2009 33rd Annual IEEE International Computer Software and Applications Conference

An Architecture for Verification of Access Control Policies with Multi Agent System Ontologies

Fatih Tekbacak, Tugkan Tuglular
Department of Computer Engineering
Izmir Institute of Technology
Urla, Izmir, Turkey 35430
{fatihtekbacak, tugkantuglular}@iyte.edu.tr

Oguz Dikenelli
Department of Computer Engineering
Ege University
Bornova, Izmir, Turkey 35100
oguzdikenelli@ege.edu.tr

Abstract

Multi-agent systems (MAS) which communicate with intra-domain and inter-domain agent platforms have access control requirements. Instead of a central mechanism, a fine-graned access control mechanism could have been applied to MAS platforms. This paper emphasizes MAS-based domain and security ontologies with XACML-based access control approach for MAS platforms. The domain dependent behaviour and access control parameters in agent ontologies could be combined within a common XACML policy document that is used through different MAS applications. Agent-based access control requirements and common XACML policy documents should be consistent to enforce policies for MAS. To obtain this condition, the translation of organizational policies and platform based policies have to be considered in detail and the verified policy features have to be enforced in MAS to provide access for resources.

1. Introduction

Traditional access control approaches use users, groups or roles in information systems. However, organizational structure of MAS needs organizational policy based approaches in addition to traditional vision. The different multi-agent systems that have to obey same rules should agree on common policies to ensure the non-existence of conflicts.

In our architecture, flexible characteristic of XACML and semantic structure of MAS ontologies have been used together. They have been adapted to data layer by translation of XACML and OWL to description logic (DL) concepts. However, business logic layer as in the multi-agent system architecture and XACML framework could not be directly aware of the translation support. Our proposal uses XACML-DL for a flexible and granular MAS-based access control approach to express the translation process.

The remainder of this paper is organized as follows. Section 2 explains the related work for semantic based privacy and access control approaches. Section 3 presents the technologies that have been adapted to our proposal. Section 4 provides system architecture elements and explains in detail. Section 5 describes a case study to understand architecture interactions. Finally, section 6 concludes the paper.

2. Related Work

Kolovski et al. [9] proposes mapping of WS-Policy documents to a description logic based subset of OWL, OWL-DL. OWL reasoners can be used for verification and analysis of policies. Because of WS-Policy determines the usage between client and service based policy rules to connect endpoints, there is a need for more expressive model transformation approaches as combination of XACML and OWL usage for access control.

Our approach is a solution for multi-agent systems which uses [1] and [9]. Agent domain ontology and agent security ontology usage for access control has been combined and used with a general platform-independent XACML policy set. During this process, the translation of XACML to DL based ontological definitions have been fulfilled as in [8].

3. Proposed Architecture

The proposed architecture is shown in Figure 1. This infrastructure contains four basic parts as multi-agent system structure, XACML-ontology translation, distributed policy warehouses and XACML framework, which are explained below.

