Gazi University Journal of Science
GU J Sci
23(4):465-474 (2010)

GUJS
www.gujs.org

Personalizable Ontology-Based Access Control

Özgü CAN[1]♠, Okan BURSA[1], Murat Osman ÜNALIR[1]

[1] Ege University, Faculty of Engineering, Department of Computer Engineering, 35100, Izmir, TURKEY

Received: 19/03/2010 Revised: 06/04/2010 Accepted:14/05/2010

ABSTRACT

The main idea of Semantic Web is creating web pages which are also understood by machines and using ontologies to unify data. Improving a secure Semantic Web is one of the main works in Semantic Web research area. For this purpose, policies are used. Policy is a set of rules and provides an access control mechanism for a resource without making any change in that resource. Policy management in Semantic Web is used to define rules for accessing a resource and to provide users to interpret and comply with these rules. One of the key features to develop successful personalized Semantic Web applications is to build user profiles. In this paper, we developed an Ontology-Based Access Control (OBAC) model. This model represents domain and profile information semantically and has a profile based policy approach in order to achieve a personalized policy management for Semantic Web. We store personal information in profiles and model this information semantically to make it part of access control model. Thus, we created two kinds of policies: domain and profile based policies. We implemented an Ontology-Based Access Control application which creates, modifies, and deletes policy ontologies. Policy conflicts are also resolved to provide fine-grained policies in OBAC model. The main contributions of this work are: defining semantically rich resource and entity policies for an Ontology-Based Access Control mechanism and making use of these policies in terms of the personalization scope.

Key Words: *Semantic Web, Ontology, Policy, Profile, Personalization, Conflict Resolution*.

1. INTRODUCTION

Semantic Web allows machines to communicate with each other and provides sharing and reusing of the information by using formal semantics. Thus, while in today's web users decide their navigation choices by reading web pages, in Semantic Web, agents in behalf of their users make decisions by using common ontologies and description languages.

Ontologies are common definitions for entities. As ontologies are needed for the standardization of the definition of different terms, they are also important for the understanding of web pages by machines.

Sharing the information brings forth some security needs like privacy, access control, authentication, authorization and data integrity. Hence, effective mechanisms are needed to ensure the security of Semantic Web technologies. Therefore, improving a secure Semantic Web is one of the main works in Semantic Web research area.

In secure systems, access to data is controlled and the management of information is ensured. For this purpose, policies are used. Policy is providing an access control mechanism for a resource without making any change in that resource. Recently, two parallel issues are handled in access control area: to develop new access control models

♠Corresponding author, e-mail: ozgu.can@ege.edu.tr to meet the policy needs of real world application domains and to develop policy languages for access control [1].

Rei [2, 3] and KAoS [2, 4] are two of the most known semantic web based policy languages used for policy specification. In our work, we used Rei policy language for policy specification.

Rei is a policy specification language based on OWL-Lite. It allows users to express and represent the concepts of rights, prohibitions, obligations, and dispensations [2, 5]. Rei allows developers to express policies over domain-specific ontologies in e.g., RDF and OWL. Rei has a set of speech acts primitives that allow the system to exchange rights and obligations between entities. It provides a Prolog policy engine that reasons about the policy specifications. Meta policies are used to resolve policy conflicts that the Rei policy engine encounters [3]. The engine accepts policy specification in both the Rei language and in RDF-S, consistent with the Rei ontology [2]. Rei ontologies can be found at http://www.cs.umbc.edu/~lkagal1/rei/ontologies.

Besides policy languages, another issue for access control area is access control models. One of the main traditional access control model is Role Based Access Control (RBAC). In RBAC model, the security policy gives permissions directly to roles, not to the user. A user obtains her permissions according to his/her roles defined in the system. Thus, a user will inherit all the permissions associated with his/her roles and this role hierarchy also simplifies the definition of policies [6]. For example, in a hotel system, the existing roles can be: manager, desk officer, staff and guest. If a user's role is assigned as a desk officer then the user is granted to access the room files, while the hotel manager role can access both the hotel's room and accounting files. However, RBAC has some limitations like: administrative tasks needed for user-to-role and permission-to-role assignments and also the exponential grow of the number of roles and permissions makes these assignment tasks expensive [7]. These limitations of RBAC led us to use profile based policies.

