Ontology based policy interoperability in geo-spatial domain

Mohammad Farhan Husain*, Tahseen Al-Khateeb, Mohmmad Alam, Latifur Khan

Dept. of Computer Science & Engineering, University of Texas at Dallas, Richardson TX 75080, USA

ABSTRACT

With advances of digital technology two or more organizations form a federation for a temporary purpose. It is possible that an employee of an organization needs to access resource of another organization. While languages like XACML are very good for access control of a single organization, they are not able to address interoperability issues of such federations. Lack of interoperability of access control policies does not allow someone to access remote resources. In this paper we propose an approach by augmenting with knowledge-base, Ontology that will facilitate access control policies to provide resources of remote organizations. In addition, we introduce a new effect of the rule, partial permit in our application domain (Geo-Spatial).

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

In today’s world, organizations, small or large, are ubiquitous and in possession of a large amount of resources. These resources can contain sensitive and public information along with other information that lie somewhere in between. These resources need to be accessed and managed by a dedicated set of persons or groups of persons. In order for these groups or persons to be able to access, utilize, and manage these resources, a set of sound access control policies need to be put in place. Policy specification languages allow organizations to set up specific access control policies for specific requirements, and there exist several of them.

Oftentimes, two or more originations form a federation, either temporarily or permanently, for one or more reasons. One needs to address a policy definition language for these purposes. XACML is one of the prominent languages for defining access control policies since it allows defining fine grained policies. It provides good mechanisms to define and use policies for a single organization. However, it fails to address the interoperability issues of federations comprised of more than one organization described above. Lack of interoperability of access control policies prevents someone from accessing resources other than his current organization.

Ontology is a collection of concepts and their inter-relationships that collectively provide an abstract view of an application domain [7,8]. There are two types of ontologies such as generic ontology (i.e., WodNet [9], Sensus [10]) and domain dependent ontology. In this paper, we exploit domain dependent ontology.

An organization’s policy will consist of a set of rules. Each rule will be stated in terms of subject, objects and actions. The same rule across multiple organizations may be stated using different terms. Hence, heterogeneity may exist. We would like to address these heterogeneities using domain dependent ontologies.

We extend this research in the geo-spatial domain. Geo-spatial domain poses further challenges. Besides the traditional permit and deny effect we introduce a new effect partial permit. This partial permit will be valid for this domain, For example, a user may want to view a map of an area which has some military bases in it. The user may not have sufficient access privilege to know the location or even the existence of those bases. But he may be permitted to view the schools, libraries, airports etc. of that area. In such a scenario, he may be granted partial permit rather than denying the whole map. He will be provided a map with partial data containing the objects that he is allowed to view. Restricted objects like military bases will not be shown in the map.

The novelty of this work is as follows. First, we address policy interoperability issues across multiple organizations using ontologies. Second, to the best of our knowledge this is the first attempt to address policy interoperability issues in geo-spatial domain. Finally, we introduce a new effect of the rule, partial permit in this domain.

The paper is organized in following directions. In Section 2, we present related work. In Section 3, we present some background work. In Section 4, we present our approaches and implementation. In Section 6, we present conclusion and future work.

2. Related work

A significant number of works related to XACML has been done for the usage in distributed scenarios. A comprehensive list of references related to XACML can be found in [5].
For addressing security issues of multiple enterprises, Mazzoleni et al. [1] discuss an algorithm to integrate policies from different organizations. When two or more organizations want to collaborate and have their policies defined in XACML, their algorithm provides a single decision for any request taking into consideration policies of all the organizations involved. Policies are integrated two at a time in their approach. However, unlike us, the algorithm does not take into consideration the heterogeneity of different policies of different organizations. It also does not have any provision for partial permit.

Similar work has been done in web service domain. Web Service Policy Constraints Language, WSPL [2–4] is based on XACML and manages authorization in that domain. Along with solutions to specify, verify and apply constraints, it provides a way to intersect two policies. This work is a bit similar to that of Mazzoleni et al. [1]. It only works when policies contain “Permit” rules only and their combination algorithm is “Permit Override”. However, it also lacks solution for heterogeneity policies. In addition, it does not have partial permit provision.