0730-3157/09 $25.00 © 2009 IEEE
DOI 10.1109/COMPSAC.2009.114

IEEE computer society

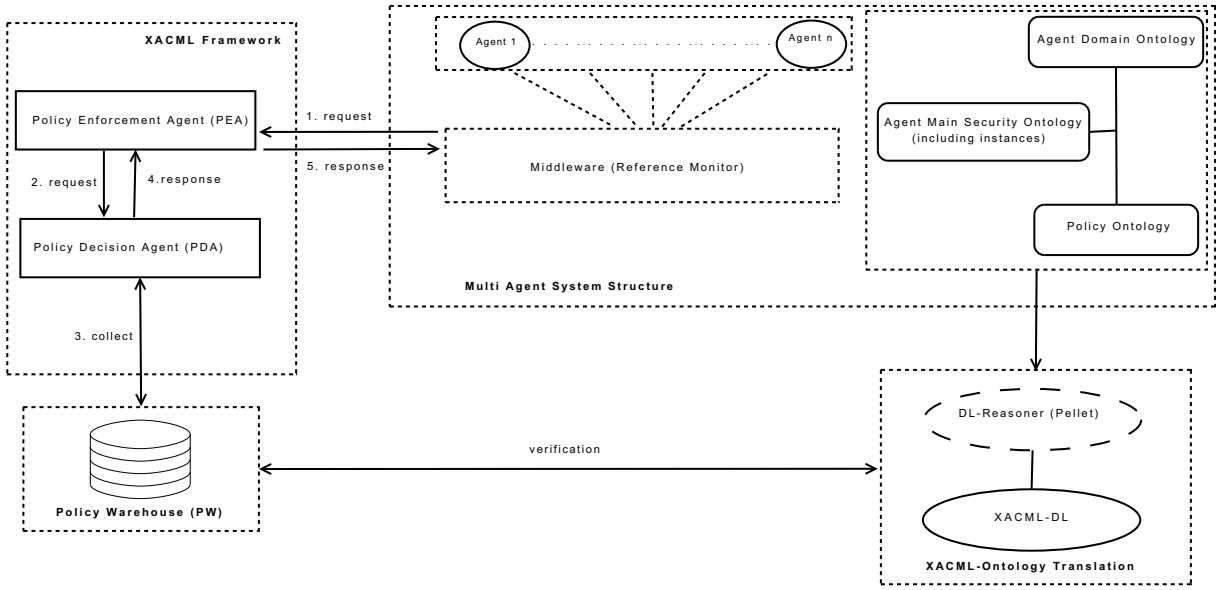

Figure 1. The architectural view of a multi agent system using XACML for access control

3.1. Multi-Agent System Structure

The proposed MAS structure is composed of agents, reference monitor, agent domain ontology, agent main security ontology and policy ontology.

Agents request XACML framework to obtain access privileges for communicating with other agents in the system [11]. If XACML framework responses positively, the requestor agent could have the access right for its goal to interact with the related agent. Agents in MAS operate according to their implicit actions and behaviours nearby the ontologies in the system structure.

Reference monitor acts as a handler to capture agent requests and transform these requests to communicate with XACML framework. It also behaves like a mediator between MAS ontologies, XACML framework and agents to manage access control operations.

Agent domain ontology represents the basic concepts that has been shared through different agents according to domain.

Agent main security ontology is based on the main security ontology that has been defined in [6]. Security objectives define goals that satisfies policy for the systems to assure security constraints. Mechanisms and protocols enforce the policies to realize the goal state successfully.

Policy ontology is based on [4] and [6] which define capabilities for consumer and producer-side semantic web services whereas our approach simply seems like this approach. All agents describe their security-related capability information to allow requestor agent for querying semantic policy definitions. Security requirements define the desired security policies that would be needed to access the related agent.

In a multi-agent system, there has been platform-based and agent-based security policies. Platform-based security policies imply that all of the agents have been affected by them directly or indirectly. Agent-based policies interest merely the related agent. For example, platform-based certificates could have been mandatory for the interaction of open multi-agent system platforms as a platform-based policy. Otherwise, a specific agent could be obliged to certify itself to interact with the desired agent.

3.2. XACML-Ontology Translation

Kolovski et al. [7], [8] use Description Logics (DL) to provide a formalization of XACML. DL are a family of knowledge representation languages which are decidable subsets of First-Order logic commonly used as the formal bases of object/class style ontology languages. Expressing policies in DL allows us both to define and effectively implement an array of policy analysis services for a fairly large subset of XACML. Formalization in [7] also benefits from Description Logics being the basis for the Web Ontology Language (OWL), which allows to use off-the-shelf OWL reasoners as policy analysis tools and to potentially integrate ontology-based policy descriptions with XACML policies.

Approaches in the work of [7], [8] is based on analysis services used at design time of a system. In this paper,

these analysis services translation phase of XACML to DL formalism have been combined with agent-based ontologies to create a hybrid solution for XACML and ontology usage for policies in a distributed environment.

