Using Abductive and Inductive Inference to Generate Policy Explanations

Fabio Marfia
DEIB Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milan, Italy

Keywords: Policy Explanation, Access Control, Policy Languages, Reasoning.

Abstract: Providing reliable explanations for the causes of an access response represents an important improvement of applications usability and effectiveness, in a context where users are permitted or denied access to resources. I present an approach composed by two different procedures, both relying on OWL-DL and SWRL Rules, in order to generate policy explanations. The first procedure makes use of OWL Explanation and abductive reasoning. The second uses an algorithm of Association Rule Learning to identifying attributes and states arising together with policy privileges, in an inductive way. The PosSecCo IT Policy language is used in the present paper for representing the policies, but the approach is general enough to be applied in other environments as well.

1 INTRODUCTION

Policies are a widespread approach for protecting users privacy and security, and for allowing or enforcing users to abide by different norms and laws. More and more complex and distributed scenarios are requesting access control solutions, in the last years, where a centralized framework is required to manage a large set of functionalities related to policies. Those functionalities include policy editing, storing, harmonization, and decision making.

The more such a framework grows in complexity, the more difficult is for a common user to realize what are the inner workings that result in specific policy-dependent behaviors, as, e.g., an access deny after a specific datum is requested. Also, even if the data consumer is not a user, but, e.g., a software, any chance to better understand specific outputs would be appreciated by developers of applications with adaptive functionalities. Such applications have grown more and more during the last years.

Providing an explanation together with a policy decision would represent a useful service in complex or distributed scenarios, even if not in every of them. In fact, there are different cases in which the disclosed information can be used to support illicit access to resources. Anyway, it is easy to identify many environments in which the chance of returning valid policy explanations would represent a significant improvement of the technology effectiveness and ease in its whole.

I present an approach in this paper where OWL-DL, SWRL Rules, OWL Explanation (Horridge et al., 2008) and Association Rules (Agrawal et al., 1993) are used in order to generate reliable explanations, either in formal or natural language, about the causes of an access permission or deny. The present work makes use of the PoSecCo policy management framework (Basile et al., 2012), but the approach is general enough to be applied to other policy languages as well, provided some constraints (see section 6).

The approach is based on two different logic procedures: the first is an exact method based on abductive inference, where OWL Explanation is used in order to identify the policies that are involved in a specific decision. The second is a probable method based on inductive inference, where it is tried to identify, between the theorems, the ones that characterize the identities that have access to specific resources.

While the first approach usually results in more reliable explanations, there is to notice that the number of policies involved in a decision, in complex environments, can be so high that the explanation can not be always read by a common user with ease. So, the second approach can be applied instead: attributes or facts that distinguish the identities that are able to access to specific resources can be identified. That gives to a policy explanation tool the ability to explain why an access permission or deny is returned without directly taking into account the policies involved.

Marfia F..
Using Abductive and Inductive Inference to Generate Policy Explanations.
DOI: 10.5220/0005116004570462
In *Proceedings of the 11th International Conference on Security and Cryptography* (SECRYPT-2014), pages 457-462
ISBN: 978-989-758-045-1

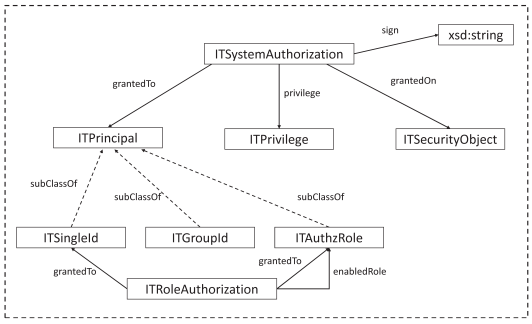

Figure 1: PoSecCo IT Policy classes related to authorizations and roles.

For instance, if every identity that can access to a resource R is also the subject of a theorem T, and a user U access request to R results in a deny, and U is not subject of T, a reason why U can not access to R could consist in the fact that he/she does not present the peculiarity T, or something factually linked to that. The identification of theorems as T is done with an algorithm of Association Rule Learning.

Probable explanation should be always considered *cum grano salis* by a user and it should be preferred to exact explanation only in environments where abductive inference is an unsuitable solution, according to the principle that a probable answer is better than no answer at all.

The paper proceeds as follows: the PoSecCo environment is presented in section 2, together with the PoSecCo IT Policy tool for representing policies. A procedure for inferring all the proper access privileges and states from a collection of axioms, policies and facts is described in section 3. The abductive generation of explanations is described in section 4, while the alternative inductive procedure is described in section 5. Applicability of the approach to other languages is discussed in section 6. Related work is presented in section 7, and future work in section 8.