Our approach is a profile based policy approach. Access to information and service can be achieved in various ways depending on the user profile. A user can create and update her profile. Modeling user profiles is an important issue of personalization. Various methods exist to collect user information and to create user profiles from this information. User information can be provided explicitly or implicitly [8]. Online registration forms, questionnaires and reviews are explicit profiling, which is also called static profile. In implicit profiling, which is also called dynamic profiling, preferences of each user are recorded and analyzed through internal devices like cookies and web server log files without the sense of the user.

In our approach, we define two kinds of policy which are based on Rei policy language. One of these policies is for the related domain and the other policy is for the profile. The domain policy defines rules for the related domain, such as a hotel policy rule like "Smoking is not allowed in guestrooms". The profile policy defines rules for a chosen profile. For example, a rule which specifies that "A person who has a diabetic profile can eat salmon" is a rule for a diabetic profile.

The paper is organized as follows. Section 2 presents an Ontology-Based Access Control (OBAC) under the domain and profile based policy approaches. We outlined how policy conflicts are resolved. In Section 3, we explained the implementation details of our Ontology-Based Access Control model and gave a brief result of OBAC model. Finally, conclusion and future work follows.

2. MATERIAL AND METHOD

Policies are encountered in every area of our daily lives. There are various types of policies like access control, education, government and health policies. Policy is a statement that defines the behavior of a system. It acts as both a decision-support system and a declarative behavior system [9]. Semantic Web based policy languages allow policies to be described over heterogeneous domain data and promote common understanding among participants who might not use the same information model [1].

In Semantic Web, ontologies are also used to define policies. Ontology is a formal explicit specification of a shared conceptualization [10]. Ontologies are used to represent information and to model specific domain information by defining objects, concepts and relationships. Therefore, standard conceptual vocabularies can be defined for information exchange between systems, by this way, information can be reused, and services answer queries to simplify interoperability between heterogeneous systems.

Policy management in Semantic Web is used to define rules for accessing a resource and to provide users to interpret and comply with these rules. Thereby, semantically-rich policy representations reduce human error, simplify policy analysis, reduce policy conflicts, and facilitate interoperability [2].

We developed an Ontology-Based Access Control model to create, modify and query semantically-rich policies. This model accesses the data by using a semantic based approach. In present models, there is no metadata knowledge for the resource to be accessed and the entity which is going to access that resource. In OBAC, policies are created based on resource and entity metadata. Triples, which are part of policies, are represented in our model as: the entity which is going to access the resource is the subject, the resource itself is the object and policy objects are the predicate.

A policy consists of policy rules. Rules are used to define policies. Positive rule like "User A can read B1.doc and write to C1.doc" and negative rule like "User A can't write to A1.doc and can't read D1.doc" are examples of a policy rule. Policy rules are formed by policy objects. Policy objects are also called deontic objects. These policy objects are:

- *Permission:* Permission is what an entity can do.
- *Prohibition:* Prohibition is what an entity can't do.
- *Obligation:* Obligation is what an entity should do.
- *Dispensation:* Dispensation is what an entity need no longer do.

Users have different roles in daily life. These roles have different preferences and properties. Individual users vary so much that a model of canonical user is insufficient, thus, models of individual users are necessary [11]. For this purpose, in this work, a profile based policy approach is presented instead of using a role based approach. In order to provide a profile based policy management, each user must be handled as a different user and must have different user models. The "user model" is used to describe a wide variety of knowledge about people [11]. Users who create and update their profile are participants of the system. The entity of a policy is represented with a profile. Profile is used in spite of entity to give detailed information of the subject. A policy is shown with a triple (P, O, A), in which P is profile, O is object and A is action. Profile indicates the user who wants to access to a resource, object indicates the resource which is going to be accessed and action indicates the operations which a user wants to achieve on a resource.

The relationship between profile, policy objects, action and object is shown in Figure 1. Every policy object is related with a profile, action and object.

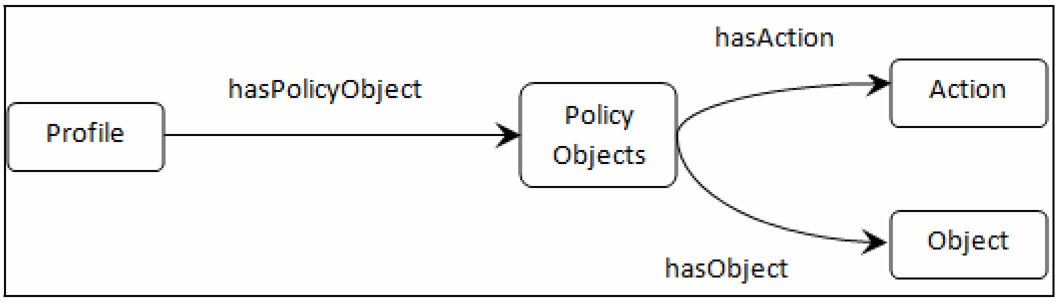

Figure 1. The relationship between profile, policy objects, action and object.

