct, is a 4-tuple $(ct\_id, c\_id, subject\_id, subject\_profile)$, where $ct\_id \in CT, c\_id \in CI, subject\_id \in U$ and $subject\_profile = (a_1 : v_1, \ldots, a_n : v_n)$, where $\{a_1, \ldots a_n\} \subseteq A(ct)$.

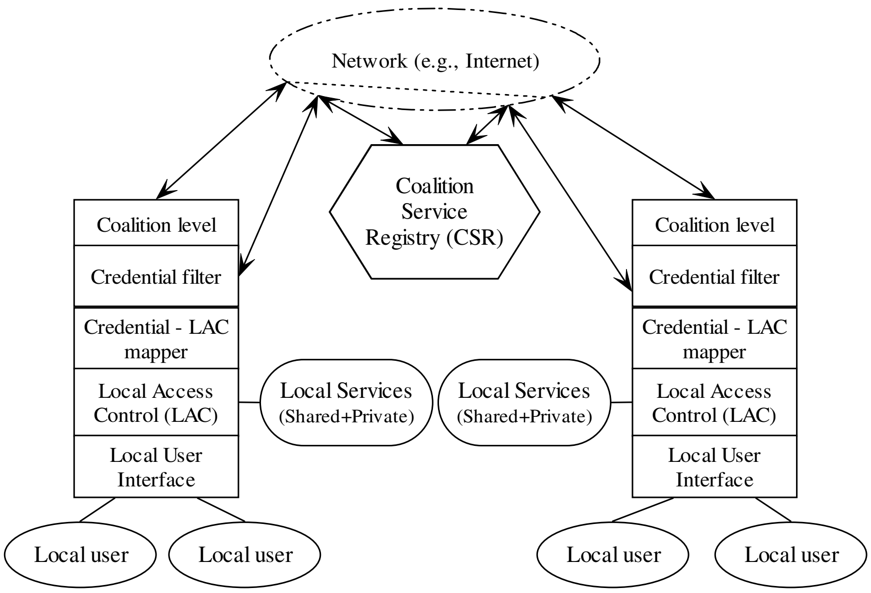

Fig. 1. Proposed Coalition Architecture

The set of credentials associated with subjects in the system is denoted by the credential base (CB). We use $c(c\_id)$, $c(subject\_id)$, $c(ct\_id)$ and $c(subject\_profile)$ to denote the credential id of c, the subject to which c is assigned, the credential type id of c and the set of attribute values of the subject u (the $subject\_profile$) for c, respectively.

Example 1. An example of a credential for credential type "doctor" is as follows: (doctor,045-999, (affiliation: Doctors-without-Borders, Specialty: immunology)). An example of a credential for an organizational level credential type "organization" is as follows: (organization, 943-777, CareNow, (headquarters: New York, NY, tax-status:non-profit)). It has two attributes - headquarters and tax-status.

3 The DCBAC System

Our DCBAC system is comprised of a four layered architecture as depicted in Figure 1. In this section, we briefly describe the functionalities of each of the components.

3.1 Coalition Service Registry

In order to facilitate dynamic and ad-hoc collaboration, we employ a registry service similar to the model adopted for web services through which resources are offered to potential collaborators. Such a model mitigates the need to negotiate and establish collaboration policies among coalition entities. Any entity can set its own sharing policies, describe the types of resources that it is willing to share, and specify the required organizational credentials needed to access these resources. Adopting a model similar to that of Web services is attractive in that it offers a readily available access interface.

We propose the use of a collaborative registry, called the *coalition service registry* (CSR), similar to the UDDI Web Service registry[7]. It is used to define the set of resources that coalition entities will make available and to describe the interfaces and credentials used to access those resources. Our CSR will also verify organization-level credentials and issue a "ticket" which can be submitted by individuals in the organization when submitting an access request for the advertised resources. This "ticket" is nothing but a SAML assertion.