2 POLICY REPRESENTATION

PoSecCo is a policy management environment that allows the design of rules for access control at different levels of abstraction. For the purposes of this paper, only the PoSecCo IT Policy level is taken into consideration (Neri et al., 2013). PoSecCo IT Policy allows the specification of Identity-Based Access Control (IBAC) and Role-Based Access Control (RBAC).

In such level of abstraction, a rule (ITSystemAuthorization) is a quadruple of the form $P = \langle \mu, \pi, \rho, \sigma \rangle$, where:

- μ is an identity (ITSingleId), a group of identities (ITGroupId) or a role (ITAuthzRole);
- π is a privilege (ITPrivilege);
- ρ is a resource or a group of resources (ITResourceObject);
- σ is the sign of the rule (positive or negative).

An identity can be assigned to one or more roles; furthermore, a role can be specified as a subset of another role, allowing to generate role hierarchies (ITRoleAuthorization). The cited PoSecCo IT Policy classes are represented in Figure 1.

3 INFERRING ACCESS PRIVILEGES

In order to generate reliable explanations for access responses, it is necessary to have an OWL-DL ontology Γ in which all the access privileges can be correctly inferred. In order to do so, the subsequent steps are foreseen:

1. Collection of all the necessary axioms in a set $\mathcal{T}$, as a TBOX ontology;
2. Policy design, and collection of the policies in an ABOX set $\mathcal{A}$;
3. Verbose descriptions, in the form of OWL annotations, are added to $\mathcal{A}$; they are necessary for generating natural language responses;
4. SWRL Rules, generated in order to infer all access privileges, are added to $\mathcal{A}$;
5. States of facts are added to $\mathcal{A}$;
6. The final ontology is obtained as $\Gamma = \mathcal{T} \cup \mathcal{A}$.

One step is described deeper in the the subsequent subsection: it is step 3 about verbose annotations.

3.1 Verbose Annotation

When the policy design is done, policies and attributes can be manually enriched with special annotations in order to generate the explanations in natural language, and as much clear as it is possible. In order to do this, the subsequent vocabulary of OWL annotations is defined:

- `IT_EXPL_rule_description`: it represents a verbose description of a security rule.
- `IT_EXPL_role_auth_description`: it represents a verbose description of a role assignment to another role or an identity.
- `IT_EXPL_identity_description`: it represents a verbose description of an identity.

- `IT_EXPL_role_description`: it represents a verbose description of a role.
- `IT_EXPL_attribute_description`: it represents a verbose description of an identity attribute.

4 ABDUCTIVE EXPLANATION

After the process described in section 3, the procedure of policy decision reduces itself to a task of checking whether Γ logically entails two specific statements. They are the two propositions explicitly asserting that the requesting user *can do* and *can not do* the requested action with the requested resource. I call them respectively α and $\neg\alpha$. The subsequent condition must occur for an access to be permitted:

$$\Gamma \models \alpha.$$

While the subsequent conditions must occur for an access to be denied:

$$\Gamma \models \neg\alpha \vee (\Gamma \not\models \neg\alpha \wedge \Gamma \not\models \alpha).$$

So, if the *can do* statement logically follows from Γ the access is permitted. If the *can not do* statement logically follows from Γ, or no logical consequence is found (no information available), the access is denied.

Three different cases can be identified then:

1. An access response is returned after a *can do* statement logically follows from the ontology ($\Gamma \models \alpha$);
2. A deny response is returned after a *can not do* statement logically follows from the ontology ($\Gamma \models \neg\alpha$);
3. A deny response is returned after no logical consequence is found ($\Gamma \not\models \neg\alpha \wedge \Gamma \not\models \alpha$).

According to which situation happens, a suitable procedure must be called in order to return a reliable explanation together with the access response. In fact, while cases 1 and 2 can be managed by the same procedure, case 3 must be necessarily considered for its own.

In any situation, the present approach makes use of OWL Explanation. It is a procedure that is able to generate one or more justifications for a theorem in an OWL-DL ontology. A justification is described as a minimal subset of the ontology that is sufficient for a specific inference to hold. It is easy to understand that if a policy or a state of facts is involved in a specific access permission or deny, that policy or state will necessarily appear in the subset of the ontology generated as a justification for that permission or deny. That principle is used in order to isolate and identify the policies that are to be returned in an explanation.