Policy ontologies, which include policy objects, subjects and objects, are all created separately in OBAC model. Figure 2 shows OBAC policy components. Subject is represented with profile and profiles come from the profile ontology. Condition is a constraint that describes under which conditions a policy will be executed. Condition is based on domain ontology. Policy, action and condition triple is used to form policy objects.

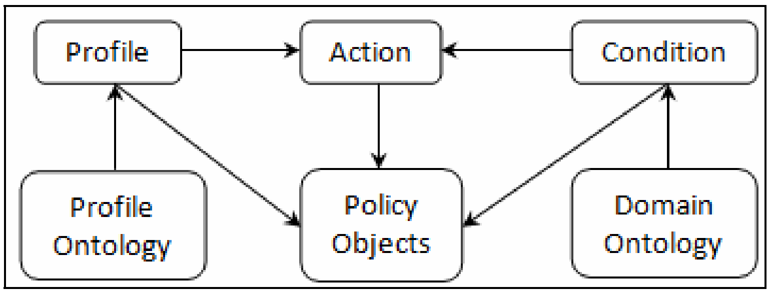

Figure 2. OBAC policy components.

Speech acts allow decentralized security control. We implemented four speech acts:

- *Delegate:* Delegate speech act gives a right to another entity or group of entities.
- *Revoke:* Revoke speech act removes a right.
- *Cancel:* Cancel speech act cancels a request.
- *Request:* Request speech act makes a request for a right.

In Ontology-Based Access Control model, we defined two kinds of policies: domain and profile based. Domain data is used to compose domain and profile based policy ontologies. Policy ontologies are created with Rei policy language. Besides, profile based policies are created by using a profile ontology which is based on a meta-profile ontology.

2.1. Domain Based Policies

Domain ontology is the machine-processable representation of the concepts in a specific area. A domain may be a department, an enterprise, a group or a project. Domain knowledge allows the specification of a policy. In domain based policies, policies are created according to the related domain's rules. Domain rules are represented by a ternary relationship between user, condition and object. Rules define constraints that must be satisfied. A rule has the form:

$B_1, B_2, \ldots, B_n \rightarrow A$ *(If* $B_1, B_2, \ldots, B_n$ *hold then A holds.)*

In this form, *A* is the head of the rule and $B_1, B_2, \ldots, B_n$ are the premises of the rule. The set $\{B_1, B_2, \ldots, B_n\}$ is also called the body of the rule [12]. For example;

mother(selin, can) → *family (selin, can)*

If Selin is Can's mother, then Selin is the family of Can.

works(can,bornova), resides(can,alsancak), location (alsancak, izmir), location(bornova, izmir) → *lives(can, izmir)*

If Can works in Bornova, resides in Alsancak and also, Alsancak and Bornova are in Izmir, then Can lives in Izmir.

Our domain is based on tourism domain concepts and domain based policies are created over this domain. We are mapping the concepts of tourism domain to concepts in the policy ontology, so we are using these concepts to express domain based policies.

Example rules of a hotel domain can be written as below: *Pets are allowed in Hotel Louvre. (Permission)*

$$Accomodation(HotelLouvre) \wedge petsAllowed(HotelLouvre, true) \rightarrow Permission(OwningPet, HotelLouvre)$$

Guests can't use wireless connection without any extra payment in Hotel Seine. (Prohibition)

$$Accomodation(HotelSeine) \wedge wirelessConnection(HotelSeine, true) \wedge wirelessConnection(Guests, true) \wedge extraPayment(Guests, false) \rightarrow Prohibition(InternetAccess, HotelSeine)$$

