A COMPREHENSIVE APPROACH FOR SOLVING POLICY HETEROGENEITY

Rodolfo Ferrini and Elisa Bertino
Department of Computer Science, Purdue University, U.S.A.

Keywords: Policy Analysis, Policy Heterogeneity, Ontology Extraction, Ontology Merging.

Abstract: With the increasing popularity of collaborative application, policy-based access control models have become the usual approach for access control enforcement. In the last years several tools have been proposed in order to support the maintenance of such policy-based systems. However, no one of those tools is able to deal with heterogeneous policies that is policies that belong to different domains and thus adopting different terminologies. In this paper, we propose a stack of function that allow us to create a unified vocabulary for a multidomain policy set. This unified vocabulary can then be exploited by analysis tools improving accuracy in the results and thus applicability in real case scenarios. In our model, we represent the vocabulary of a policy adopting ontologies. With an ontology it is possible to describe a certain domain of interest providing richer information than a plain list of terms. On top of this additional semantic data it is possible to define complex functions such as ontology matching, merging and extraction that can be combined together in the creation of the unified terminology for the policies under consideration. Along with the definition of the proposed model, detailed algorithms are also provided. We also present experimental results which demonstrate the efficiency and practical value of our approach.

1 INTRODUCTION

The increasing popularity of collaborative applications and technologies has improved the flexibility and scalability in data provisioning and resource sharing. Access control in such environments is usually enforced by using policy-based access control models supported by specialized services, often based on the principles of the XACML reference architecture. Such an approach allows one to decouple access control management from the application logic. However, even for simple scenarios, the maintenance of consistent access control policy sets is not a trivial task[1]. Therefore tools supporting the analysis of policies are crucial, especially for highly dynamic environments. In the last years, several approaches and tools for policy analysis have been proposed (Fisler et al., 2005; Rao et al., 2008; Kolovski et al., 2007). However, a common shortcoming of these tools is that the analyses they support are based on a string-based matching of the terms used in the policies. Such a simple approach makes the assumption that different strings represents different concepts (a.k.a. *Unified Name Assumption* - UNA). This assumption unfortunately does not always hold. When the analysis is performed on policies from different organizations or administrative domains, it is often the case that different terminologies are used and thus the same concept may be represented by different terms. In such a context, current policy analysis tools have limited applicability.

In this paper we address the problem of policy heterogeneity by proposing a stack of technologies that when applied to a multidomain policy set is able to generate a unified terminology that can be exploited by the analysis tools without requiring changes to these tools. The key feature of our approach is the adoption of ontologies for the specification of the vocabulary of policies. The advantage of using semantic schemas is in the powerful techniques that can be used for generating mappings between entities belonging to these schemas. However, we cannot assume that policies define their vocabulary according to an ontology and for this reason a comprehensive approach for ontology management in the specific context of policy analysis is needed. The goal of our work is to devise and implement such an approach. We cast our work in the context of the XACML standard policy

[1] In the remainder of the paper we use the term 'policy' as a shorthand for the term 'access control policy'.

Ferrini R. and Bertino E. (2009).
A COMPREHENSIVE APPROACH FOR SOLVING POLICY HETEROGENEITY.
In *Proceedings of the 11th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification*, pages 63-68
DOI: 10.5220/0001951500630068

language, because it is very well known general language. Our approach can however be almost directly applied to other attribute-based access control models.

The rest of the paper is organized as follows. in Section 2 we give background information about XACML, ontologies, and ontology mapping. In Section 3 we present an overview of our approach. Preliminary concepts are introduced in Section 4, whereas in Section 5 we describe the key functions underlying our approach. Section 6 reports implementation details and experimental results. Finally, Sections 7 and 8 discuss related work and conclusions, respectively.

2 BACKGROUND NOTIONS AND PRELIMINARY DEFINITIONS

In this section we introduce background notions and preliminary definitions that are used throughout the rest of the paper.

