An Ontological Framework for Determining the Repercussions of Retirement Actions Targeted at Complex Access Control Policies in Cloud Environments

Simeon Veloudis
South East European Research Centre (SEERC)
The University of Sheffield, International Faculty CITY College
Thessaloniki, Greece 54622
sveloudis@seerc.org

Iraklis Paraskakis
South East European Research Centre (SEERC)
The University of Sheffield, International Faculty CITY College
Thessaloniki, Greece 54622
iparaskakis@seerc.org

Christos Petsos
South East European Research Centre (SEERC)
The University of Sheffield, International Faculty CITY College
Thessaloniki, Greece 54622
chpetsos@seerc.org

ABSTRACT

By migrating their data and operations to the cloud, enterprises are able to gain significant benefits in terms of cost savings, increased availability, agility and productivity. Yet, the shared and on-demand nature of the cloud paradigm introduces a new breed of security threats that generally deter stakeholders from relinquishing control of their critical assets to third-party cloud providers. One way to thwart these threats is to instill suitable access control policies into cloud services that protect these assets. Nevertheless, the dynamic nature of cloud environments calls for policies that are able to incorporate a potentially complex body of contextual knowledge. This complexity is further amplified by the interplay that inevitably occurs between the different policies, as well as by the dynamically-evolving nature of an organisation's business and security needs. We argue that one way to tame this complexity is to devise a generic framework that facilitates the governance of policies. This paper presents a particular aspect of such a framework, namely an approach to determining the repercussions that policy retirement actions have on the overall protection of critical assets in the cloud.

CCS CONCEPTS

• **Information systems** → **Ontologies**; *Secure online transactions*; • **Security and privacy** → **Software and application security**; • **Applied computing** → *IT governance*; • **Computer systems organization** → **Cloud computing**;

KEYWORDS

Policies; security; ontologies; cloud computing; governance; OWL 2

UCC'17 Companion, December 5–8, 2017, Austin, TX, USA.

ISBN 978-1-4503-5195-9/17/12…$15.00
DOI: https://doi.org/10.1145/3147234.3148114

1 INTRODUCTION

It is generally conceded that, by migrating their data and operations to the cloud, enterprises and organisations are able to gain significant benefits in terms of cost savings, increased data availability, greater business agility and productivity [2]. Nevertheless, despite the benefits, the shared and on-demand nature of the cloud paradigm introduces a new breed of security threats that generally deter stakeholders from relinquishing control of critical corporate assets to third-party cloud providers [2, 3].

In order to thwart these threats, adequate *access control policies* that convey an organisation's business logic and overall stance towards security, must be devised and infused into the applications through which critical assets are accessed in the cloud [14]. Nevertheless, the inherently dynamic and heterogeneous nature of cloud environments calls for policies that are able to incorporate a potentially *complex* body of *contextual knowledge* [15]. As an example, consider a policy that incorporates the following rules for specifying the contextual conditions under which an entity s is granted, or denied, write access to a sensitive data object o: s is permitted to write to o when s resides in the geographical area identified as $bldg_X$; s is denied to write to o when s resides in the area identified as $bldg_Y$, or in the network location identified as 144.0.0.0/8; s is denied to write to o when the request is issued during a prescribed time interval. This complexity is further amplified by the *interplay* that inevitably occurs between the different policies, and the rules thereof, that an organisation collectively employs in order to protect its sensitive assets—an interplay that potentially affects the *effectiveness* of the policies by influencing the manner in which they permit, or deny, access requests. The situation is further perplexed by the dynamically-evolving nature of an organisation's business and security needs that often necessitates changes in the policies, such as the introduction of new policies and/or the retirement of existing ones.

We argue that, in order to tame this complexity, hence increase assurance on the *effectiveness* of the policies, a generic framework that is capable of facilitating the overall *governance* of policies is required. Our work, conducted as part of the PaaSword project [1], offers such a framework. In particular, it aspires to provide a governance mechanism that is underpinned by an *ontological representation* of policies that semantically describes the various *knowledge artefacts* that are incorporated in the policies. In this respect, it promotes a clear separation of concerns by disentangling

the definition of policies from the code of the applications into which they are infused. This enables the provision of the following seminal capabilities in a *generic* and *automated* manner: (i) reasoning about the *correctness* of the policies, by constraining the knowledge artefacts that must, may, or may not be embodied in the policies; (ii) determining potential *inter-policy relations*, such as subsumption and contradiction; (iii) determining the repercussions of *policy lifecycle* actions, such as policy updates and retirements, on the ability of the policies to protect critical assets.

This paper focuses on the third capability. More specifically, it proposes an XACML-based [9] approach that is capable of determining, in an automated manner, how *retirement actions* targeted at the *rules* comprised by access control policies, influence the manner in which such policies permit, or deny, access to sensitive assets. To achieve this, our approach *semantically captures* the interplay that occurs between the rules of a policy for arriving at a decision for a particular access request; on that basis, it then proposes a *semantic characterisation* of all those requests that are affected by the retirement of a rule, i.e. it *ontologically* describes—in OWL [18]—the *knowledge artefacts* that characterise these requests. Evidently, such a semantic characterisation increases awareness among stakeholders about the repercussions of rule retirement actions.