Figure 3 shows the policy definition of the prohibition domain policy rule above.

```
<deontic:Prohibition rdf:ID="Prohibition_Seine_WirelessConnection">
   <policy:desc rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Guests can't
        use WirelessConnection without any extra payment.
   </policy:desc>
   <deontic:constraint>
       <constraint:And rdf:ID="GuestAndNoExtraPayment">
        <constraint:second>
         <constraint:Not rdf:ID="Not_extraPayment">
          <constraint:first>
           <constraint:SimpleConstraint rdf:ID="extraPayment">
            <constraint:object
              rdf:resource="http://efe.ege.edu.tr/~odo/Ontology/I_Tour_mdfy.owl
              #Seine"/>
            <constraint:predicate rdf:resource="#makePayment"/>
            <constraint:subject rdf:resource="#Guest"/>
           </constraint:SimpleConstraint>
          </constraint:first>
         </constraint:Not>
        </constraint:second>
        <constraint:first rdf:resource="#any_Guest"/>
       </constraint:And>
   </deontic:constraint>
   <deontic:actor rdf:resource="http://efe.ege.edu.tr/~odo/Ontology/Profile.owl#Guest"/>
   <deontic:action rdf:resource="#Seine_InternetAccess"/>
</deontic:Prohibition>
```

Figure 3. Domain based prohibition policy definition example.

Our case study is based on e-tourism ontology (http://e-tourism.deri.at/ont/e-tourism.owl) which is improved by DERI (Digital Enterprise Research Institute). We extended the tourism domain ontology within the case study. We added new object properties, data properties and individuals. The extended version of the tourism ontology can be found at http://efe.ege.edu.tr/~obac/I_Tour_mdfy.owl.

2.2. Personalizable Access Control

Personalizable access control specifies whether a profile can perform certain actions on an object. In this section, we first expressed the user profiling model, how these profiles are created and then present profile based policies.

2.2.1 Personalization and Profile Management

Modeling user profiles is an important part of the personalization. Personalization can be achieved by filtering out irrelevant information and identifying additional information of likely interest of the user [13]. There are various methods to collect user information and to compose user profiles from this user information. User information can be obtained explicitly or implicitly [8]. Online forms, questionnaires and reviewing are explicit information. This information is structured in order to create static profiles. In implicit profiling, which is also called dynamic profile, preferences and behaviors of each user are recorded and analyzed through cookies and log files. In Figure 4, implicit and explicit profile structures can be seen [8].

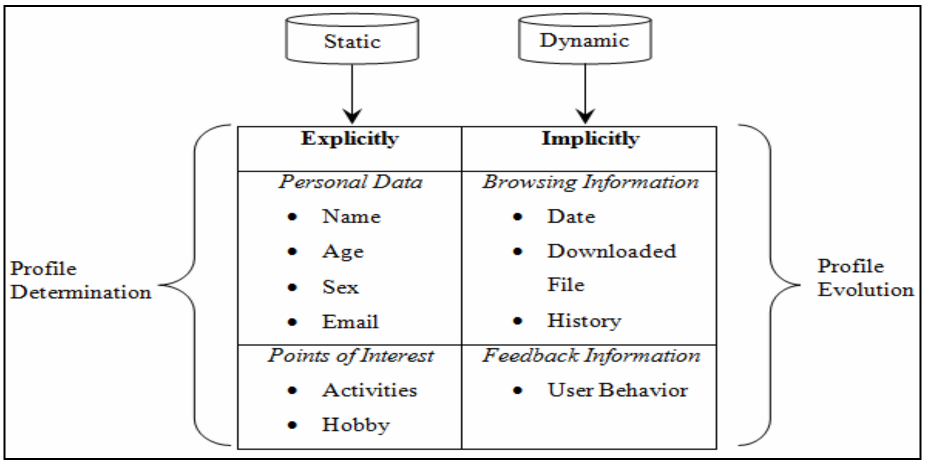

Figure 4. Static and dynamic profile structures.

In our approach, we use FOAF (Friend Of A Friend, http://www.foaf-project.org) ontologies to store static user profiles. Thus, the static structure of FOAF ontologies forms an appropriate environment for our user profiling approach. Canonical properties determine the individual set of the specific user profile. In the sense of our approach, these properties are grouped inside our meta-profiles based on their values. For example, users whose age are between 18 and 35 are in "Young People" profile, users who work as a Professor are in "Academic" profile.

However, grouping profile properties and describing new meta-profiles for the user creates contradictions. Canonical properties of FOAF ontologies cannot be used inside the meta-profiles. Due to their nature, these properties were created as static variables those can be object/data type properties. Thus, in order to use these canonical properties inside our meta-profiles, these properties must be redefined as an interval or a restriction set. It is not easy to use properties as classes. This interpretation needs meta-modeling.