2.1 XACML

XACML (eXtensible Access Control Mark-up Language) (Moses, 2005) is the OASIS standard language for the specification of access control policies. It is an XML language able to express a large variety of policies, taking into account properties of subjects and protected objects as well as context information. In general, a subject can request an action to be executed on a resource and the policy decides whether to deny or allow the execution of that action. Several profiles, such as an RBAC profile, and a privacy profile, have been defined for XACML. An XACML policy consists of three major components, namely a `Target`, a `Rule set`, and a `Rule Combining Algorithm` for conflict resolution. The `Target` identifies the set of requests that the policy is applicable to. It contains attribute constraints characterizing subjects, resources, actions, and environments. Each Rule in turn consists of another optional `Target`, a `Condition`, and an `Effect` element. The rule Target has the same structure as the policy Target. The Condition specifies restrictions on the attribute values, provided as part of the request, that must hold in order for the request to be permitted or denied as specified by the Effect. The Effect specifies whether the requested actions should be allowed (`Permit`) or denied (`Deny`). The *Rule combining algorithm* is used to solve conflicts among applicable rules with different effects. In our context, we are interested in the user-defined values used in policies that from now on we refer to as *policy terms*. We thus assume that one can extract such information from XACML policies; we assume also that such information is represented in the form of $\langle attribute, value \rangle$ pairs.

2.2 Ontologies and Ontology Matching

An ontology typically provides the specification of a domain of interest in terms of classes, class instances (or individuals), and the relations according to which these classes are related. The actual W3C standard Ontology Web Language (OWL) is an XML-based language with a well defined semantics grounded in Description Logics (DL). In OWL relations are distinguished into *Object properties* and *Datatype properties*: Object properties relate instances of two classes, whereas Datatype properties relate class instances to some typed value. Given an ontology O_i, from now on we denote the set of the *entities* that belong to the ontology O_i as $E(O_i)$. Moreover, we use the terms *ontology*, *knowledge base*, and *vocabulary of a policy* as synonyms. *Ontology matching* is the process whereby two ontologies are related at the conceptual level. From now on we refer to a matching between the two ontologies O_i and O_j as π_{O_i,O_j}. State of the art ontology matching approaches are able to generate mappings that reduce to the same high level general form[2]. Given two ontologies O_i and O_j, a *mapping element* is a tuple $\langle e_{O_i}, e_{O_j}, s \rangle$, where $e_{O_i} \in E(O_i)$, $e_{O_j} \in E(O_j)$, and s is a confidence measure in some mathematical structure (typically in the [0, 1] interval). We discuss the specific Ontology Matching algorithm adopted in our approach in Section 4.1.

3 OVERVIEW OF THE APPROACH

The key idea in our approach is the definition of a process for the creation of a unified vocabulary with respect to which all policies from a multidomain policy set[3] can be specified. In our model, we adopt ontologies for the formalization of the policy vocabulary. Since an ontology provides richer information than a simple list of strings, it is well suited for representing the terminology of a policy. However, we cannot assume that all policies in a multidomain policy set adopt the same ontology for the specification of their vocabularies. The reason is that the development of an ontology is often a complex, time-consuming and

[2] Proposed in (Shvaiko and Euzenat, 2005).

[3] By multidomain policy set we refer to a set including policies from different domains or organizations.

error-prone task. Hence, it is usually a good practice to reuse, when possible, already defined ontologies instead of generating new ones. However, different ontologies may describe overlapping domains and these similarities need to be detected to avoid repetition of entities in the unified vocabulary we aim to create. Moreover, the definition of the vocabulary becomes even more complicated when a policy combines entities that belong to more than one ontology and plain strings. We can thus summarize the cases that may arise when dealing with multidomain policy sets as follows: (i) all policy terms are simple strings and no ontologies are used; (ii) all policy terms are associated with concepts in the same ontology; (iii) all policy terms are associated with concepts defined in more than one ontology; and (iv) some of the policy terms are associated with concepts defined in more than one ontology, while the remaining terms are simple strings. Our approach is able to deal with all these cases by combining together different ontology management techniques. From now on we refer to the terms associated with an ontology concepts as *semantic data*, while we refer to all other terms as *non-semantic data*. Figure 1 shows the architecture of our model. All the technologies involved in the ontology creation process are organized into a stack of functions. The dependencies between the various approaches (if any) are represented as arrows. The basic building block is the *Ontology Matching* process that is used by all the other methods. On top of such process we have the *Ontology Merging* and *Ontology Extraction* processes. These processes are not completely decoupled because, in the general case, the result of an ontology merging can be exploited during the extraction of an ontology from the non-semantic data of a policy. The topmost blocks represent the creation of the *Policy Reference Ontology* and the *Reference Ontology of the policy aset*.