UDDI offers a standard way for potential collaborators to search registries for resources based on a classification scheme or keyword. Queries for these resources can be modulated with acceptable security and transport protocols. These searches are performed against the information provided by the entity detailing to the desired extent who the entity is (using the "BusinessEntity" structure), what resources are being provided (using the "businessService" structure), and the details on how to request access to the resources (using the "binding template"). The binding template would indicate the network address of the *Coalition Access Point* (CAP) for the resource publishing entity. The CAP is able to interpret the requests and make access decisions. Specifics about the access requests would be posted in the binding template as well, including the format of credentials accepted, the format of the request and security requirements (i.e., digital signatures, encryption).

While publishing resources in a public registry would allow the potential collaborators to learn about offered resources, public announcement of shared resources is not desirable. The CSR is expected to be hosted at one or more secure sites for a community of interest. For our example, the Red Cross may state a coalition level policy by registering their service at a registry and make their service only accessible to reliable relief organizations through membership.

To gain access to a desired resource, a user (or an organization on behalf of its user) submits the requested organizational level credentials to the CSR. The registry validates the credentials and issues a SAML assertion. This is the "ticket" that a user from the authenticated coalition entity must present to attempt to access the specific resources being made available. Note that receipt of the ticket is not sufficient for access to the resources. Instead, the assertion merely confirms that the user is from an organization that matches the organizational level policy of the organization offering the resources.

Coalition entities may be permitted to join a coalition for a specific period of time. Of course, revocation may be desirable at any time. Revocation procedures may be performed at each individual CAP by implementing a function that follows decision rules on whether a given assertion is accepted at a given time.

3.2 The DCBAC Layers

Additional specification of the layers are given in the following.

Coalition level. The top layer is the coalition level. It interacts with the coalition level at other coalition entities. For simplicity, in this paper, we are considering only a single coalition (in which an entity participates). When a entity participates in multiple coalitions, there would be multiple interfaces at this level (similar to a virtual machine model that supports multiple virtual machine interfaces to its processes [11]). In that case, we can perceive this layer as having multiple coalition level software all existing simultaneously at an entity's CAP. This layer receives service requests from other coalition entity CAPs. The exact components of this request are presented later in Section 3.3.

On receiving an external service request, the top layer authenticates the requesting coalition entity by validating the "ticket" received with the request. It checks if the coalition policy has changed since the ticket was issued. If so, the request is rejected by this level. The authentication is for the coalition entity that is sending the request rather than for the individual user who may have initiated the request. The ticket is stripped off and the request is then forwarded to the credential filter.

Credential Filter. The credential filter layer is responsible for filtering incoming and outgoing requests and their associated user credentials. When this layer receives a service request from the coalition level layer, it checks whether or not the service is made available to the coalition (i.e., registered in CSR). It then checks whether or not the provided credentials are adequate to execute the request. If they are not adequate, it rejects the request and sends an exception to its coalition level. In addition, depending on its own organizational policies, it may downgrade or upgrade the credentials of a specific entity in the coalition. For example, if a coalition consists of 10 entities, and entity A has less trust on specific credentials offered by entity B, then it could downgrade them.

If one entity, for example, is suspected of revealing private information, another entity could downgrade the credentials from this entity. After such filtering/transformation of credentials, it forwards the request with credentials to its lower layer. It should be noted that the credential layer has access to both the CSR and the local policies (if any) regarding exception polices with regard to coalition entities. Below this layer, there is no distinction between local and non-local user requests.

Let us now consider its handling of requests made by a local user to access non-local services of the coalition. When its lower layer hands it a request and the associated user credentials, it accesses the CSR to check: (i) If the service is registered at the registry; (ii) If the provided credentials are adequate to provide the service. If both checks are positive, it filters out the credentials to be sent so that only the needed credentials ($required\_subject\_credentials$) are provided to the service provider. For example, if the requester is a PhD, an MD, and the director of a research institute, and if the service only requires MD as a credential, the credential filter would filter out the PhD and director credentials. This is in line with the need-to-know principle adopted in operating systems to provide resource protection and privacy [11]. It then sends up the request and filtered credentials to the coalition level.