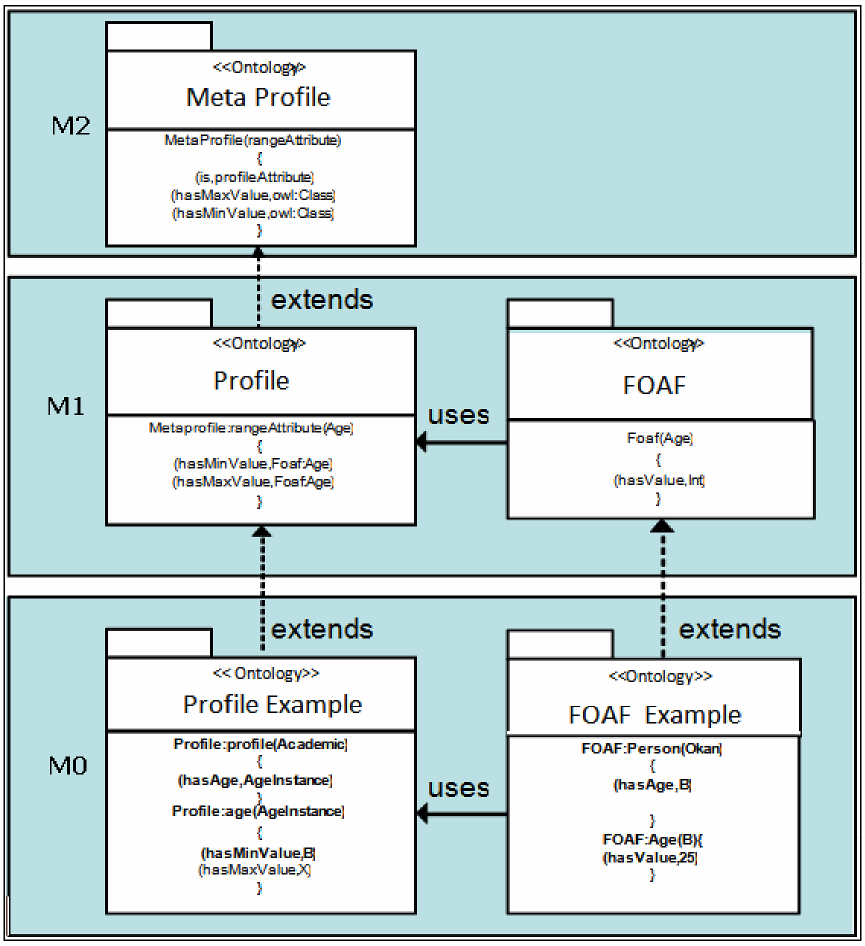

Figure 5. Relationship between Meta-Profile, Profile and FOAF ontologies.

We propose a meta-modeling approach based on FOAF and meta-profiles which can be seen in Figure 5. In this approach, static FOAF descriptions like occupation are turned into a restricted set and properties those are not described inside FOAF are also added. There are no max/min values for data type definitions inside OWL, however, some properties like age must be defined as numerical interval. We define a meta-metaprofile M2, to define different kinds of data type/object type intervals and restrictions. Further, these defined properties are available for the use of meta-profile definition in M1 level. As the meta-profile properties and objects are defined, meta-profile classes are instantiated in M0 level. These instances are the meta-profiles like "Young People" and "Academic". For example, as seen in Figure 6, academic profile that is defined in profile ontology has three properties: age (hasAge), occupation (hasOccupation) and name (hasName). Age instance value of an academic profile is defined by the maximum (AcademicAgeValueMax) and minimum (AcademicAgeValueMin) age properties of meta-profile ontology. "ageValue" data property of FOAF ontology is used to define these min/max values. Also, "can-be" data property is used to define occupation instance and "name" data type property is used to define the academician name instance.

```
<metap:Age rdf:ID="AcademicianAge">
  <metap:hasMinValueAge
rdf:resource="AcademicAgeValueMin"/>
  <metap:hasMaxValueAge
rdf:resource="AcademicAgeValueMax"/>
</metap:Age>
<foaf:Age rdf:ID="AcademicAgeValueMin">
  <foaf:ageValue>22</foaf:ageValue>
</foaf:Age>
<foaf:Age rdf:ID="AcademicAgeValueMax">
  <foaf:ageValue>67</foaf:ageValue>
</foaf:Age>
<metap:Occupation
rdf:ID="AcademicianOccupation">
  <foaf:canbe>Professor</foaf:canbe>
</metap:Occupation>
<metap:Profile rdf:ID="Academician">
  <metap:hasName
rdf:resource="AcademicianName"/>
  <hasAge rdf:resource="AcademicianAge"/>
  <hasOccupation>
   <rdf:Description
rdf:about="#AcademicianOccupation"/>
  </hasOccupation>
</metap:Profile>
```

Figure 6. Academic profile example in Profile ontology.