4 DEFINITIONS AND FUNCTIONS

In this section we introduce definitions that are used in the rest of the paper and the key functions in our model.

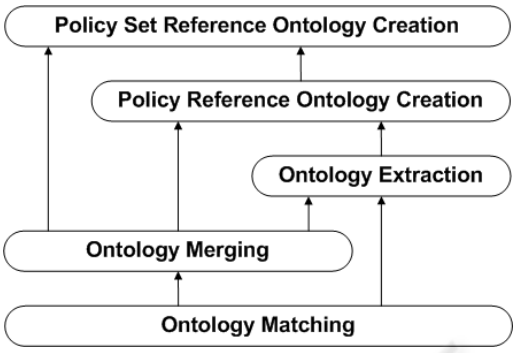

Figure 1: The stack of technologies addressing policy heterogeneity.

4.1 Ontology Matching

Ontology Matching is the basic function in our architecture. To implement it, we have adopted the Falcon-AO approach (Hu et al., 2008). The 2007 Ontology Alignment Evaluation Initiative (OAEI 07) results indicate Falcon to be the best performing ontology matcher available. However, several ontology matchers have been proposed, each with different strengths and weaknesses. Because we adopt a layered architecture, our model can be easily extended by adopting the ontology matching algorithm that is more suitable with specific scenarios. The only assumption we make on an ontology matching π_{O_i,O_j} between the ontologies O_i and O_j, is that if $\langle e_{O_i}, e_{O_j}, s\rangle \in \pi_{O_i,O_j}$, then $\nexists\ \langle e'_{O_i}, e'_{O_j}, s'\rangle \in \pi_{O_i,O_j}$ such that $e_{O_i} = e'_{O_i}$ or $e_{O_j} = e'_{O_j}$.

4.2 Ontology Extraction

In this section we address case (i) introduced in Section 3. When a policy does not use semantic data, it is necessary to create a new ontology extracting semantic knowledge by the information that can be deduced from the policy itself. The problem of extracting meaningful knowledge from unstructured data is usually referred to as *Ontology Extraction* or *Ontology Learning* and has been extensively investigated, especially after the introduction of the Semantic Web paradigm. In our context the data are XACML policies; thus we can exploit the explicit knowledge provided by the policy language to obtain a first classification of terms. In doing so, we adopt the mapping XACML to Description Logics proposed in (Kolovski et al., 2007). The key idea of such mapping is that each *attribute-value* pair in a XACML policy can be translated by adding two entities to the extracted knowledge base: given the pair $\langle attribute, value\rangle$, the

relation `attribute` and the concept `value` are added to the ontology. In our case we work with OWL and for this reason we need to check the data type of the value before translating the attribute into the correct OWL properties. In this paper we deal with the string datatypes and all the numeric data types. We plan to extend this mapping for managing more data types as part of our future work. In our mapping, if *value* is a string, then it is translated into a new concept and *attribute* become an object property. If *value* is a XML Schema numeric data type, no concepts are added and *attribute* becomes a data type property.