One of the main benefits of our approach is that it enables—by virtue of *semantic inferencing*—reasoning about the effects of rule retirement even in cases in which the knowledge artefacts conveyed by a request do not match, at the syntactic level, the corresponding knowledge artefacts that characterise the affected requests. Assume, for instance, that a particular retirement action is determined to affect all requests that are issued from within the location identified as $bldg_X$; semantic inferencing allows us to conclude that any request originating from, say, a particular room of $bldg_X$, is also affected.

The rest of this paper is structured as follows. Section 2 proposes an ontological representation of our XACML-based model for policies; it also proposes an ontological representation of access requests. Section 3 specifies the conditions under which a rule comprised by a policy is enforceable upon an access request. Section 4 investigates the repercussions that the retirement of such a rule has by providing a semantic characterisation of all those access requests that are affected by the retirement. Section 5 presents a simple example that demonstrates the applicability of our approach. Finally, Section 6 presents related work and Section 7 outlines conclusions and future work.

2 ACCESS CONTROL POLICIES AND REQUESTS

Attribute-based Access Control (*ABAC*) policies [5], due to their inherent reliance on the generic concept of an *attribute*, are particularly suitable for capturing the contextual knowledge that characterises access requests [14]. This section presents a declarative model for the representation of ABAC policies—a model that forms the basis of our approach for determining the repercussions of retirement actions. More specifically, Sections 2.1 and 2.2 present, respectively, an informal account and an ontological representation of the model, and Section 2.3 outlines an ontological expression for access requests.

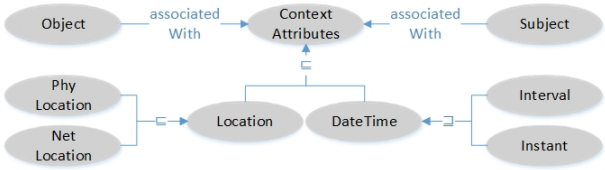

Figure 1: Simplified view of the CM

2.1 ABAC Policy Model

According to the XACML standard [9], an ABAC policy consists of one or more ABAC *rules* that convey its core logic. Each rule comprises an *antecedent* and a *consequent*. The latter captures the *decision* with which the rule responds to an access request which invariably resolves to either a '*permit*' or a '*deny*'. The former articulates, the rule's '*target*', i.e. a (*pre-*)*condition* that must be satisfied in order for the rule to be *enforceable* upon an access request. In particular, it incorporates a set of knowledge artefacts—its so-called *attributes*—whose values need to be taken into account when deciding whether to permit, or deny, a request. These attributes are drawn from an underlying *Context Model* (*CM*)—an extensible ontological framework that includes interrelated concepts suitable for capturing attributes and the properties thereof. A simplified view of the CM that is used in this work, one which includes only concepts and properties considered in this paper, is depicted in Fig. 1 (for more details the interested reader is referred to [16, 17]).

An ABAC policy is also associated with a *rule-combining algorithm* [9] that determines which of its rules (if any) is to be enforced upon an access request. For each access request, an ABAC policy resolves to at most one of its rules; a policy that does not resolve to any of its rules for a particular request is considered '*Not Applicable*' for that request. ABAC policies may also be bundled into ABAC *policy sets* which may recursively include other policy sets as well. Each ABAC policy set is associated with a *policy-combining algorithm* [9] that determines which of its elements—either a policy or a nested policy set—is to be enforced upon an access request. For each access request, an ABAC policy set resolves to at most one of its elements; a policy set that does not resolve to any of its elements for a particular request is considered 'Not Applicable' for that request.

2.2 Ontological Representation of ABAC Policies

Our ontological representation of the ABAC policy model is depicted in Fig. 2. At the core of this representation lie the concepts: *Policy*, *Rule*, *PolicySet* and *CombiningAlg*; as we would expect, instances of these concepts represent, respectively, policies, rules, policy sets and combining algorithms (both rule- and policy-combining ones). Policies and policy sets are associated with their corresponding combining algorithms through the functional property *hasCombAlg*. Similarly, rules are associated with their antecedents and consequents through the functional properties *hasAnt* and *hasCons* respectively; the consequent of a rule invariably resolves to exactly one of the individuals[1] *permit* or *deny*.

[1]The terms 'instance' and 'individual' are used interchangeably.