Huang [6] proposes a framework to work with policies of multiple business organizations. It uses an ontology repository to address heterogeneity. The repository would hold two types of ontologies: business ontology and security ontology. But the framework does not propose any algorithm to use the repository.

In Atluri and Chun [13], they proposed on implementing security and privacy policies on (high-resolution) Satellite images. They have pointed out that these high-resolution images along with their locations on maps can pose potential threats to national security. They proposed an access control model Geospatial authorization model (GSAM) that allows one to specify complex authorization policies (e.g. resolution-based authorization), But they don’t address the interoperability issues between multiple organizations.

In Damiani et al. [14], authors proposed an extension of RBAC, GEO-RBAC, which deals with location based information. Their model deals with both real and logical positions. They fail to address the interoperability issues as well as the partial permit.

3. Background

Before proceeding, a brief introduction to XACML is needed. XACML is a policy language based on XML which allows resource administrators define their access control requirements for data and application resources [11]. It is based on XML and proposed by OASIS. Data types, functions, and combining logic are included in the language schema. The combining logic allows complex or simple rules to be defined [11]. A very important part of XACML is the part of the language which deals with decisions. It is used to represent the runtime request for a resource. When a request is made, a policy is looked for which protects the resource which request is trying to access. Functions, defined in the language, compare attributes in the request against attributes in the policy rules and then provide permit or deny decision.

XACML is not only a policy language. It also defines a process to evaluate the requests. It defines entities like Policy Enforcement Point (PEP) and Policy Decision Point (PDP) (see Fig. 1). PEP is the entity charged with enforcing the policy. And the decision making is done by PDP which has the policies in its store. The following is architecture of the process.

3.1. Defining policies

An administrator uses XACML to create policies. The top-level element of a policy document is the PolicySet which may contain other PolicySet elements or Policy elements. The Policy element may have Target, Rule and Obligation elements and is evaluated at the Policy Decision Point to produce an access decision.

Multiple policies may be found to be applicable to an access decision. Likewise, multiple rules of a single policy may be applicable to a request. In such scenarios, there will be multiple outcomes which may be conflicting e.g. some of them are “permit”, some are “deny”. Combining Algorithms come to rescue in these situations. They are used to reconcile these multiple outcomes into a single decision. Standard Combining Algorithms defined in XACML are Deny-Overrides, Permit-Overrides, First Applicable, and Only-One Applicable Algorithms.

To determine which resource is protected by a policy, the Target element is used. It contains conditions that the requesting Subject, Resource, or Action must meet for a Policy Set, Policy, or Rule to be applicable to the resource [11]. A very important use of the Target element is building index for efficient and fast lookup of Policies.

Rules are the smallest elements which are used to determine an outcome. It is the actual workhorse of XACML. They provide the conditions to be met to have the outcome of the policy. A policy may contain any number of Rule elements each of which will produce a true or false outcome. A single decision can be derived out of a policy by combining these outcomes. The possible decisions defined by XACML are “Permit”, “Deny”, “Indeterminate”, or a “NotApplicable” decision.

An example policy set is shown in Fig. 2 which illustrates how a policy set is defined.

In this example, a policy set has one policy. The policy has one rule. The rule combination algorithm is permit-override which means if one applicable rule has the effect of permit the overall decision will be permit. The rule specifies that the high access subjects i.e. subjects with high access permissions will not be able to access the Plano city map between 10 am to 2 pm. The target of the rule specifies the two users (Alam and Dr. Latifur Khan) who are regarded as high access subjects. The condition element of the rule takes care of the time. It uses an And function and two relational operators. Current time is taken from the system to compare with the time specified in the rule. If the current time falls in the specified range the effect will be deny.

3.2. Policy attributes

Attributes are place holders for typed values. They are used both in requests and policies. For example, in a request, attributes may include date and time, requestor identification. In Fig. 3, “MathId” and “DataType” are attributes.