4.3 Ontology Merging

In the general case a policy may adopt more than one ontology for the specification of its vocabulary. For this reason, we have to define an approach for combining the set of exploited ontologies together in the unified vocabulary. The problem of combining two (or more) ontologies in a single knowledge base is usually referred to as *Ontology Merging*. The intuitive idea is that, given ontologies O_i and O_j, we aim at constructing the union of entities $e_i \in E(O_i)$ and $e_j \in EO_j$ such that if e_i and e_j can be considered the same, then just one of them is added to the resulting ontology. For example, we may consider equivalent the entities e_{O_i}, e_{O_j} belonging to the mapping element $\langle e_{O_i}, e_{O_j}, s\rangle$ if s is greater than a certain threshold τ. Thus, in building up the merged ontologies we can consider just one of the two entities.

Definition 1 (Merged Policy Ontology). *Let P_i be a policy. Its merged ontology, denoted as $\tilde{O}_{P_i}$, is the ontology recursively defined as:*

- $\tilde{O}^0 := \emptyset$
- $\tilde{O}^t := MERGE(\tilde{O}^{t-1}, O_t)$
- $\tilde{O}_{P_i} := MERGE(\tilde{O}^{|\sigma(P_i)|-1}, O_{|\sigma(P_i)|})$

where $\sigma(P_i)$ is a partially ordered set of the ontologies in P_i; $O_t \in \sigma(P_i)$ with $1 \leq t \leq |\ \sigma(P_i)\ |$; and $MERGE(O_i, O_j)$ is a function that takes in input the ontologies and adds the entities $e_{O_j} \in O_j$ to O_i such that $\nexists\ \langle e_{O_i}, e_{O_j}, s\rangle$ with $e_{O_i} \in O_i$ and s greater than an acceptance threshold τ.

4.4 Hybrid Scenarios

The more complicated scenario is when we have both heterogeneous domains and partial knowledge. However, we can easily manage such scenario by combining together the approaches defined in Definition 1 and in Section 4.2. We define the Policy Reference Ontology as follows:

Definition 2 (Policy Reference Ontology). *Let P_i be a policy. Its reference ontology, denoted by $\dot{O}_{P_i}$, is defined as follows:*

$$\dot{O}_{P_i} := MERGE(\tilde{O}_{P_i}, \ddot{O}_{P_i}).$$

It is important to notice that $\dot{O}_{P_i}$ is a general case of both $\tilde{O}_{P_i}$ and $\ddot{O}_{P_i}$. Such general definition can be applied to all of the cases introduced in Section 3. This is the reason why we refer to $\dot{O}_{P_i}$ as the Policy Reference Ontology for policy P_i.

5 CREATING THE UNIFIED VOCABULARY OF A POLICY SET

In this section we present the algorithms that we have developed based on the definitions introduced in Section 4.

5.1 Ontology Extraction

Our Ontology Extraction algorithm improves the mapping defined in Section 4.2 by refining the resulting ontology through a hierarchical organization of the new entities. The motivation for such refinement is that after the extraction of the entities based on the XACML-DL mapping, we obtain a simple list of new properties and concepts. However, the extracted ontology may be involved in some ontology matching processes. Since ontology matching takes advantage of both the entity names and their organization in the ontology, a knowledge base without a structure, that is, without a hierarchical organization of the entities, is useless for our purposes. For this reason, we combine the mapping defined in Section 4.2 with the subsumption relation between the terms that from are extracted from lexical databases, such as WordNet [4]. Details about the extraction of hierarchies are reported in (Ferrini and Bertino, 2009). The algorithm in Figure 2 gives the details of the Ontology Extraction process. For simplicity we only show the creation of object properties; the creation of data type properties is straightforward because we just need to add the new property without creating any concept. Lines 2-5 of the algorithm create a new property and a new concept given an attribute-value pair. Line 6 updates the range of the property with the new concept, finally line 7 returns the $\ddot{O}_{P_i}$ enriched with the hierarchies added by the `CREATE_HIERARCHY` function.

One may argue that our Ontology Extraction algorithm might be improved by taking into account the

[4] http://wordnet.princeton.edu/

knowledge that can be inferred by comparing different rules within a policy. For example, if subject s_i has more privileges than subject s_j, we may organize the associated ontology concepts in a specialized subject hierarchy. However, such additional information is a property of the policies and is not knowledge describing a given term. This can misleads the matching algorithm since the same concepts in different policies may be differently related. Moreover, this additional knowledge is typically taken into account during subsequent steps in the policy analysis process.