Credential$\Longleftrightarrow$LAC Mapper. The credential$\Longleftrightarrow$LAC mapper is responsible for mapping the requestor's credentials to the local access control terminology and vice versa. It takes the local access control rules and converts them into a policy based on credential attributes and resource attributes.

When the mapper receives a request with credentials from the credential layer, it looks at its map and determines the possible local access controls that may be attributed to that request. For example, when it receives the credentials of $\langle$Location: Turkey, Specialty: infectious disease, Education: MD$\rangle$, and if the local access control policy is RBAC, then the mapper would search its local map to determine possible roles that may be assigned to the request. The map would specify required credentials derived from those credentials that internal users in a role have minus the set that are either individual attributes (e.g., name, e-mail address) or internal organizational attributes (e.g., department, project). The local map, for example, may associate the received credentials with the local roles of $\langle$Doctor$\rangle$ and $\langle$Intern$\rangle$. If two roles are in a single hierarchy, then the map identifies only the highest role among them. Otherwise, it will form a union of the roles and forwards this to the LAC layer along with the request. In other words, if the hierarchy consists of doctor and intern, only doctor will be forwarded.

For outgoing requests, the mapper receives determines the role of the requestor and computes the union of all credentials ($subject\_credentials$) associated with the associated roles. It forwards the request and credentials to its credential filter layer.

Local Access Control (LAC) Layer. The local access control layer enforces control on local services for both local and non-local requests. As shown in Figure 1, the local requests are received through the Local-user-Interface (LUI). The non-local requests are received through its upper layer. To understand its functionality, let us assume a specific LAC such as RBAC. Since both local and non-local requests are accompanied by the appropriate set of applicable roles, RBAC checks whether or not the service is permitted for any of those roles. Of course, problems may arise when the requested service is permitted for one of the roles and explicitly denied for some other role. Since this is not specific to coalitions, we do not handle this problem here.

If a service request passes the access control, it is forwarded to the local services module which executes the request and returns the result to the LAC. If the request came from the local user, the results are returned through LUI. For non-local requests, the results are forwarded to its upper layer.

3.3 The Request-Response Protocol

In the following, we present the detailed steps of how an access request is processed and give an example using an access request by Dr. Roberts of Doctors Without Borders for an object (RID_730) at the International Red Cross:

1. At the LAC Layer of the Requesting Entity: At the requesting coalition entity, a user requests an resource by specifying the user request for a specific resource type, which is as follows: $\langle request\_id, subject\_id, rt\_id \rangle$.
   For example, ⟨ 744, roberts, Red_Cross_RID_730 ⟩.
2. At the credential$\Longleftrightarrow$LAC Mapper Layer of the Requesting Entity:
   $\langle request\_id, subject\_credentials, rt\_id \rangle$. For example, ⟨ 744, (degree:MD, gender:M, location:Turkey, specialty: infectious disease), Red_Cross_RID_730 ⟩.
3. At the credential filter layer of the Requesting Entity:
   $\langle request\_id, required\_subject\_credentials, rt\_id \rangle$. For example, ⟨ 744, (location:Turkey, specialty: infectious disease), Red_Cross_RID_730 ⟩.
4. At the coalition layer of the requesting entity: $\langle request\_id$, $(requesting) entity\_id$, $(resourceprovider) entity\_id$, $organizational\_credential\_assertion$, rt_id, $required\_subject\_credentials \rangle$. For example, ⟨ 744,Doctors Without Borders, Red Cross, SAML Assertion, Red_Cross_RID_730, (location:Turkey, specialty: infectious disease) ⟩.

The message is sent to the service provider coalition entity.

1. At the Coalition Level Layer of the source entity, it validates the $organizational\_credential\_assertion$ and sends the following to the credential filter layer. $\langle request\_id, required\_subject\_credentials, rt\_id \rangle$. For example, $\langle$ 744, (location:Turkey, specialty: infectious disease), RID_730 $\rangle$.
2. At the credential filter layer of the requesting entity, it verifies that the $required\_subject\_credentials$ are included in the request. It then passes $\langle request\_id, required\_subject\_credentials, rt\_id \rangle$ to the mapper layer. For example, $\langle$ 744, (location:Turkey, specialty: infectious disease), RID_730 $\rangle$.
3. At the credential$\Longleftrightarrow$LAC mapper layer of the source entity, it compares the credentials of remote subjects to the determined equivalent of the local access control. If they are acceptable, the mapper layer requests access from the LAC layer $\langle rt\_id \rangle$. For example, $\langle$ 744, RID_730 $\rangle$.
4. At the LAC layer of the source entity: This layer serves the access request. It identifies the requested resource or set of resources and makes them available to the requesting coalition entity.