Pellet is an open-source DL reasoner that is based on the tableaux algorithms developed for expressive Description Logics. Pellet supports the full expressivity of OWL DL and is only sound and complete DL reasoner that can handle this expressivity [10]. The aim of Pellet in our proposal is to express ontological concepts in DL perspective to compare with XACML-based documents.

XACML-DL approach maps a large fragment of XACML (including core XACML, XMLSchema datatypes and the Administrative Policy profile) to Description Logics (DL). This mapping causes DL reasoners to provide analysis services for XACML documents. These services are provided by basic software engineering principles (e.g. unit testing to verify policies). While using this approach, XACML-DL users don't need to be aware of details for the logic formalism and XACML mapping [7], [8].

In our approach, the architecture is divided into two layers as business logic layer (multi-agent system structure and XACML framework) and data layer (XACML-ontology translation and policy warehouse(s)). In data layer, the ontological platform and agent-based policies have been translated to DL based descriptions. Then translated policy constraints have been compared and verified to achieve consistency among security policies. If the verification is successful, business logic operations could be executed. If there is a conflict between requestor's information and data layer translations, then the XACML framework will warn authorities in the system for conflict. If the conflict has not been solved in XACML framework, related agent's access request is rejected without need further XACML enforcement. So that the business-logic and data layers could determine their tasks with the help of other layer. However, there is no dependency between these layers according to software paradigms.

3.3. Policy Warehouse

Policy warehouse (PW) is the data repository definition of distributed policies in an open environment. There could be one or more policy warehouses in the system and the XACML-ontology translation have been operated with the related organizational warehouses. These warehouses include organizational domain and security policies. In a conference management system, review right of the program committee member for a paper defines organizational policy in Figure 2.

```
<Policy
    PolicyId="ConferenceOrganizationPolicy1"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-
applicable">
<Target>
<Subjects>
   <Subject>
      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
   DataType="http://www.w3.org/2001/XMLSchema#string">ProgramCommitteeMember</AttributeValue>
          <SubjectAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
       </SubjectMatch>
       </Subject>
</Subjects>
<Resources>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">Paper</AttributeValue>
        <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
</Resources>
<Actions>
    <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">review</AttributeValue>
            <ActionAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ActionMatch>
    </Action>
</Actions>
```

Figure 2. XACML case study example in policy warehouse

3.4. XACML Framework

XACML [5] framework has three basic components as Policy Enforcement Point (PEP), Policy Decision Point (PDP) and Policy Administration Point (PAP). PEP intercepts access requests and transmits these requests to PDP. PDP retrieves and evaluates these policies. PAP(s) have the task to specify and store policies [11]. PAP(s) have been defined as PW in our approach.

The agents in multi-agent system requests XACML framework to obtain an access for an element in the system. In this proposal, there has been shown just one Policy Enforcement Agent (PEA) for simplicity. In a realized distributed environment, there are a lot of PEAs around the network. Then XACML framework passes the agent's request to related PEA(s) according to the PEA's organizational information. Then PEA enforces the request and sends the related information to Policy Decision Agent (PDA). PDA responses the final decision information to PEA(s). If there is more than one PEA in the system, PDA sends all PEAs the decision information and chooses a communicator PEA to send decision to the multi-agent system structure.

PDA collects all enforcement information coming from PEA(s) [2]. Then it controls if any conflict between the enforcement information. If there is a conflict, PDA warns related PEAs by using its exceptional case ontology to correct the problem. Then related PEA(s) updates their internal behaviours and informs PDA again until there isn't any conflict for the system. When there is no conflict between the collected information, PDA responses policy based decision to PEA(s).

4. Case Study

Case study presents a conference management system to explain the architecture better. The proposed conference managent system is is composed of following elements:

Program committee member agent requests a paper review (Agent Domain Ontology) by using their username/password or X.509 certificate (Agent Main Security Ontology and Policy Ontology) capabilities.

When the requestor triggers reference monitor, reference monitor accesses the ontologies' knowledge base and related ontological information with request's content.