INPUT: $Pairs(P_i)$: The attribute-value pairs of P_i **OUTPUT:** $\ddot{O}_{P_i}$: The ontology extracted by the policy P_i

```
1: Ö_{P_i} = new Ontology() ;
2: FOR EACH ⟨ attribute_j, value_k ⟩
3:   IF Ö_{P_i}.not_contains(attribute_j)
4:     Ö_{P_i}.add(new(ObjectProp(attribute_j)));
5:   Ö_{P_i}.add(new(Concept( value_k )));
6:   Ö_{P_i}.attribute_j.range = Ö_{P_i}.value_k;
7: return CREATE_HIERARCHY(Ö_{P_i});
```

Figure 2: The Ontology Extraction Algorithm.

5.2 Ontology Merging

In our model, ontology merging plays a crucial role since is one of the core functions in the solution stack. We have developed two different algorithm to address the requirements concerning ontology merging (cfr.1): (i)

1. `MERGE`. It is the base Merge algorithm that takes in input two ontologies and returns the reconciled knowledge base. This algorithm (see Figure 3 implements the function *MERGE* defined in 1.
2. `ONTOLOGY_MERGING`. It is the function that applies `MERGE` to all the ontologies exploited by policy P_i. This algorithm implements all he components defined in 1. Because of space limitation the algorithm is not reported here.

5.3 Policy and Policy Set Reference Ontology

The creation of the Policy and Policy Set Reference Ontology is the final step in our approach. Such functions are straightforward and because of space limitation are not reported here. The Policy Reference Ontology is obtained by combining the Ontology Merging and the Ontology Extraction functions. With respect to the Policy Set Reference Ontology, the key idea is to extract the reference ontology for each policy in the multidomain set under consideration and merge all the resulting ontologies.

INPUT: O_i: The first ontology to be merged O_j: The second ontology to be merged **OUTPUT**: $\tilde{O}_{i,j}$: The merged ontology

```
1: Õ_{i,j} = new(Ontology());
2: mapping = MAP(O_i, O_j);
3: FOR EACH me_k ∈ mapping
4:   IF me_k.s > τ
5:     O_j.update(me_k.e_{O_j}, me_k.e_{O_i});
6:     Õ_{i,j}.add(me_k);
7: FOR EACH e_{O_i} ∉ Õ_{i,j}
8:   Õ_{i,j}.add(e_{O_i});
9: FOR EACH e_{O_j} ∉ Õ_{i,j}
10:
  Õ_{i,j}.add(e_{O_j});
11: return Õ_{i,j};
```

Figure 3: The MERGE Algorithm.

6 IMPLEMENTATION AND EXPERIMENTAL RESULTS

We have implemented a JAVA prototype of the proposed approach. The prototype uses the Sun implementation of XACML, the OWL API for loading, updating, and creating ontologies, the Falcon-AO library for ontology matching, and the MIT Java WordNet Interface for managing the WordNet database. For the experimental evaluation we generated a set of XACML policies. Attributes were randomly selected from a predefined list, while semantic data was obtained by randomly selecting entities from a set of ontologies retrieved by using the SWOOGLE ontology search engine. Figure 4 shows the total execution time of our process for increasing values in the number of attributes. We plotted the execution time of the approach for varying values in the number of total attributes[5] ranging from 10 to 50. Each column shows the time of: (i) the merge algorithm, (ii) the extraction algorithm, and (iii) the combination of their result. As expected, most of the execution time is spent in merging ontologies. Conversely, the extraction is very quick and even for a high number of attributes (not reported in the figure) e.g. ≈ 100, the execution time is ≈ 150 msec.

Table 1 reports data concerning the accuracy of our model. We evaluate the number of the detected

[5]Since in our approach we consider attribute-value pairs, it makes more sense to analyze the times with respect to the number of attributes instead of the policy number.

similar concepts and the correctness of the related mappings for varying values of the threshold τ from 0.65 to 0.95[6]. The results show that for values of τ between 0.75 and 0.80, our model achieves a good balance between the correctness of the mappings and the increase in the detected similarities. More experiments along with an optimized version of the ontology matching process are reported in (Ferrini and Bertino, 2009).