4 Policy Specification

In this section, we present our approach to specifying coalition based access control policies. These policies are specified at two levels – coalition level and resource level. The coalition level policies state the high level access control rules of coalition entities on sharing resources among themselves that are publicized in the CSR.

The resource level policies state the access control rules on accessing a specific resource by a user belonging to a coalition entity. These policies are stored and maintained at the resource owners. We use XACML [6] policy language to specify both the coalition level and resource level policies. Finally, as noted in Section 3 we use SAML for specifying organizational tickets.

We have chosen to use XML-based specifications for realizing our model. This is because, using XML-standards, specifically UDDI, SAML and XACML provides many benefits. Standards exist through the OASIS organization, the protocols are being implemented, and parsers exist that can be readily used. The standards are extensible, allowing the addition of functions and labels as needed. In addition, since collaboration clearly involves multiple distributed parties, use of namespaces and semantics that have already been defined for credentials and resources can speed implementation due to re-use and can facilitate dynamic interoperability though common definitions.

4.1 Coalition Level Policies

The coalition level access policy specification can be compactly represented as follows: $\langle entity\_id, credential\_set, rt\_id \rangle$. The e policy states that only organizations that possess the credentials specified in the $credenial\_set$ are allowed to access the resources belonging to the rt_id owned by the coalition entity with the specified $entity\_id$. These organizational level policies are specified at the CSR. This coalition level policy represents only the organizational level credentials that must be provided by an individual who requires access.

```
01 XACML HEADER
02 <Policy>
03  <PolicyId=1>
04  <RuleCombiningAlgId= "deny-overrides">
05 <Description>
06  read access to emergency archives with non-profit
    relief organizations who are listed as ReliefWeb members
07 </Description>
09  <Subjects>
10      <Subject>
11         <SubjectMatch MatchId="name-match">
12               <Attribute Value DataType=string "www.reliefweb.org/~orglist">
13            <SubjectAttributeDesignator
14               AttributeId=organization_id
15                     DataType=string/>
16         </SubjectMatch>
17         <Subject Match matchId="string-match">
18               <Attribute Value DataType=string "non-profit">
19               <SubjectAttributeDesignator
20                     AttributeId=tax-status
21                     DataType=string/>
22        </SubjectMatch>
23     </Subject>
24  </Subjects>
25   <Resources>
26      <Resource>
27               <Attribute AttributeId=resource_type>
28               <AttributeValue>
29               emergency-archive
30               </AttributeValue>
31      </Resource>
32    </Resources/>
33 </Policy>
```

Fig. 2. An Example of the Coalition Level Policy

XACML [6] provides a way to describe the above policy. We have adopted XACML to specify this policy because of the following capabilities critical to our approach: (i) It provides a method for basing an authorization decision on attributes of the subject (e.g., subject credentials) and resource (e.g., resource type).(ii) It provides a method for combining individual rules and policies into a single policy set. This helps in combining rules applicable to one coalition into a policy set. (iii) XACML was written explicitly to provide a common way to express policies and ensure enforcement in a distributed environment, making it appropriate for a coalition based environment. (iv) Although not explicitly used in our example, the logical and mathematical operators on attributes of the subject, resource and environment will aid in flexible policy descriptions.

Example 2. Returning to our example, assume that the Coalition Level Policy for the Red Cross is as follows: "Allow read access on emergency archives to non-profit relief organizations who are listed as ReliefWeb members". This policy can be stated as $\langle$*RedCross,* $\{$*ReliefWeb-member, non-profit*$\}$*, emergency-archives*$\rangle$.