2.2.2. Profile Based Policies

In OBAC, domain rules are assigned to users. Policy determines the ideal behaviors of the user with using the user profile information. Profiling is creating the set of same type of users. Profiles can help in tailoring information delivery to the specific user [14]. As the user profiles share common properties, grouping of user profiles based on their interests make new sets of users.

Profile rules are as follows:

A company employee profile can not stay in Eiffel view rooms. (Prohibition)

$$Profile(CompanyEmployee) \wedge hasView(Guestroom, Eiffel) \rightarrow Prohibition(StayIn, Guestroom)$$

A tourist profile can use steam bath if she pays an extra bill. (Obligation)

$$Profile(Tourist) \wedge hasPayment(extraBill, true) \rightarrow Obligation(Use, SteamBath)$$

In profile based policy ontology, profile instances of the profile ontology are used instead of the instances of "entity:Variable" class as the actor of an action. Entity:Variable is a class of ReiEntity ontology. In this case, there is no mapping between entity:Variable class and profile ontology. Entity:Variable class holds entity examples of a policy ontology. While defining profile based policies, these entities are taken from semantically rich profile ontology instead of entity:Variable class. http://efe.ege.edu.tr/~obac/Profile.owl is used as profile ontology and http://efe.ege.edu.tr/~obac/MetaProfile.owl as meta-profile ontology. Figure 7 shows the prohibition policy definition of the above prohibition profile rule. Here, deontic:actor instance, which defines the actor of the policy action, comes from profile ontology.

```
<deontic:Prohibition rdf:ID="Prohibition_EiffelViewRoom_HiltonParis">
        <deontic:action rdf:resource="#StayingInEiffelViewRoom_HiltonParis"/>
            <deontic:actor rdf:resource="http://efe.ege.edu.tr/~ozgucan/Gazi/ontology/
              Profile.owl#CompanyEmployee"/>
        <policy:desc rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
         Company employees can't stay in Eiffel view rooms in Hilton Paris.
        </policy:desc>
        <deontic:constraint rdf:resource="#any_CompanyEmployee"/>
</deontic:Prohibition>
```

Figure 7. Profile based prohibition policy definition example.

2.3. Conflicts over Policies

Conflict occurs if policies are about the same action, on the same target but the modalities are different [15]. Policy conflicts can arise due to omissions, errors or conflicting requirements [16]. Specifying priorities and precedence relations are the regulation of conflicting policies [2]. We use meta-policy specifications of Rei policy language to resolve conflicts. Meta-policies are policies about how policies are interpreted and how conflicts are resolved dynamically.

Policy conflicts occur in two manners: policy conflict and rule conflict. Policy conflicts are resolved with specifying priorities and precedence relations. In specifying priorities, priorities between policies and rules are defined. For example, “Rule1 overrides Rule2.” or “PolicyA overrides PolicyB.” In precedence relations, precedence of negative modality and precedence of positive modality are defined. In precedence of negative modality, prohibition has precedence over permission and dispensation is stronger than obligation. In precedence of positive modality, permission has precedence over prohibition and obligation is stronger than dispensation.

If a conflict occurs, the ultimate decision is given by the policy. For example, the hotel policy rules are like “Conference participants can use snack bar.” and “Tourists can’t use snack bar.” The user can be either a tourist or a conference participant. If the user operates with her conference participant profile then the final choice will be permission, but if the user operates with her tourist profile then there will be a restriction for using snack bar. To resolve this conflict, deontic conflict resolution algorithm, which is shown in Figure 8, is used. Inside this algorithm, if the action property of prohibition and permission rules are equivalent then rule priorities are generated based on user properties. Figure 9 shows the generated “ruleofLesserPriority” and “ruleOfGreaterPriority” meta-policy object properties.

```
deonticConflictResolution
    for each individual of deonticClass of Policy
        if (deonticProhibitionActionProperty==deonticPermissionActionProperty)
            print conflictError
            choice ← (“PermOverridesPro”|| “ProOverridesPer”)
            if (choice==”PermOverridesPro”)
                metaPolicy_ruleOfGreaterPriority ← deonticPermission
                metaPolicy_ruleOfLesserPriority ← deonticProhibition
            else if (choice==”ProOverridesPer”)
                metaPolicy_ruleOfGreaterPriority ← deonticProhibition
                metaPolicy_ruleOfLesserPriority ← deonticPermission
```