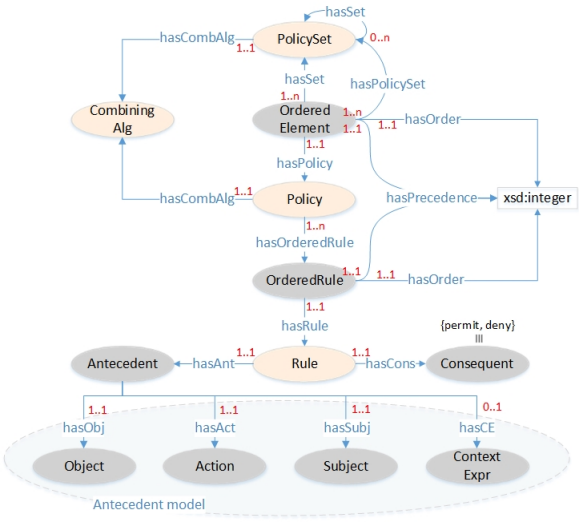

Figure 2: ABAC model

In addition, our ontological representation also includes the concepts *OrderedRule* and *OrderedElement* (see Fig. 2). These capture the *order* that is imposed upon the rules of a policy, or upon the elements of a policy set; they also capture the *precedence* with which these rules, or elements, are enforced upon an access request. More specifically, each instance of the concept *OrderedRule* is associated, via the property *hasRule*, with *exactly one* rule; similarly, each instance of the concept *OrderedElement* is associated, via one of the propertes *hasPolicy* or *hasPolicySet*, with *exactly one* element of a policy set—i.e. with either a particular policy or with a particular (nested) policy set. In addition, each instance of the concept *OrderedRule* is associated, via each of the properties *hasOrder* and *hasPrecedence*, with *exactly one* non-negative integer that represents that rule's order and precedence respectively in a containing policy. The same properties are used for assigning order and precedence to the elements of a policy set. ABAC policies are associated with their constituent ordered rules through the property *hasOrderedRule* (see Fig. 2), and ordered ABAC policies, or ordered (nested) ABAC policy sets, are associated with their encompassing policy set(s) through the property *hasSet*. The ontological model outlined above is formally articulated in terms of *terminological* (TBox) and *assertional* (ABox) axioms expressed in the $\mathcal{SROIQ}$ Description Logic (DL) [4]; the interested reader is referred to [12] for a relevant account.

Now the antecedent of an ABAC rule, as already mentioned, comprises *attributes* whose values need to be taken into account when deciding whether to permit, or deny, an access request. Below we outline a set of restrictions regarding these attributes. The first restriction demands that the antecedent of an ABAC rule invariably includes *exactly one* protected asset; formally, this is expressed by articulating that each instance of the concept *Antecedent* is associated, through the object property *hasObj*, with exactly one instance of the class *Object* of the CM (see Fig. 2). The second restriction demands that the antecedent of an ABAC rule invariably features *exactly one* association, via the property *hasAct*, with an action from the class *Action*, i.e. with an action that is to be performed on the protected asset[2]. The third restriction demands that the antecedent of an ABAC rule invariably features *exactly one* association, via the property *hasSubj*, with at least one entity requesting access to the protected asset, i.e. with at least one instance of the CM class *Subject*. Finally, the fourth restriction requires that the antecedent of an ABAC rule refers, through the property *hasCE*, to at most one *context expression*—i.e. to at most one expression that constrains the values of the *contextual attributes* that pertain to an access request. Context expressions are ontologically represented as instances of the class *ContextExpr* and are further elaborated below.

A context expression (CE) is a propositional logic expression that articulates the *contextual conditions* that must be satisfied in order for a rule to be *enforceable* upon a request. These conditions may refer to the subject and/or object of a request, or to the request itself. The various *attributes* that are bound by a CE, i.e. its *parameters*, take the form of instances of one or more concepts that are included in the *ContextAttributes* concept of the CM (see Fig. 1), and may be combined through the usual propositional logic connectives. A CE is associated with its parameters through the object property *hasParam*, whilst it may be defined recursively, in terms of one or more other CEs. A CE is associated with the entity that it refers to through the object property $refersTo$. As an example, consider a CE identified by the individual e that articulates that the subject s resides either in the physical location identified as $bldg_Y$, or in the network location identified by 144.0.0.0/8. Formally, such a CE is expressed by asserting that e is an instance of the abstract OWL class that comprises all those individuals that have some association with the individual $bldg_Y$, or with the individual 144.0.0.0/8, through the property *hasParam*, and also some association through the property $refersTo$ with the individual s; assertion 1 of Appendix A provides a formal representation (in $\mathcal{SROIQ}$) of such a CE.

2.3 Ontological Representation of Access Requests

Closely akin to the antecedent of an ABAC rule, an access request may include attributes that specify: the protected asset that it targets (i.e. its 'object'); the action that it aspires to carry out on that object; the entity that issues the request (i.e. its 'subject'); the CE that articulates the contextual circumstances under which the request is issued. Ontologically, an access request is expressed as an instance of a concept named *Request*, whilst it is associated with its attributes through the same properties used for interrelating ABAC rule antecedents with their attributes, namely: *hasObj*, *hasAct*, *hasSubj* and *hasCE*. Note that, in contrast to the case of ABAC rule antecedents, we avoid articulating here any restrictions concerning the attributes of an access request. The reason for this is that access requests, as opposed to ABAC rules, are externally-generated artefacts whose structure cannot be prescribed.

3 DETERMINING ENFORCEABILITY OF ABAC RULES

Reasoning about the enforceability of an ABAC rule upon an access request is an integral part of our approach for determining the

[2]The class $Action$ also forms part of the CM too but it is not depicted in Fig. 1.

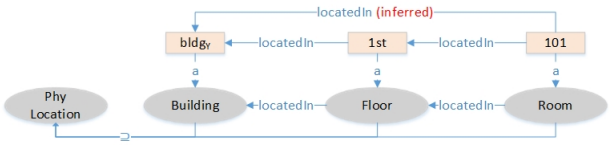

Figure 3: Inferencing at the CM

repercussions of policy retirement actions. This section outlines how such reasoning is performed.