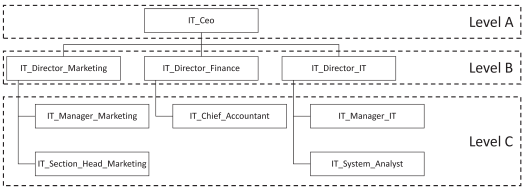

Figure 2: Sample company organogram.

In the two subsequent subsections I present the two cases separately in which the logical consequence of α or $\neg\alpha$ is obtained from Γ (section 4.1) or not (section 4.2).

4.1 Abductive Explanation from a Reference Statement

The first case taken into consideration is when the logical consequence of a *can do* statement is verified from Γ and an access response results in a permission, or the logical consequence of a *can not do* statement is verified from Γ and the response is a deny. I call such α or $\neg\alpha$ statement *reference statement*. In order to return an explanation, the subsequent passages are done:

1. OWL explanation is used to generating one or more justifications for the reference statement in Γ, every justification γ_i is put in a set $\mathcal{J}$;
2. For every $\gamma_i \in \mathcal{J}$ all the rules, role authorizations and state of facts listed in that justification are identified and put in a set $\mathcal{S}_i$ of propositions;
3. All $\mathcal{S}_i$ sets are examined: if two sets are equal, one is removed; if a set is a subset of another set, it is removed;
4. One explanation is generated for every remaining $\mathcal{S}_i$ set, if the explanations are more than one they are all returned in parallel.

Rules, role authorizations and state of facts can be presented as an explanation either in the form of an ontology or a set of natural language propositions. In the latter case, the verbose description of every single element has to be retrieved from its `IT_EXPL` annotation (see section 3.1).

4.1.1 Example

I define a fictitious company where every employee is assigned to a security role that can be of three different types: `Level C`, `Level B`, or `Level A`. Every level inherits every privilege from the underlying roles. So, `Level B` employees have every privilege that is released to `Level C` role, and a `Level A`

employee has every privilege that is released both to `Level B` and `Level C` roles. A sample organogram of such a company is shown in Figure 2.

Let us consider a policy that allows `Level B` employees to read the document `managementDocument`. `IT_CEO`, of `Level A` role, wants to read `managementDocument`. The answer is a permit, inheriting `Level A` all the `Level B` privileges. We want now to return to `IT_CEO` also the causes of such a response.

OWL Explanation identifies one justification with one rule and two role authorizations:

```
Rule: all Level B users can read
managementDocument;
Role Authorization: Level A users have
all privileges of Level B users;
Role Authorization: IT_CEO is a Level A
user.
```

The final explanation can be returned with the access response:

```
Access permitted for reading
managementDocument. Your access is
granted as a consequence of the
subsequent conditions and policies:

- all Level B users can read
managementDocument;
- Level A users have all privileges of
Level B users;
- You are a Level A user.
```

4.2 Abductive Explanation with no Reference Statement

An access deny can occur also when no statement that explicitly asserts an access permission or deny is found as a logical consequence of Γ. That implies that there is no information in Γ about the user rights to access to the requested resource, and the access can not be granted. In that case, there is no theorem on which apply abductive inference. Anyway, we can however generate an explanation, relying on the consideration that explaining why a user can not access to a resource can be done showing him/her in what he/she differ from who can access to the same resource. So, even if nothing can be said about the requesting user, we can look at other users that have access to the resource and identify what are the differences of role or attributes between them and the requesting user. So, an explanation can be generated with the subsequent passages:

1. Identification of the statements that assert that any subject *can do* the requested action with the requested resource, between all the privileges that can be inferred from Γ; every such α_k statement is put in a set $\mathcal{L}$;
2. OWL explanation is used to generating one or more justifications for every $\alpha_k \in \mathcal{L}$, every generated justification γ_i is put in a set $\mathcal{J}$;
3. For every $\gamma_i \in \mathcal{J}$ all the rules, role authorizations and state of facts listed in that justification are identified and put in a set $\mathcal{S}_i$ of propositions;
4. All $\mathcal{S}_i$ sets are examined: if two sets are equal, one is removed; if a set is a subset of another set, it is removed;
5. One explanation is generated for every remaining $\mathcal{S}_i$ set, if the explanations are more than one they are all returned in parallel;
6. For every explanation, information about the attributes of the user that are involved in the propositions found is added, in order to make the user able to compare it against the provided status.

Again, rules, role authorizations and state of facts can be presented as an explanation either in the form of an ontology or a set of natural language propositions. In the latter case, the description of every single element has to be retrieved from its `IT_EXPL` annotation (see section 3.1).