Figure 8. Rule conflict resolution algorithm.

```
<metapolicy:rulePriority rdf:ID="ProhOverridesPerm_SnackBar">
            <metapolicy:ruleOfLesserPriority rdf:resource="#Permission_SnackBar"/>
            <metapolicy:ruleOfGreaterPriority rdf:resource="#Prohibition_SnackBar"/>
</metapolicy:rulePriority>
```

Figure 9. Rule conflict resolution

We also set priorities between policies which might cause a policy-policy conflict in the model. Thus, we are defining meta-policy rules for conflicting policy ontologies. The meta-policy rule in Figure 10 is an example to a policy-policy conflict where CompanyEmployee Policy overrides TouristPolicy. “policyOfLesserPriority” and “policyOfGreaterPriority” meta-policy object properties are used to resolve policy conflict. A policy-policy conflict ontology can be seen at http://efe.ege.edu.tr/~obac/PolicyPolicyConflict.owl.

```
<metapolicy:rulePriority rdf:ID="CompanyEmployeePolicyOverridesTouristPolicy">
        <metapolicy:policyOfLesserPriority rdf:resource="#TouristPolicy"/>
        <metapolicy:policyOfGreaterPriority rdf:resource="#CompanyEmployeePolicy"/>
</metapolicy:rulePriority>
```

Figure 10. Policy conflict resolution.

In conflict resolution, we provide fine-grained access control policies by defining user specific policies. When the profile set of a user grew, the user can have different profiles. In this situation policy rules and policies can conflict due to different policies of different profiles. The access rights between the resource and the user must not be conflict. We are controlling this conflict by specifying priorities and precedence relations.

3. IMPLEMENTATION AND RESULTS

In this work, we present an Ontology-Based Access Control model which has a profile based approach in order to achieve a personalized policy management. We emphasize that both data and people must be understood for the success of an access control mechanism. In our work, we represent data semantically. However, to provide a complete ontology based application, we also have to represent people’s information semantically. For this purpose, we store personal information in profiles and model this information semantically to make it part of our Ontology-Based Access Control model. We also use profiles to provide personalization which is not possible by using roles. In the literature, roles are used for access control mechanisms and personal information is not represented semantically. The main contribution of our work is representing personal information semantically in an access control model and using this information to achieve personalization in policy management.

In OBAC, we define two kinds of policies: domain and profile based policies allowing the system to behave according to the specific requirements of users for better socialization. Thus, OBAC model provides personalized query results to users.

In our work, firstly, we created policy ontologies by using Rei policy language. Protégé ontology editor is used to create ontologies. After creating policy ontologies, we developed an Ontology-Based Access Control application to create, modify, delete and query policies. Java programming language is used to implement the application. Jena Semantic Web Framework (http://jena.sourceforge.net) for building Semantic Web applications is utilized to interpret and reason over policies. Jena provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine.

3.1. Running Example & Comparative Analysis Of RBAC and Personalized OBAC

In this section, we will give a brief running example of personalized OBAC model for a university domain. Some individuals of profiles, resources and actions in a university domain can be seen in Table 1.

Table 1. Profile, resource and action individuals of a university domain.

Profiles			**Resources**	**Actions**
Faculty Member	Professor		• Course Web Sites • FTP Service • Labs • Printer • Fax	• Accessing course materials and FTP service • Usage of library, labs, printer and fax.
	Associate Professor			
	Assistant Professor			
	Lecturer			
	Research Assistant			
Students	Undergraduate			
	Graduate			
	Master			
	PhD			
Staff	Secretary			
	Officer	Student Affairs		
		Personal Affairs		
		Accounting		
		Library		
	Technician			

Users choose their profiles and they can have more than one profile. For example, a user can be a PhD student and research assistant at the same time. In this situation, policy conflicts may occur for different profiles due to policy rules like: “PhD students can’t use Semantic Web Lab printer” and “Research Assistants can use Semantic Web Lab printer”. So, the same user has permission and prohibition on using printer action. OBAC model determines and resolves these policy conflicts by specifying priorities and precedence relations. RBAC does not provide to determine and resolve policy conflicts.