The example coalition level policy can be expressed in XACML-like specification, as shown in Figure 2, in which lines 1-7 introduce the policy and 8-23 indicate the subject attributes that must be matched. There are two attributes to be matched. The first, as specified in lines 11 - 16 is a name-match of the organization to names listed at "www.reliefweb.org/∼orglist". The second, as specified in lines 17 - 22, is a string-match for tax-status which must equal non-profit. Lines 25-32 provide the resource attributes for the resources that are made available.

4.2 Resource Level Policies

The resource level access policy specification can be specified as: $\langle credential\_set, rt\_id\rangle$. This policy states that only the subjects who possess the credentials specified by the $credential\_set$ are authorized to access the resources of type specified in the rt_id. These policies are specified within the coalition entity organization, and are maintained in the local policy base. These are nothing but the local policies translated into this form by the credential$\Longleftrightarrow$LAC Mapper.

Example 3. Let us consider the following two resource level policies: (1) External individuals may only access information related to the emergency site in which they are currently operating. (2) Individuals must have credentials with attribute values comparable to the values of internal users of the resources.

The first policy serves as a filter, specifying that there must be a match between the subject's location and the location for which the resource is concerned. Specification of this policy using XACML requires that a variable be defined based on the credential attribute "location". This variable is then used to match resource attribute values. To specify the second policy in XACML is more straightforward. The required credentials generated by the mapper are specified in XACML as attributes of the subject to match.

Assume that Dr. Roberts, a member of Doctors Without Borders, wishes to access data on infectious diseases in the area of an earthquake in Turkey, an emergency on which he is currently working. The requested resources are distributed in two parts of our resource hierarchy, RID 517 and 730. To access these resources, the resource policy specifies that the subject must be a doctor with the specialty of infectious diseases to access RID 517, but there are no specialty restrictions for RID 730. Internally, the access control policy is that the subject must also be assigned to at least one of a set of specific projects. However, since this is an internal attribute, it would not be included in the attribute requirements for external users. This policy can be specified in XACML as shown in Figure 3, where it consists of a target (lines 2 - 9) and a rule for each resource type. The target is used to (1) check that the user has a location credential and (2) to store the value of the location attribute in a variable "LOC". The rules for each resource type match (1) the variable "location" to the resource attribute "theater-of-operation" (lines 21 to 24 for the first policy, and 41 to 44 for the second policy, (2) match the subject attributes presented by Dr. Roberts to the requirements for the requested resource. In the first policy, the credentials needed are specified in lines 13 to 15. In the second policy, there are no subject credentials matches to be found.

Note that there would be policies associated with all other shared resources, which are not illustrated here. Note also that the policy is specified on the highest resource of the hierarchy. For example, a policy is specified over resource 510 rather than resource 517 because the same policy applies to all of the children of 510.

```
XACML HEADER
<Target>
 <VariableDefinition VariableId="LOC">
         <Apply Function-Id="string-equal">
     <SubjectAttributeDesignator AttributeId="location"
     DataType=string/>
          </Apply>
 </VariableDefinition>
</Target>
<Rule RuleId=1 Effect=permit>
   <Description>
 Read  access is provided to users who present credentials showing a
 specialty of "infectious disease" and location credentials that
 match the "theater-of-operations" attribute of the requested
 resource.
 </Description>
   <Subjects>
     <Subject>
         <SubjectMatch>
                   <AttributeValue ="infectious disease"
                   Datatype = specialty>
     </Subject>
   </Subjects>
   <Resources>
     <Resource>
         RID 510
         <ResourceMatch>
                   <AttributeValue = LOC
             Datatype = theater-of-operation>
         </ResourceMatch>
     </Resource>
     </Resources/>
 <Actions>
     <ActionMatch matchId=string-equal>
         <AttributeValue DataType=string> read
     </ActionMatch>
</Rule>
<Rule RuleId=2 Effect=permit>
 <Description>
 Read access is provided to users who present location credentials that
 match the "theater-of-operations" attribute of the requested resource.
   </Description>
   <Subjects>
          <Subject>
         <AnySubject>
          </Subject>
    </Subjects>
   <Resources>
         <Resource>
               RID 730
               <ResourceMatch>
                    <AttributeValue = LOC
                    Datatype = theater-of-operation>
               </ResourceMatch>
          </Resource>
   </Resources/>
   <Actions>
     <ActionMatch matchId=string-equal>
          <AttributeValue DataType=string> read
     </ActionMatch>
   </Actions>
 </Rule>
```