An ABAC rule r is deemed *enforceable* upon an access request q iff q *entails* r's antecedent, i.e. iff r's antecedent is *semantically inferable* from q. As an example, let r's antecedent feature exactly one association, through the properties *hasObj*, *hasAct*, *hasSubj* and *hasCE*, with the individuals o, w, s and e respectively, where e represents the CE of assertion 1. Also, let q be associated, through the same properties, with the individuals o, w, s and e' respectively. e' represents a CE that expresses the fact that s resides in, say, room 101 of $bldg_Y$. r is enforceable on q as: (i) q is associated with the individuals o, w, s as demanded by r; (ii) e' entails e—an entailment that is based upon semantic inferencing that generates the knowledge that room 101 indeed resides in $bldg_Y$. It follows that, reasoning about the enforceability of a rule upon an access request may entail *semantic inferencing* in the CM. Nevertheless, for such inferencing to be possible, certain knowledge artefacts must be embodied in the CM. For instance, in the example above, the concept *PhyLocation* needs to include such concepts as *Building*, *Floor* and *Room*, along with their appropriate instances (see Fig. 3); it also needs to include the transitive property *isLocatedIn* that interrelates these instances. It is to be noted here that these artefacts are encoded into the CM during a priming process that is undertaken by stakeholders for customising the CM for their particular purposes.

We outline now our approach to reasoning about the enforceability of a rule r upon an access request q. Initially, we programmatically construct an *abstract* OWL class that comprises all those individuals that could, potentially, play the role of r's antecedent. This class, say A_r, includes all those individuals that are related, through the appropriate properties, with certain *attribute instances* from the CM. For instance, assertion 2 of Appendix B provides a formal representation of A_r for the example rule r outlined in the previous paragraph. Subsequently, the same process is followed in order to construct an abstract class, say R_q, that groups together all those individuals that may potentially play the role of the request q. The rule r is enforceable upon q iff R_q is a subclass of A_r, i.e. iff $R_q \sqsubseteq A_r$. The validity of this relation is assessed through the use of the Pellet reasoner [10].

4 DETERMINING THE REPERCUSSIONS OF RULE RETIREMENT ACTIONS

We are now ready to present our approach to determining the repercussions of policy retirement actions. Two kinds of retirement actions are discerned: those that are targeted to the rules of an ABAC policy, and those that are targeted to the elements (either policies or nested policy sets) of an ABAC policy set. Although our approach is capable of determining the repercussion of both kinds of retirement actions, for reasons of space in this paper we exclusively focus on the former kind of actions.

Let p be an ABAC policy and S_p its corresponding ruleset. In order to determine the repercussions that result from the removal of a rule from S_p, we first need to determine how p actually resolves to one of its rules when responding to an access request. This requires consideration of the relevant *precedence* that the rules in S_p enjoy. For any two rules $r_1, r_2 \in S_p$, we say that r_2 has a precedence greater or equal than the precedence of r_1, denoted $r_1 \preceq_p r_2$, iff the precedence value associated with r_1 in p (see Section 2.2) is less or equal than the precedence value associated with r_2 in p. Precedence values are assigned to rules on the basis of the *rule-combining algorithm* associated with p; a relevant outline is in order.

4.1 Rule-Combining Algorithms

In [9], the following rule-combining algorithms are reported: '*deny overrides*' (DO), '*permit overrides*' (PO)[3] and '*first applicable*' (FA). The DO algorithm imposes the following precedence on S_p: (i) all 'deny' rules (if any) are of equal precedence; (ii) all 'permit' rules (if any) are of equal precedence; (iii) each 'deny' rule takes precedence over each 'permit' rule. An entirely symmetrical precedence is imposed on S_p by the PO algorithm. The case of the FA algorithm is quite different as it imposes a precedence that is detached from the decisions entailed by the rules: instead, it coincides with the *order* imposed on S_p by p's creator through the *hasOrder* property (see Section 2.2).

4.2 Access Control Decisions at the Policy Level

We turn now our attention to the conditions that must be satisfied in order for p to yield the decision $v ::= permit|deny$ in response to a request q. First of all, there must exist at least one rule in S_p which is associated with the decision v; formally, the set $S_{p,v} = \{r \in S_p | C_r \equiv \{v\}\}$ must be non-empty, where C_r is the class that represents r's decision. Secondly, at least one of the rules in $S_{p,v}$ must be *enforceable* on q, i.e. the abstract class R_q must be a subclass of at least one of the abstract classes A_r where $r \in S_{p,v}$; formally: $R_q \sqsubseteq \bigsqcup_{r \in S_{p,v}} A_r$. Thirdly, there must exist at least one enforceable rule r' in $S_{p,v}$ such that *none* of the rules that have a precedence greater or equal than that of r, and which yield a decision opposite than r's, is enforceable on q; formally, the set $H_{p,v,r} = \{r' \in S_p | C_r \equiv \{\bar{v}\} \wedge r \preceq_p r'\}$, where $\bar{v} ::= permit|deny$ and $\bar{v} \neq v$, must comprise rules that are *not* enforceable on q as otherwise one of these rules would be enforced on q (instead of r) yielding a decision different than v. It follows that the class of all requests for which p yields the decision v, denoted $R_{p,v}$, takes the form of assertion 3 provided in Appendix D.1.

4.3 Removing Rules

We are now ready to determine the repercussions resulting from the removal of a rule r from S_p. In particular, we are interested in semantically characterising all those requests that are responded to with a certain decision $v ::= permit|deny$, as long as r remains in S_p, and which are considered 'Not Applicable', or are responded to with the opposite decision $\bar{v}$, when r is removed from S_p.

[3] In [9], *ordered* versions of these algorithms are also reported – refer to Appendix C for a justification regarding why these ordered versions do not concern us here.