It is described, in step 6, that the explanation must be enriched with information about the attributes of the user that are involved in the propositions found, in order to make the user able to compare them against the provided status. For example, if at least one role authorization appears in the explanation, the user has to receive its own roles as an additional information; if an attribute appears in the explanation, the user has to receive its value for that attribute, or the information that the attribute was not found, if that is the case. Again, verbose explanations require the presence of `IT_EXPL` annotations for roles and attributes to be returned.

5 INDUCTIVE EXPLANATION

Abductive explanation can generate usable explanations in many use cases. Anyway, it can be understood that the presence of a considerable number of policies and states of facts can easily make such types of explanation unreadable for a common user.

Given that, I propose a different procedure, that generates probable, but more user-friendly and policy-independent explanations. The method consists in looking for the attributes and roles that appear more frequently, between all the theorems inferred

from Γ, together with the access conditions requested. Those attributes and roles can be presented as probably necessary conditions for the access to be granted. The user can use them to understand the causes of an access response.

More formally, given an access condition α, we have to look for association rules of the form $(\pi_1 \wedge \pi_2 \wedge \ldots \pi_k) \Rightarrow \alpha$ between all the theorems inferred from Γ, where π_k is a proposition in Γ that express an attribute, a role, a state of facts about an identity, a group or an identity template (ITPrincipal).

In order to do that, the Apriori algorithm of Association Rule Learning (Agrawal and Srikant, 1994) is used for looking for propositions in the ontology that appear frequently with α. The algorithm starts with an empty set $\mathcal{S}$. Then:

1. For every pair of propositions $\sigma_i = \{\pi_i, \alpha\}$, the number of ITPrincipal that are subject in the ontology of both the propositions is counted. Every pair with a number of subjects higher or equal to a subject count threshold δ_s is added to $\mathcal{S}$.
2. Frequent triples are identified, taking into consideration every $\tau_{i,j} = \{\pi_i, \pi_j, \alpha\}$ set, given that $\exists \sigma_i \in \mathcal{S} : \pi_i \in \sigma_i \wedge \exists \sigma_j \in \mathcal{S} : \pi_j \in \sigma_j$. For every triple $\tau_{i,j} \notin \mathcal{S}$, the number of ITPrincipal that are subject in the ontology of every proposition is counted. Every triple with a subject count higher than δ_s is added to $\mathcal{S}$.
3. Frequent quadruples are identified, taking into consideration every proposition that appear at least in a $\tau_{i,j} \in \mathcal{S}$ triple, and so on... the algorithm stops when no set is added to $\mathcal{S}$ in a step.

From every set $\{\pi_1, \pi_2, \ldots \pi_k, \alpha\} \in S$ the association rule $(\pi_1 \wedge \pi_2 \wedge \ldots \pi_k) \Rightarrow \alpha$ is identified. The confidence of every association rule is computed, and every association rule with a confidence higher than a threshold δ_c is kept and used to generate the explanation.

Confidence of an association rule is computed as:

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \tag{1}$$

where $supp(X)$ is the support of the proposition X, that is, in our case, the number of ITPrincipal that are subject of such proposition, divided by the number of all ITPrincipal in the ontology.

Every identified association rule represents a different explanation, in which are expressed the conditions $\pi_1, \pi_2, \ldots \pi_k$ that are, with a certain measured confidence, necessary for the presence of α. $\pi_1, \pi_2, \ldots \pi_k$ can be expressed in an explanation, eventually in a verbose description identifying the corresponding `IT_EXPL` annotations, as presented in section 3.1.

5.1 Example

I consider again the company in Figure 2, and a policy that allows `Level B` employees to read the document `managementDocument`. `IT_Manager_Marketing`, of `Level C` role, asks for the possibility to read `managementDocument`. The answer is a deny, being `IT_Manager_Marketing` assigned to a level lower than `Level B`. In that case, proposition α is:

```
α = IT_canRead(USER,
        managementDocument)
```

We want to generate a probable explanation, with δ_s threshold set as $\delta_s = 3$. The algorithm identifies the co-occurrence of α with the proposition β with count 3 in the ontology, where β is:

```
β = is_a(USER, Level_B)
```

The algorithm adds β to $\mathcal{S}$. No other co-occurrence count is higher or equal to δ_s.