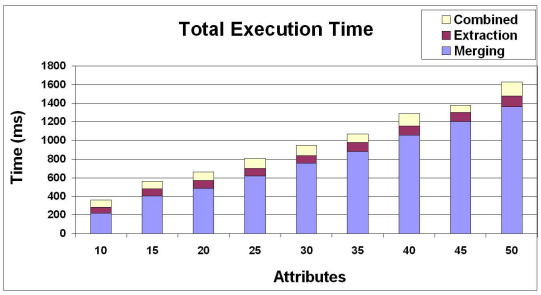

Figure 4: Total execution times for increasing values in the number of policy attributes.

Table 1: The accuracy of the model.

τ	Sim. Con. Detected	Correctness
[0.65, 0.70]	86,458%	58,823%
[0.70, 0.75]	85,416%	64,705%
[0.75, 0.80]	**80,208%**	**80,411%**
[0.80, 0.85]	58,333%	85,294%
[0.85, 0,90]	52,083%	91,176%
[0.90, 0.95]	46,875%	94,117%
> 0.95	42,708%	97,059%

7 RELATED WORK

To the best of our knowledge, this is the first approach that exploits ontology-based techniques for addressing policy heterogeneity. However, policy analysis and ontology-based technologies have already been investigated. Rein (Kagal et al., 2006) is a general policy framework based on semantic web technologies. Rein is able to support general purpose policy systems and for this reason it is well suited for solving mismatches among different policy languages. However, Rein does not address the problem of heterogeneity among vocabularies. Kolovski et al. (Kolovski et al., 2007) propose a mapping between XACML and Description Logics along with some interesting analysis services. However, they do not address the problem of policy heterogeneity. Finally, Lin et al. (Lin et al., 2007) propose a policy similarity function exploited as a filter before applying more accurate analysis tools.

[6]We run our prototype on a set of policies with an average number of 50 attributes.

8 CONCLUSIONS

In this paper, we have addressed the problem of heterogeneity in the context of policy analysis. Our approach represents the terminology of a policy through the use of ontologies and consists of a stack of functions that allows one to generate a unified vocabulary for a multidomain policy set. This vocabulary can be then exploited by policy analysis tools for analyzing and comparing policies. We have implemented a prototype of the proposed approach and analyzed its performance.

REFERENCES

Ferrini, R. and Bertino, E. (2009). A comprehensive approach for solving policy heterogeneity. Technical report, Purdue University, Department of Computer Science, CERIAS.

Fisler, K., Krishnamurthi, S., Meyerovich, L. A., and Tschantz, M. C. (2005). Verification and change-impact analysis of acces scontrol policies. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 196–205.

Hu, W., Qu, Y., and Cheng, G. (2008). Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering*, 67(1):140–160.

Kagal, L., Berners-Lee, T., Connolly, D., and Weitzner, D. (2006). Using semantic web technologies for policy management on the web. In *21st National Conference on Artificial Intelligence (AAAI)*.

Kolovski, V., Hendler, J., and Parsia, B. (2007). Analyzing web access control policies. In *Proceedings of the International World Wide Web Conference WWW 2007*, pages 677–686.

Lin, D., Rao, P., Bertino, E., and Lobo, J. (2007). An approach to evaluate policy similarity. In *SACMAT '07: Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*, pages 1–10, New York, NY, USA. ACM Press.

Moses, T. (2005). *Extensible access control markup language (XACML) version 2.0*. OASIS Standard.

Rao, P., Lin, D., Bertino, E., Li, N., and Lobo, J. (2008). Exam: An environment for access control policy analysis and management. In *POLICY*, pages 238–240.

Shvaiko, P. and Euzenat, J. (2005). A survey of schema-based matching approaches. *Journal on Data Semantics IV*, pages 146–171.