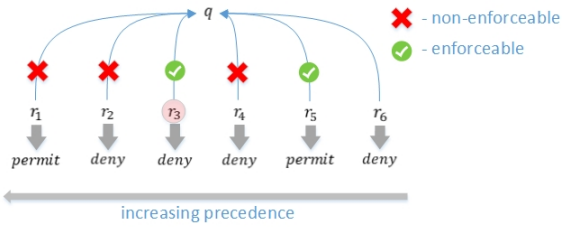

Figure 4: A v-inducing request turned into a $\bar{v}$-inducing one

4.3.1 v-Inducing Requests Turned Into 'Not Applicable'-Inducing Requests. The class, call it $R^{NA}_{p,r}$, of requests that induce the response v, as long as r remains in S_p, and the response 'Not Applicable', when r is removed from S_p, comprises all those requests upon which r, but no other rule from S_p, is enforceable; assertion 4 of Appendix D.1 provides a formal expression for $R^{NA}_{p,r}$.

4.3.2 v-Inducing Requests Turned Into $\bar{v}$-Inducing Requests. Let S_p comprise the rules depicted in Fig. 4 and let r_3 be the rule under removal. For a request q to induce the response *deny* prior to the removal of r_3, and the response *permit* after the removal of r_3, the following conditions must conjunctively hold. Firstly, r_3 must be enforceable on q. Secondly, *no* rule with a precedence greater than that of r_3 must be enforceable on q. This is because, if one or more such rule is indeed enforceable on q, then one of them will be ultimately enforced on q and thus p will yield the same decision for q both prior, and after, the removal of r_3. That is, if either of r_1, r_2 of Fig. 4 is enforceable on q, p will respond to q with either r_1's or r_2's decision, irrespective of whether r_3 is removed from S_p. Thirdly, at least one rule r' with a precedence less or equal than that of r_3, and a decision different than that of r_3 (i.e. rule r_4 of Fig. 4), must be enforceable on q. This is because if no such rule exists, i.e. if all rules with a precedence less than or equal to that of r_3 entail the same decision as r_3 (i.e. *deny*) then, after the removal of r_3, p cannot respond to q with the decision *permit*. In addition, any rules that have a precedence greater than that of r', but less or equal than that of r_3, and which are associated with the same decision as r_3 should *not* be enforceable on q for otherwise, one of these rules, and not r', will be enforced on q. For instance, if in Fig. 4 there were a rule r'' which entailed the same *deny* decision as r_3 and had a precedence between that of r_3 and r_4, then that rule would be enforced on q after the removal of r_3, preventing r_4 from yielding a decision opposite than that of r_3.

Generalising, the class $R^{\bar{v}}_{p,r}$ of all requests that induce the response v for as long as r remains in S_p, and the opposite response $\bar{v}$, when r is removed from S_p, comprises all those requests q such that: (i) r is enforceable on q. (ii) No rule in S_p with a precedence higher than that of r, i.e. no rule from the set $U_{p,r} = \{r' \in S_p | r' <_p r\}$, is enforceable on q. (iii) At least one rule r' which is associated with the decision $\bar{v}$ and has a precedence less or equal than that of r, is enforceable on q, i.e. at least one rule from the set $D_{p,v,r} = \{r' \in S_p | r' \leq_p r \wedge C_{r'} \equiv \{\bar{v}\}\}$ is enforceable on q. In addition, no other rule r'' which is associated with the same decision v as r and has a precedence higher than that of r', but less or equal than that of r, is enforceable upon q, i.e. no rule from the set $B_{p,v,r} = \{r'' \in S_p | r' <_p r'' \leq_p r \wedge C_{r'} \equiv \{v\}\}$ is enforceable on q. Assertion 5 of Appendix D.1 provides a formal expression for $R^{\bar{v}}_{p,r}$.

4.3.3 Unaffected Requests. In an analogous manner, we discern the class $R^{v}_{p,r}$ of all requests that are *unaffected* by the removal of r from S_p, i.e. they induce the same response v both prior, and after, the removal of r. More specifically, $R^{v}_{p,r}$ comprises all those requests q such that either: (i) one or more rule with a precedence greater than that of r, i.e. one or more rule from the set $U_{p,r}$ (see above), is enforceable on q; or (ii) one or more rule with a precedence less or equal than that of r, but not r itself, is enforceable on q, i.e. one or more rule from the set $L_{p,v,r} = \{r' \in S_p | r' \leq_p r\}$ is enforceable on q; or (iii) r and one or more rule r' with a precedence less or equal than that of r and the same decision v as r's, is enforceable on q—i.e. one or more rule from the set $D'_{p,v,r} = \{r' \in S_p | r' \leq_p r \wedge C_{r'} \equiv \{v\}\}$ is enforceable on q; at the same time, no rule that bears the opposite decision $\bar{v}$ and has a precedence greater than that of r', but less or equal than that of r, is enforceable on q—i.e. no rule from the set $B_{p,v,r} = \{r'' \in S_p | r' <_p r'' \leq_p r \wedge C_{r'} \equiv \{\bar{v}\}\}$ is enforceable on q. Assertion 6 of Appendix D.1 provides a formal expression for $R^{v}_{p,r}$.