Fig. 3. An Example of the Resource Level Policy

```
<saml:Assertion
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"//
    MajorVersion="1" MinorVersion="1"
    AssertionID="buGxcG4gILg5NlocyLccDz6iXrUa"
    Issuer="Coalition-service-registry095
    IssueInstant="2005-02-28T12:24:37">
    Recipient="Red Cross"
        <saml:Conditions NotBefore="2005-03-01T01:00:00"
            NotOnOrAfter="2006-06-15T01:00:00"/>
        <saml:AuthenticationStatement
            <saml:Subject>
                Doctors-without-Borders
            </saml:Subject>
        </saml:AuthenticationStatement
</saml:Assertion>
```

Fig. 4. SAML Assertion

5 Access Request Evaluation

In this section, we describe how an access request is evaluated. In order to send an access request for a resource, the coalition entity of the requesting subject should first have the required credentials to access the resource. This verification is done at the CSR and a ticket to access the resource is issued to the requesting entity. In Section 5.1, we describe how such assertions are generated and their format. A user of a requesting entity then sends an access request to a coalition entity by appending this ticket. In Section 5.2, we discuss how this access request is specified in XML specification language and how it is evaluated.

5.1 SAML Assertion

In order to gain access to a CAP, a user must submit the SAML assertion generated by the CSR. The SAML assertion consists of a header, an assertion id, the Issuer element containing the identifier for the specific CSR, the issue instant element providing the date and time when issued, the recipient for the assertion, conditions for when the assertion can be applied and the assertion itself which consists of the subject, indicating the organization identity for which the assertion applies and any optional attributes.

In our example, Dr. Roberts' organization, Doctors without Borders, would have obtained a SAML assertion from the CSR, specified as shown in Figure 4.

5.2 Access Request Specification and Evaluation

XACML defines several functional components that work together to perform access control. We make use of the following components at the CAP. The functionalities supported by our CAP include: (i) a XACML Policy Enforcement Point (PEP), which assesses external resource requests and enforces authorization decisions (ii) a Policy Decision Point (PDP), which evaluates applicable policy and makes authorization decisions, and (iii) a Policy Information Point (PIP), which provides attribute values for outgoing requests as well as environmental attribute values such as time and date as necessary. Not included in our CAP is the Policy Administration Point functionality (PAP) which is the system used to create policies. A common policy system for all

```
<Request>
     <Subject>
         <SubjectAttribute>
             specialty = cr:doctor.specialty
         </SubjectAttribute>
         <SubjectAttribute>
             location = cr:work.site
         </SubjectAttribute>
     </Subject>
     <Resource>
         <ResourceAttribute>
             RID = 517 AND RID = 730
         </ResourceAttribute>
     </Resource>
     <Action>
         Read
     </Action>
</Request>
```

Fig. 5. Access Request

policies (local and external) is important to ensure that policies are not conflicting and carefully administered. This is out of scope of this paper.

Looking at our example again from end-to-end, Dr. Roberts would first search the CSR and find information that could be useful to him relevant to his relief work in Turkey is available from the Red Cross CAP. Triggered by his request, his organization submits its organizational level credentials to the registry, which verifies the credentials and then returns a signed SAML assertion to be used at the Red Cross CAP. The request can be expressed as: Request = $\langle$SAML Assertion, Credential-set, Resource, Action$\rangle$. It consists of the SAML Assertion from the CSR, a credential set associated with Dr. Roberts that meets the requirements of the access control rules for the requested resources, the resources and finally the action requested. The last two items as well as a reference to the portions of the submitted credentials that are applicable will be referred to as the "Req". The "Req" can be expressed in XACML with the $\langle$SubjectAttribute$\rangle$ tags referring to the portions of the submitted credentials that apply.