User’s profiles can change in time. For example, an assistant professor profile may change into associate professor profile. In this case, this profile change will not affect the model. But, in RBAC, change in user’s personal information will affect the system and making new role assignments will increase administrative tasks expense. In OBAC, profiles are assigned to users through their personal information, if any change happens in the user’s personal information user’s profile changes without any administrative task and policies related to this new profile will be executed. Thus, OBAC has no administrative tasks expense and does not deal with state changes in user’s data.

4. CONCLUSIONS AND FUTURE WORK

Semantic Web based policies are used to access the data in a secure way. Although policies are widely used in access control mechanisms, we use ontology based policy management for achieving personalization in our approach. In this work, we have proposed an Ontology-Based Access Control model for the Semantic Web by specifying policies over domain and profile knowledge. Therefore, we defined two kinds of policies: domain and profile based. We also developed a user interface and policy engine to interpret and reason over policies by using JENA Semantic Web Framework.

As part of our future work, we will add preference based policies to our model in order to achieve a better personalization. Preferences are clustering entities that make preferences. User preferences are constrained by enforcing domain and profile policies, so, users can achieve the best query results. Hence, effective personalization to serve the users based on their requests will be achieved. For example, if a tourist wants to stay in a pets-allowed guestroom, after the enforcement of the hotel domain policy, pets-allowed hotel names will be listed for this user. In addition, we will implement some queries. For this purpose, SPARQL query module of SESAME framework will be used.

REFERENCES

[1] Finin, T. et al., “ROWLBAC - Representing Role Based Access Control in OWL”, ***Proceedings of the 13th Symposium on Access Control Models and Technologies***, Colorado, USA (2008).

[2] Tonti, G., Bradshaw, J. M., Jeffers, R., Monranari, R., Suri, N., Uszok, A., “Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KaoS, Rei, and Ponder”, ***2nd International Semantic Web Conference (ISWC 2003)***, 419-437 (2003).

[3] Kagal, L., Finin, T., Joshi, A., “A Policy Language for a Pervasive Computing Environment”, ***POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks***, 63 (2003).

[4] Uszok, A., Bradshaw, J. M., Jeffers, R., "KAoS: A Policy and Domain Services Framework for Grid Computing and Semantic Web Services", ***Second International Conference on Trust Management***, Springer-Verlag (2004).

[5] Kagal, L., Finin, T., Joshi, A., "A Policy Based Approach to Security for the Semantic Web", ***2nd International Semantic Web Conference (ISWC 2003)***, Sanibal Island, Florida, USA 402-418 (2003).

[6] Cuppens, F., Miège, A., "Modelling Contexts in the Or-BAC Model", ***19th Annual Computer Security Applications Conference*** (2003).

[7] Yuan, E., Tong, J., "Attributed Based Access Control (ABAC) for Web Services", ***In ICWS'05: IEEE International Conference on Web Services*** 569 (2005).

[8] Jrad, Z., Aufaure, M.A., "Personalized Interfaces for a Semantic Web Portal", ***Tourism Information Search, In KES 2007/WIRN 2007***, Part III, LNAI 4694, 695-702 (2007).

[9] Thuraisingham, B., "Building Trustworthy Semantic Webs", ***Auerbach Publications***, ISBN:0849350808 (2007).

[10] Studer, R., Benjamins, V. R., Fensel, D., "Knowledge Engineering: Principles and Methods", ***Data Knowl. Eng.***, 25(1-2): 161-197 (1998).

[11] Rich, E., "Users are individuals: individualizing user models", ***International Journal of Man-Machine Studies,*** 18: 99-214 (1983).

[12] Antoniou, G. and van Harmelen, F., "A Semantic Web Primer", ***The MIT Press***, ISBN 0-262-01210-3 (2004).

[13] Gauch, S., Speretta, M., Chandramouli, A., Micarelli, A., "User Profiles for Personalized Information Access", ***The Adaptive Web 2007***, 54-89 (2007).

[14] Dzbor, M., Motta, E., "Engineering and Customizing Ontologies", ***In Ontology Management, Semantic Web, Semantic Web Services, and Business Applications***, 25-57 (2008).

[15] Kagal, L., "Rei: A Policy Language for the Me-Centric Project", ***TechReport***, HP Labs, September (2002).

[16] Lupu, E. C. and Sloman, M., "Conflicts in policy-based distributed systems management", ***IEEE Transactions on Software Engineering***, November/December 25(6):852–869 (1999).