5 A SIMPLE EXAMPLE

Consider the ruleset of Fig. 4 and let the antecedent of each rule r_i ($i \in [1 \ldots 4]$) enjoy exactly one association, through the properties *hasObj*, *hasAct*, *hasSubj* and *hasCE*, with the individuals o, w, s and e_i respectively (see Fig. 5). e_1 represents a CE that demands that the physical location of the subject s of a request is identified by the individual $bldg_Y$ (see Fig. 5); e_2 represents a CE that demands that the network location from which s issues a request is the subnet 144.0.0.0/8; e_3 represents a CE that demands that a request is issued during the interval identified by the individual *nonWH* (stands for 'non working hours'); finally, e_4 represents a CE that demands that the physical location of s is identified by the individual $bldg_X$.

Suppose now that rule r_3 is removed from S_p. According to assertion 4 of Appendix D.1, the class R^{NA}_{p,r_3} of all requests that induce a *deny* decision prior to the removal of r_3, and a 'Not Applicable' decision after the removal of r_3, is formally expressed by assertion 7 of Appendix D.2. R^{NA}_{p,r_3} essentially includes all requests that bear the following characteristics: (i) they enjoy exactly one association, through the properties *hasObj*, *hasAct* and *hasSubj* with the individuals o, w and s respectively; (ii) they enjoy exactly one association with a CE that articulates that the request is issued during the interval *nonWH*; (iii) they do not enjoy any associations with any CEs that articulate that s resides in any of the physical locations $bldg_X$ or $bldg_Y$, or in the network location 144.0.0.0/8.

We now turn our attention to the class R^{permit}_{p,r_3} of requests that induce a *deny* decision prior to the removal of r_3, and a *permit* decision after the removal of r_3. According to assertion 5, this class is given by assertion 8 of Appendix D.2 as it essentially includes all requests that bear the following characteristics: (i) they enjoy exactly one association, through the properties *hasObj*, *hasAct* and *hasSubj* with the individuals o, w and s respectively; (ii) they do not enjoy any associations with any CEs that articulate that s resides either in the physical location $bldg_Y$ or in the network location 144.0.0.0/8; (iii) they enjoy exactly one association with a CE that

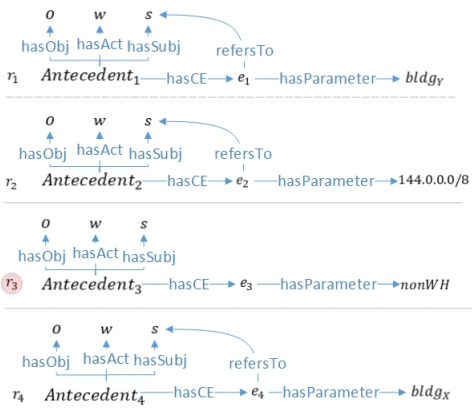

Figure 5: Example rules

articulates that the request is issued during the interval *nonWH*; (iv) they enjoy exactly one association with a CE that articulates that s resides in the physical location $bldg_X$.

Finally, the class R^{v}_{p,r_3} of all requests that remain unaffected from the removal of r_3 (i.e. they continue to invoke the same response v both prior and after the removal of r_3) is given, according to assertion 6, by assertion 9 of Appendix D.2. R^{v}_{p,r_3} essentially includes all requests that bear the following characteristics: (i) they enjoy exactly one association, through the properties *hasObj*, *hasAct* and *hasSubj* with the individuals o, w and s respectively; (ii) they either enjoy exactly one association with a CE that articulates that s resides in $bldg_Y$ or in 144.0.0.0/8, or they enjoy exactly one association with a CE that articulates that s resides in $bldg_X$ but do not enjoy any associations with any CEs that articulate that the request is issued during the interval *nonWH*.

6 RELATED WORK

A number of approaches have been proposed for semantically representing policies [6, 8, 11]. These generally rely on OWL [18] for capturing the knowledge artefacts that dwell in policies. In [11] KaoS is presented—a framework providing: a human interface layer for the expression of policies, a policy management layer for the manipulation of policies, and a monitoring and enforcement layer that encodes policies in a programmatic format suitable for their enforcement. Nevertheless, KaoS's mechanism for policy management is mainly oriented towards resolving conflicting policies and does not provide any determination of the repercussions of policy retirement actions.

In [6] Rei is proposed. Rei provides a framework for specifying, analysing and reasoning about policies. Similar to our work, a policy comprises a set of rules and it is also associated with an underlying context that defines its domain of application. Nevertheless, unlike our work, Rei resorts to the use of constructs adopted from rule-based programming languages for defining and reasoning about policy rules. In particular, it resorts to the use of placeholders for the definition of variables which are required for expressing rule attributes whose values are defined relative to the values of other rule attributes. This, however, essentially prevents Rei from exploiting the full inferencing potential of OWL as policy rules are modelled in a formalism that is alien to OWL. In addition, Rei does not provide any mechanism for determining the repercussions of policy retirement actions.

In [8] the authors propose POLICYTAB: a framework designed to facilitate trust negotiation in Semantic Web environments. POLICYTAB adopts an ontological approach to the representation of policies that guide a trust negotiation process that aims at granting, or denying, requests to act upon sensitive Web resources. These policies essentially articulate the credentials that an entity must possess in order to carry out a certain action on a sensitive resource that is under the ownership of another entity. Nevertheless, no attempt is made to semantically model the context associated with access requests; in addition, no mechanism is provided for determining the repercussions of policy retirement actions.