For example, Dr. Robert's request would consist of
$\langle$SAML-Assertion-for-Doctors-without-Borders, doctor-credential, work-credential, Req$\rangle$ where Req would be formatted in XACML policy language as shown in Figure 5. The Red Cross CAP would match the request with the associated policy for the requested resources and validate the credentials if necessary. If there is a match, access is allowed. If not, access is denied.

6 Conclusions and Future Work

In this paper, we have presented a coalition-based access control system designed to automatically translate coalition level policies into subject-resource level policies by employing an attribute-based approach. It considers the attributes associated with user credentials and those associated with resources, making the formation of specific groups of subjects and resources unnecessary. Our system extends the original model proposed [1] in the following three directions: First, our system is capable of catering to "true" ad-hoc dynamic coalition by facilitating a coalition service registry where resources to be shared among potential coalition entities are advertised. Organizations can obtain tickets to access the resources if they satisfy the required organizational-level credentials. Second, our system is capable of assessing the exact credentials necessary by a remote user and be able to map the access request as if it was an access request from a local user. Finally, we have demonstrated how our CBAC system can be implemented using OASIS XML-based standards, including XACML, UDDI, and SAML.

Our DCBAC model assumes that the local policies are mapped to credential-based to facilitate external users to access the local resources. However, we have not addressed how this mapping can be accomplished. Our future work includes mapping of different types of local policies, including DAC, MAC, RBAC, etc., into credential based policies. Moreover, CSR and ticket issuing function could become a performance bottleneck. We are exploring the issue of distributing/replicating the CSR so that some of the functionalities of the CSR can be accomplished by the coalition entities themselves. The challenge is to accomplish this without having to have a trusted coalition entity. We are also exploring the case where resources are not owned by only one entity in the coalition, but instead are shared by several entities. We intend to extend our approach to facilitate such cooperative environments similar to the work on cooperative role-based administration in [12]. Finally, we believe that delegation is an important feature, which must be supported in coalition-based systems [4] and we intend to include this support as well.

References

1. V. Atluri and J. Warner. Automatic enforcement of access control policies among dynamic coalitions. In *International Conference on Distributed Computing and Internet Technology (ICDCIT) 2004*, December 2004.
2. V. Bharadwaj and J. Baras. A framework for automated negotiation of access control policies. *Proceedings of DISCEX III*, 2003.
3. E. Cohen, W. Winsborough, R. Thomas, and D. Shands. Models for coalition-based access control (cbac). *SACMAT*, 2002.
4. P. Freudenthal, K. Pesin, Keenan Port, and Karamcheti. drbac: Distributed role-based access control for dynamic coalition environments. *ICDCS*, July 2002.
5. H. Khurana, S. Gavrila, R. Bobba, R. Koleva, A. Sonalker, E. Dinu, V. Gligor, and J. Baras. Integrated security services for dynamic coalitions. *Proc. of the DISCEX III*, 2003.
6. OASIS. extensible access control markup language (XACML), version 2. *OASIS Standard*, February 2005.
7. OASIS. Universal description discovery and integration (UDDI), version 3.0.2. *OASIS Standard*, February 2005.
8. OASIS. Assertions and protocols for the oasis security assertion markup language (saml), version 2. *OASIS Standard*, January 2005.
9. C. Philips, E. Charles, T. Ting, and S. Demurjian. Towards information assurance in dynamic coalitions. *IEEE IAW, USMA*, February 2002.
10. C. Philips, T.C. Ting, , and S. Demurjian. Information sharing and security in dynamic coalitions. *SACMAT*, 2002.
11. A. Silberschatz, P. Galvin, and G. Gagne. *Operating System Concepts with Java*. John Wiley and Sons, 6 edition, 2004.
12. H. F. Wedde and M. Lischka. Cooperative role-based administration. *SACMAT*, 2003.
13. T. Yu, M. Winslett, and K.E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security*, 6(1):1–42, February 2003.