The element which is used to retrieve attribute values from a request by specifying the name, type, and issuer of attributes is called
Attribute Designator. There are several attribute designators like SubjectAttributeDesignator, ResourceAttributeDesignator, ActionAttributeDesignator, and EnvironmentAttributeDesignator which retrieves attributes from the respective elements in the request. Request attributes are looked up using XPath queries by an AttributeSelector element.
4. Problem definition

A motivating example will help understand the problem we are trying to solve. Suppose there are two organizations: OrgA and OrgB, both having geo-spatial data. Their access control policy in XACML. For some temporary purpose, they form a federation and want that their subjects (e.g. People, client s/w etc.) of one organization will be able to access resources (e.g. Data, file etc.) of other organization.

Both organizations have policies based on their own naming convention, data type which are not recognized by other organization. Similarly, access requests will contain organization specific keywords and data types. With existing XACML processing model, these requests, sent across organizations, will fail always even though there must be some cases where they could be allowed. The root cause of the problem is heterogeneity. Two types of heterogeneity contribute in this scenario:

1. Naming heterogeneity and
2. Data type heterogeneity.

Naming heterogeneity arises because subjects, resources and attributes are differently defined in different organizations. For example, a Network Administrator can be defined as System Admin in another organization, the action Read can be defined as View and the resource Directory can be defined as Folder. In such cases, policy of one organization is not applicable to another when they form a federation.

The following is an example of parts of two XACML policies defined in two different organizations (see Fig. 3) but essentially saying the same thing that, a certain group of users having high authoritative power (HighAccessSubjects and PowerUsers) can read (View/Read) the description of airport data (VECTOR,VECTOR,FEATURE_AIRPORTS_text and VECTOR,VECTOR,FEATURE_AIRPORTS_description).

4.1. Proposed solution

One brute force solution is to this problem is to rewrite the policies entirely. But that takes a lot of effort, time and man-hours. And also after the federation is broken, policies need to be restored in their previous states. Our solution requires the minimum effort of building ontology and then using the existing policies as they are. Our algorithm will determine whether a request to access some resource should be permitted or not dynamically. Moreover, we are focusing on geo-spatial data.

4.1.1. Approach

Suppose there are two organizations: OrgA and OrgB, both having geo-spatial data. They have access control policy defined in XACML. They form a federation and want that subjects (e.g. People, client s/w etc.) of one organization will be able to access resources (e.g. Data, file etc.) based on their existing policies without any modification and human assistance. As we are focusing on geo-spatial domain, we introduce a new decision type: Partial Permit.

The new rule effect Partial Permit is particularly suitable to geo-spatial domain to grant request partially. Examples will clarify the understanding. Suppose, a request comes to the view map of an airport, but the subject does not have enough privilege to view the whole airport. In GML, the map of airport contains outer and inner boundaries. We can permit the request by granting only the outer boundary of the airport. This scenario is with vector data. Another one involving image airport. In GML, the map of airport contains outer and inner boundaries.

We can permit the request by granting only the outer boundary of the airport. This scenario is with vector data. Another one involving image airport. In GML, the map of airport contains outer and inner boundaries.

Suppose a subject of OrgA sends request "requestA" to OrgB having policy set "policy". The following algorithm will provide a decision:

GetDecision(requestA, policyB)
1. ruleB ← FindRule(requestA, policyB)
2. score ← CalculateScore(requestA, ruleB)

3. If score ≥ γ1 then
4. return “Permit”
5. Else If score ≥ γ2 then
6. return “Partial Permit”
7. Else
8. return “Deny”

FindRule(requestA, policyB)
1. RuleSet ← policyB
2. RuleSet ← { R = RuleSet | min SemanticDistance(SubjectOf(requestA), SubjectOf(R), SubjectOntology) }
3. If Cardinality(RuleSet) > 1 then
4. RuleSet ← { R = RuleSet | min SemanticDistance(ResourceOf(requestA), ResourceOf(R), ResourceOntology) }
5. If Cardinality(RuleSet) > 1 then
6. RuleSet ← { R = RuleSet | min SemanticDistance(ActionOf(requestA), ActionOf(R), ActionOntology) }
7. return R = RuleSet

The algorithm GetDecision works as follows:

Step 1. Within all the policies and rules of OrgB, find the rule which has a subject of minimum semantic distance from the subject of the request in the ontology of subjects (see Fig. 4). In case of ties, find the rule among the tied rules which has a resource of minimum semantic distance from the resource of the request in the ontology of resources. In case of ties, find the rule among the tied rules which has an action of minimum semantic distance from the action of the request in the ontology of actions.