There are program committee members and a program committee chair who is also a program committee member within the ontology. These members review papers and decides the acceptance as short paper, regular paper or a poster. Representation of these concepts in an ontology causes machine interpretable semantic meanings of them that also supplies a specific platform rule to map organizational specifications.

Program committee members that have username/password and a related authentication protocol capabilities could be concerned to review and comment papers in the system. They also require program committee chair agents to certify themselves to trust their interaction behaviors.

Pellet is used to query and reason about agent-based policies ontology and other ontologies in multi-agent system structure.

Policies located in PW are translated from XACML to DL using XACML-DL. In Figure 2, program committee members' review action for the papers have been shown.

Program committee member (organizational policies) can review papers and has also a capability to authenticate and require certification (security policies) from MAS.These policies have been verified by the sources coming from PW and MAS ontologies by XACML-DL.

After verification, reference monitor is prepared to send access control request to XACML framework in suitable format and requests XACML framework (Figure 1-Step1).

XACML framework passes request message to all related PEA(s). PEA(s) enforce policies and send their results to PDA.

PDA collects the data from PEA(s) (Figure 1-Step2) and PW(s) (Figure 1-Step3). It gives a final decision and sends this decision to all PEA(s) (Figure 1-Step4). But one PEA is chosen as a head PEA to communicate with XACML framework, also to the MAS structure (Figure 1-Step5).

Positive response from XACML framework has been transformed to agree program committee member agent. Then this agent could review papers according to its capabilities and requirements in MAS.

5. Conclusion and Future Work

In this paper, we presented an XACML and ontology-based access control mechanism by using software agents. By the way, the organizational policies located in PW and agent-based policies located in multi-agent system structure have been combined and verified in a data-layer approach. Formalization of XACML and OWL under DL concept helped the verification of policies under a common point.

We have a goal to define and implement XACML-policy framework and policy warehouse with open-multi agent system concepts as norm for the future work. Then agent policy, platform policy, organizational policy and role-based access control policy concepts would be determined in a detailed architecture. Also adaption of Or-BAC [3] terms to agent terminology (e.g. provisional context) for agent organizations will be realized.

References

[1] D. D. I. Abou-Tair and S. Berlik. An ontology-based approach for managing and maintaining privacy in information systems. In *OTM Conferences (1)*, 2006, pp. 983-994.

[2] C. A. Ardagna, E. Damiani, S. D. C. D. Vimercati, and P. Samarati. A web service architecture for enforcing access control policies. In *Proc. of the First Int. Workshop on Views on Designing Complex Architectures, Electronic Notes in Theoretical Computer Science*, 2006, pp. 47-62.

[3] F. Cuppens and A. Miège. Modelling contexts in the or-bac model. In *ACSAC '03*, Washington, DC, USA, 2003, pp. 416.

[4] G. Denker, L. Kagal, T. W. Finin, M. Paolucci, and K. P. Sycara. Security for daml web services: Annotation and matchmaking. In *International Semantic Web Conference*, 2003, pp. 335-350.

[5] eXtensible Access Control Markup Language (XACML). Oasis. Feb 2005.

[6] A. Kim, J. Luo, and M. H. Kang. Security ontology to facilitate web service description and discovery. *J. Data Semantics*, 9:167–195, 2007.

[7] V. Kolovski, J. Hendler, and B. Parsia. Formalizing xacml using defeasible description logics. In *Technical Report TR-233-11, University of Maryland - College Park*, 2006.

[8] V. Kolovski, J. A. Hendler, and B. Parsia. Analyzing web access control policies. WWW, 2007, pp. 677-686.

[9] V. Kolovski, B. Parsia, Y. Katz, and J. A. Hendler. Representing web service policies in owl-dl. In *International Semantic Web Conference*, 2005, pp. 461-475.

[10] Pellet. Pellet-owl dl reasoner. In *http:// clarkparsia.com/ pellet/download*.

[11] T. Priebe, W. Dobmeier, C. Schläger, and N. Kamprath. Supporting attribute-based access control in authorization and authentication infrastructures with ontologies. *Journal of Software (JSW)*, 2(1):27–38, 2007.