The work reported in [7] presents a formalisation of XACML using DLs which allows the use of off-the-shelf reasoners for supporting a wide range of policy-related actions such as comparisons between policies, policy verification and policy querying. Similar to our work, these actions are underpinned by a semantic characterisation of access requests. Nevertheless, unlike our work, no underlying ontological model for rules, policies and policy sets is provided thus precluding any explicit modelling of the precedence imposed by combining algorithms. To overcome this limitation, the authors resort to the use of Defeasible DLs (DDLs) for capturing precedence implicitly. This, however, incurs the overhead of reducing provability in DDLs to concept satisfiability in OWL and also, as the authors concede, hinders the expression of certain combining algorithms. In addition, it fails to model the potentially complex contextual knowledge attached to rules and thus precludes from the outset any context-based semantic inferencing that may be required in dynamic cloud environments. Last but not least, it fails to provide any mechanisms for determining the repercussions of policy retirement actions.

7 CONCLUSIONS

We have presented an approach to determining the repercussions of retirement actions targeted at the rules comprised by access control policies in dynamic cloud environments. The determination is based on a semantic characterisation of all those access requests that are affected by the retirement of one or more policy rules, and it is performed automatically, through semantic inferencing that is carried out by off-the-shelf reasoners. We are currently in the process of developing a mechanism that implements this approach.

As part of future work, we intend to incorporate this mechanism in an editor (that is currently also under development) that aims at facilitating the governance of ABAC policies. This way, each time stakeholders issue retirement actions, the editor will be able to inform them about the repercussions of their actions and hence assist them in governing effectively their policies.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644814.

A FORMAL REPRESENTATION OF A CONTEXT EXPRESSION

$$\{e\} \sqsubseteq (\leq 1\ refersTo.\{s\}) \sqcap ((\leq 1\ hasParam.\{bldg_Y\}) \sqcup (\leq 1\ hasParam.\{144.0.0.0/8\})) \tag{1}$$

Note that, for any object property P and concept C, $(\leq 1P.C)$ represents the abstract class that comprises all those individuals that have at least one association through P with an instance of C. The symbols $\sqcup$ and $\sqcap$ represent, respectively, class union and intersection. Also note that all concepts and properties of the ABAC policy model belong to the *pac* namespace (stands for PaaSword Access Control) [13]; in order to avoid notational clutter, in this work we omit the *pac* prefix from concept and property identifiers.

B FORMAL REPRESENTATION OF A RULE ANTECEDENT

$$A_r \equiv ((=1\ hasObj.\{o\}) \sqcap (=1\ hasSubj.\{s\}) \sqcap (=1\ hasAct.\{w\}) \sqcap (=1\ hasCE.C)) \tag{2}$$

Example abstract class that represents (in $\mathcal{SROIQ}$) the antecedent of a rule r. Note that, for any object property P and concept C, $(=1P.C)$ represents the abstract class that comprises all those individuals that feature *exactly one* association through P with an instance of C; it is an abbreviation for the $\mathcal{SROIQ}$ notation $(\leq 1\ P.C) \sqcap (\geq 1\ P.C)$.

C ORDERED DO AND PO ALGORITHMS

The ordered versions of the DO and PO algorithms invariably drive a policy to respond to an access request with exactly the same decision as their unordered counterparts. It follows that the ordered and unordered versions of the DO and PO algorithms are indistinguishable when determining the repercussions of rule removals. This is because, as already discussed, these repercussions are determined on the basis of the manner in which access requests are responded to by a policy. The ordered versions of the DO and PO algorithms will therefore not further concern us in this paper.In addition, the following variations of the DO and PO algorithms are often encountered in practice: '*permit unless deny*' and '*deny unless permit*'. The former algorithm (meaning that p yields by default a 'permit' decision unless one of its 'deny' rules, if any, is enforced) becomes equivalent to the DO algorithm through the inclusion in S_p of a special 'permit' rule that has OWL's universal class as its antecedent (and thus is enforceable upon any access request, regardless of the attributes piggybacked on that request); an entirely symmetrical treatment applies to the latter algorithm.

D FORMAL REPRESENTATION OF ABSTRACT CLASSES

D.1 Abstract Classes of Section 4

The class of all requests for which p yields the decision v:

$$R_{p,v} \equiv \bigsqcup_{r \in S_{p,v}} (A_r \sqcap \neg(\bigsqcup_{r' \in H_{p,v,r}} A_{r'})) \tag{3}$$

The class of all requests that induce v prior to the removal of r, and 'Not Applicable' after the removal of r:

$$R^{NA}_{p,r} \equiv A_r \sqcap \neg \bigsqcup_{\{r' \in S_p | r' \neq r\}} A_{r'} \tag{4}$$

The class of all requests that induce v prior to the removal of r, and $\bar{v}$ after the removal of r:

$$R^{\bar{v}}_{p,r} \equiv A_r \sqcap (\neg \bigsqcup_{r' \in U_{p,r}} A_{r'}) \sqcap (\bigsqcup_{r' \in D_{p,v,r}} A_{r'} \sqcap (\neg \bigsqcup_{r'' \in B_{p,v,r}} A_{r''})) \tag{5}$$

The class of all unaffected requests that induce v both prior and after the removal of r:

$$R^{v}_{p,r} \equiv (\bigsqcup_{r' \in U_{p,r}} A_{r'}) \sqcup (\neg A_r \sqcap \bigsqcup_{r' \in L_{p,v,r}} A_{r'}) \sqcup (A_r \sqcap (\bigsqcup_{r' \in D'_{p,v,r}} A_{r'} \sqcap (\neg \bigsqcup_{r'' \in B'_{p,v,r}} A_{r''}))) \tag{6}$$