Step 2. Use a function “CalculateScore” to get a match score.

Step 3. The score is compared with a certain threshold called full-effect threshold γ1.

Step 4. If it is above it, return its effect.

Step 5. The score is compared with another threshold called partial-effect threshold γ2.

Step 6. If it is above it, return partial-permit.

Step 7. The algorithm comes here if the score is below the partial-permit threshold.

Step 8. In such case deny is returned.

In case of multiple rules having tie, we use rule combination algorithm specified in the policy to break the tie.

At the heart of the algorithm lie the semantic distance score function “CalculateScore” and two thresholds: full-effect threshold γ1 and partial-effect threshold γ2. The semantic distance score formula is described below.

To find the matching similarity score between two nodes C1 and C2, we first determine their closest common parent C. Then the score S(C1, C2) is formulated as follows:

\[ S(C_1, C_2) = -\log \frac{\text{len}(C_1, C)}{2D} - \log \frac{\text{len}(C, C_2)}{2D} \]

Where \( \text{len} \) is a length operator that calculates the shortest distance between two nodes in an ontology tree and \( D \) is the overall depth of the ontology tree.

We calculate three different score values, \( S_s(C_1, C_2), S_r(C_1, C_2), \) and \( S_a(C_1, C_2) \) for subject, resource and action parameters, respectively. The score values are combined by an aggregation function \( \Theta: E \rightarrow \Psi \) where \( E \) is a set of 3-ary tuples is and \( \Psi \) is the set of real numbers. The function, henceforth referred to as Aggregation function, is represented as

\[ \Theta(S_s, S_r, S_a) = S_s + S_r + S_a \]

The Aggregation function result is compared against two pre-determined threshold values to resolve the policy decision. These values, full-effect threshold and partial-effect threshold, have to be
determined by the system administrator. The decision could be either one of the three effects: Permit, Deny, and Partial-Permit.

5. A case study and implementation

We worked with geo-spatial data in GML format. The data was a map of DFW, which was taken from [12]. It had city and county boundaries and locations of all airports of DFW. We created a small and simple domain ontology for our purpose. We defined a policy containing some rules by using some of the terminologies of the ontology. XACML requests were sent to PEP. The requests had different kind of subject, action and resource. For simplicity we only wrote one program which contained everything: the client generating the requests, the PEP, the PDP. The program was written in Java.

In our experiments, a request was sent to view airport data of Dallas area in one scenario. The subject was of HighAccess group and the decision was full permit. Fig. 5 was the map returned by the program. It shows all the airports in Dallas area.

In another, a request was sent to view the same data but from a different subject. The result was partial permit. The requester could view 7 out of 8 airports. The results can be seen in Fig. 6. The airport circled red in Fig. 5 is missing in Fig. 6 because of partial permit. The partial permit decision blocked the circled airport from being shown to the user.
We used the sensitivity value to implement partial permit. For instance, consider the Airport feature in the dataset. Each property of Airport is tagged with their sensitivity value in the database. A client request resolves into a Partial-Permit, which leads to our algorithm checking for the sensitivity value of the Airport properties. The polygon that corresponds to the outer boundary of the Airport has a sensitivity value of 0 and therefore amenable to be included in the client response. However, the runway lines of the Airport have high sensitivity values and cannot be included in case of a partial permit decision.

6. Conclusion and future works

As more and more organizations are becoming computerized, access control is increasingly becoming a vital issue of security for them. Forming federation for temporary collaboration is also becoming more common than before. Our solution facilitates access control in federations with minimal effort. All that has to be done is to create ontology and use our algorithm as a middleware agent.

In the future, we want to extend our algorithm to take into account all the policies of involved organizations. We will look for what might be the decision for the request if it were sent to the subject’s own organization and take that into consideration. In implementation, we will also experiment our algorithm with complex policies.

References