Confidence of the association rule $\beta \Rightarrow \alpha$ is computed as:

$$conf(\beta \Rightarrow \alpha) = \frac{3/9}{3/9} = 100\% \tag{2}$$

Confidence is 100%, so it is higher or equal to every δ_c that can be chosen. β can be returned then as an explanation, presented as a probable necessary peculiarity to be able to read `managementDocument`:

```
Access denied while trying to read
managementDocument. Level B role
co-occurs with a 100% confidence with
such privilege, and may be a necessary
condition. Your role is Level C.
```

6 APPLICABILITY OF THE APPROACH TO OTHER POLICY LANGUAGES

Many policy languages are present in the access control scenario, with different expressiveness and functionalities (Coi and Olmedilla, 2008). Every policy language relying on Description Logics with a syntax no more expressive than $\mathcal{SHOIN}(\mathcal{D})$ logic, is, in principle, a language on which the present approach can be applied, supporting a subset or the whole of its functionalities.

There is to add that, being involved a reasoning procedure, one must ensure that all the data types (e.g. numbers, dates, geo-locations) in the ontology are correctly managed by an OWL reasoner in order to generate complete inferences.

7 RELATED WORK

Bonatti, Olmedilla and Peer (Bonatti et al., 2006) analyzed the possibility of generating policy explanations, with the Protune policy language, using abductive logic procedures. They do not rely on standard abductive tools, explaining that "There is no support for explaining infinite failed derivations" (p. 1). The problem is resolved in the present work with the different approach of reasoning on the identities that are able to access to specific resources, and then explaining in what the current user is different, as presented in section 4.2. Also, they do not deal with the problem of non user-friendly explanations, while the present work proposes the different approach of inductive reasoning. Furthermore, their approach is not able to manage specific data types and data constraint, as, e.g., expressing that a user must have a minimum age to access to a resource.

KNOW System (Kapadia et al., 2004) is a method for providing feedback to users who are denied access to resources, using Ordered Binary Decision Diagrams. The feedback returned consists in a set of policy changes, that are sufficient and necessary to obtain a permission. No verbose explanation is foreseen; also, they make use of no reasoning procedure.

8 FUTURE WORK

Computational resources required for OWL reasoning are usually consistent for large ontologies. Needing the present work to eventually generate all the possible privileges and theorems for even large knowledge bases, the performances in a real environment are to be evaluated, and ways to lower the computational load are eventually to be identified. The usage of SPARQL-DL query language (Sirin and Parsia, 2007), allowing queries for inferred knowledge, represents a possible way.

Moreover, the approach must be tested in real environments also to evaluating interfaces usability and user satisfaction.

Also, the possibility to expand further the expressivity of the used policy language, starting from PoSecCo IT Policy, has to be studied. For example, as presented in section 6, some languages express obligations to be fulfilled by the user. Other advancements must concern general environment states, as, e.g., generation of policy decisions as a consequence of current date and time.

Finally, applicability to other policy languages (see section 6) has to be studied in deep, especially for the most widespread policy languages, as XACML.

REFERENCES

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Basile, C., Lioy, A., and Paraboschi, S. (2012). The posecco security decision support system. In Reimer, H., Pohlmann, N., and Schneider, W., editors, *ISSE 2012 Securing Electronic Business Processes*, pages 64–74. Springer Fachmedien Wiesbaden.

Bonatti, P. A., Olmedilla, D., and Peer, J. (2006). Advanced policy explanations on the web. In *Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29 – September 1, 2006, Riva Del Garda, Italy*, pages 200–204, Amsterdam, The Netherlands, The Netherlands. IOS Press.

Coi, J. L. D. and Olmedilla, D. (2008). A review of trust management, security and privacy policy languages. In *International Conference on Security and Cryptography (SECRYPT 2008)*. INSTICC Press.

Horridge, M., Parsia, B., and Sattler, U. (2008). Laconic and precise justifications in owl. In *Proceedings of the 7th International Conference on The Semantic Web*, ISWC '08, pages 323–338, Berlin, Heidelberg. Springer-Verlag.

Kapadia, A., Sampemane, G., and Campbell, R. H. (2004). Know why your access was denied: Regulating feedback for usable security. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 52–61, New York, NY, USA. ACM.

Neri, Mutti, Psaila, Salvaneschi, Verdicchio, and Basile (2013). *D2.5 - IT Policy Meta-Model and Language*. PoSecCo WP2, Business and IT level policies.

Sirin, E. and Parsia, B. (2007). Sparql-dl: Sparql query for owl-dl. In *In 3rd OWL Experiences and Directions Workshop (OWLED-2007)*.