D.2 Abstract Classes of Section 5

$$R^{NA}_{p,r_3} \equiv C \sqcap (=1\ hasCE.\{e_1\} \sqcap \neg(\bigsqcup_{i \in \{1,2,4\}} hasCE.\{e_i\})) \tag{7}$$

where $C \equiv (=1\ hasObj.\{o\}) \sqcap (=1\ hasAct.\{w\}) \sqcap (=1\ hasSubj.\{s\})$. It is to be noted here that assertion 7, as well as assertions 8 and 9 below, have been mathematically derived from the (considerably more complex) $\mathcal{SROIQ}$ expressions that result from the initial application of assertions 4–6 to the rule antecedents of Fig. 5; the mathematical derivation is based on the associativity and distributivity properties of the $\sqcap$ and $\sqcup$ operators and it is omitted here for reasons of space as it is inconsequential to the rest of the work.

$$R^{permit}_{p,r_3} \equiv C \sqcap \neg(\bigsqcup_{i \in \{1,2\}} hasCE.\{e_i\}) \sqcap (=1\ hasCE.\{e_3\}) \sqcap (=1\ hasCE.\{e_4\}) \tag{8}$$

$$R^{v}_{p,r_3} \equiv C \sqcap ((\bigsqcup_{i \in \{1,2\}} hasCE.\{e_i\}) \sqcup (\neg(=1\ hasCE.\{e_3\}) \sqcap (=1\ hasCE.\{e_4\}))) \tag{9}$$

REFERENCES

[1] 2015. PaaSword - A Holistic Data Privacy and Security by Design Platform-as-a Service Framework. https://www.paasword.eu. (2015).

[2] Cloud Security Alliance 2015. *What's Hindering the Adoption of Cloud Computing in Europe?* Cloud Security Alliance. https://blog.cloudsecurityalliance.org/2015/09/15/whats-hindering-the-adoption-of-cloud-computing-in-europe/.

[3] Cloud Security Alliance 2016. *The Treacherous 12.* Cloud Security Alliance. https://cloudsecurityalliance.org/download/the-treacherous-twelve-cloud-computing-top-threats-in-2016/.

[4] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. 2006. The Even More Irresistible SROIQ. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning*, Patrick Doherty, John Mylopoulos, and Christopher A. Welty (Eds.). AAAI Press, 57–67.

[5] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone. 2014. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations.* NIST. http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf.

[6] L. Kagal, T. Finin, and A. Joshi. 2003. A policy language for a pervasive computing environment. In *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*. 63–74. DOI: https://doi.org/10.1109/POLICY.2003.1206958

[7] Vladimir Kolovski, James Hendler, and Bijan Parsia. 2007. Analyzing Web Access Control Policies. In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*. ACM, New York, NY, USA, 677–686. DOI: https://doi.org/10.1145/1242572.1242664

[8] Wolfgang Nejdl, Daniel Olmedilla, Marianne Winslett, and Charles C. Zhang. 2005. Ontology-Based Policy Specification and Management, Asunción Gómez-Pérez and Jérôme Euzenat (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 290–302. DOI: https://doi.org/10.1007/11431053_20

[9] OASIS 2013. *eXtensible Access Control Markup Language (XACML) Version 3.0*. OASIS. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

[10] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. 2007. Pellet: A Practical OWL-DL Reasoner. *Web Semant.* 5, 2 (June 2007), 51–53. DOI: https://doi.org/10.1016/j.websem.2007.03.004

[11] A Uszok, J Bradshaw, R Jeffers, M Johnson, A Tate, J Dalton, and S Aitken. 2004. KAoS Policy Management for Semantic Web Services. *IEEE Intel. Sys.* 19, 4 (2004), 32–41.

[12] Simeon Veloudis and Iraklis Paraskakis. 2015. *Access Policies Model.* PaaSword Project Deliverable D3.2.

[13] Simeon Veloudis and Iraklis Paraskakis. 2015. *Access Policies Model.* PaaSword Project Deliverable D2.2.

[14] Simeon Veloudis and Iraklis Paraskakis. 2016. Defining an Ontological Framework for Modelling Policies in Cloud Environments. In *8th IEEE International Conference on Cloud Computing Technology and Science (CloudCom'16)*.

[15] Simeon Veloudis, Iraklis Paraskakis, Chris Petsos, Yiannis Verginadis, Ioannis Patiniotakis, and Gregoris Mentzas. 2017. An Ontological Template for Context Expressions in Attribute-based Access Control Policies. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,*. INSTICC, ScitePress, 151–162. DOI: https://doi.org/10.5220/0006301501510162

[16] Simeon Veloudis, Yiannis Verginadis, Iraklis Paraskakis, Ioannis Patiniotakis, and Gregoris Mentzas. 2016. Context-aware Security Models for PaaS-enabled Access Control. In *Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016) Vol. 1 and 2*. INSTICC, ScitePress, 201–212.

[17] Yiannis Verginadis, Ioannis Patiniotakis, and Gregoris Mentzas. 2015. *Context-aware Security Model, PaaSword Project Deliverable D2.1.* https://www.paasword.eu/wp-content/uploads/2016/09/D2-1_Context-awareSecurityModel.pdf.

[18] W3C 2004. *W3C Recommendation. 2004. OWL Web Ontology Language Reference.* W3C. https://www.w3.org/TR/owl-ref/.