Ontological Templates for Modelling Security Policies in Cloud Environments

Simeon Veloudis and Iraklis Paraskakis
South East European Research Centre (SEERC)
The University of Sheffield International Faculty, CITY College
Thessaloniki, Greece
(+30)2310253477
{sveloudis, iparaskakis}@seerc.org

ABSTRACT

It is generally conceded that by embracing the cloud computing paradigm enterprises are able to boost their agility and productivity whilst realising significant cost savings. However, many enterprises are reluctant to adopt cloud services for supporting their critical operations due to security and privacy concerns. One way to alleviate these concerns is to devise a set of *policies* that infuse adequate security controls in cloud services. However, the heterogeneous nature of these services, together with the dynamicity inherent in cloud environments, hinders the formulation of an effective and interoperable set of policies that is suitable for the underlying domain of application. To this end, this work proposes an approach to the construction of *ontological templates* for the *semantic representation* of policies. These templates are capable of capturing the *knowledge* that must be infused into a policy in order for it to adequately take into account the needs of the underlying domain of application in which it is to be enforced.

Categories and Subject Descriptors

• Software and its engineering ~ Cloud computing
• Information systems ~ Semantic web description languages
• Security and Privacy ~ Web application security.

Keywords

Cloud computing; cloud security; security policies; service description languages; Linked USDL

1. INTRODUCTION

Enterprises increasingly embrace the cloud computing paradigm in order to gain access to a wide range of infrastructure, platform, and application resources that are abstracted as services and delivered remotely by diverse providers [4]. The main force that fuels this trend is the significant cost savings that these services instigate, as well as the acceleration of the development and deployment of new applications that boosts innovation and productivity. However, due to security concerns, many enterprises are reluctant to move their critical operations and sensitive data to the cloud [24] [5]. One way to alleviate these concerns is to devise suitable *policies* that infuse adequate security controls in cloud services. For these policies to be effective, however, they must possess the following desirable characteristics. Firstly, they must cater for the particular needs of the underlying domain of application in which they are to be used. Secondly, they must take into account the inherently dynamic and unpredictable nature of cloud environments and the heterogeneity of cloud services. Thirdly, they must be understandable, hence applicable, across the diverse administrative domains that a cloud environment may span. Last but not least, they must lend themselves to automated checks concerning their correctness, hence their effectiveness. These characteristics call for the introduction of a novel framework for the representation of security policies, one that *generically* provides the means to express the *knowledge* that dwells in security policies, whilst advocating a clear separation of concerns by unravelling the representation of policies from the code that is ultimately employed for enforcing them.

This work introduces such a framework. More specifically, it proposes a novel approach to the construction of a set of *ontological templates* that are capable of semantically capturing policies in dynamic cloud environments. In order to model the knowledge that is reflected in policies, the proposed templates are underlain by an ontological description of a set of concepts, and the properties thereof, that are relevant to the particular domain of application. For example, concerning access control policies in dynamic cloud environments, an ontological description of such concepts as 'access location', 'access time', 'access subject' and 'type of access' (e.g. 'read' or 'write'), would be required.

One of the main strengths of our approach is that the proposed templates are formulated using an extensible RDF vocabulary that is amenable to automated reasoning about the *compliance* of policies with a set of *constraints* regarding their content and structure. Such a vocabulary therefore lays the foundation for building a *higher-level ontology* capable of accommodating these constraints through the use of a more verbose (than RDF) formalism such as OWL 2 [12]. This bears the advantage that the constraints are expressible in the same representation as the actual policies that they regulate, namely as RDF graphs, and therefore facilitates the construction of a *policy validator* – a mechanism capable of determining the validity of the policies with respect to these constraints.

Although the proposed templates are applicable to a broad range of security policies, this work focuses on the security policies encountered in the PaaSword project [15]. PaaSword aspires to provide a security-by-design solution, essentially a PaaS offering, that facilitates developers in formulating suitable *security* policies for dynamic cloud environments. More specifically, PaaSword

PCI '16, November 10 - 12, 2016, Patras, Greece

ACM 978-1-4503-4789-1/16/11 $15.00
DOI: http://dx.doi.org/10.1145/3003733.3003796

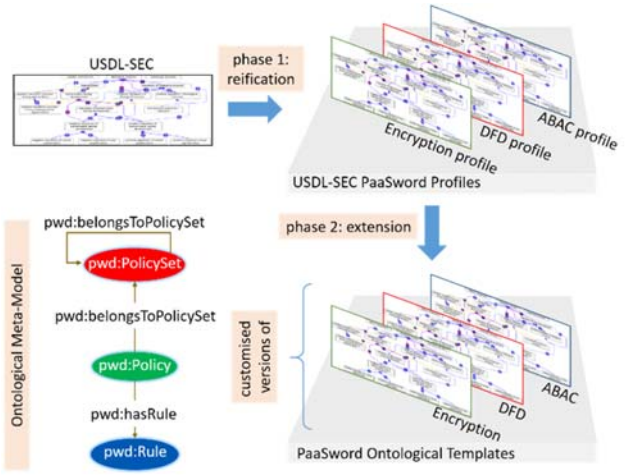

Figure 1. Approach to security policy representation

aims at supporting three types of policy: (i) *Data Encryption policies* that articulate the kind of cryptographic protection that a sensitive data object must enjoy in the cloud; (ii) *Data Fragmentation and Distribution (DFD) policies* that specify the way in which sensitive data objects, e.g. columns in a relational table, are fragmented and distributed across different cloud servers for privacy reasons; (iii) *Attribute-Based Access Control (ABAC) policies* that articulate when to allow, or forbid, access to sensitive data in the cloud on the basis of a broad range of relevant contextual attributes.

The rest of this paper is structured as follows. Section 2 presents our generic approach to the representation of security policies. Section 3 outlines the construction of an ontological template for capturing PaaSword's DFD policies. Section 4 summarises related work and Section 5 presents conclusions and future work.

2. A GENERIC APPROACH TO SECURITY POLICY REPRESENTATION

We propose an approach to the semantic representation of diverse kinds of security policy. Our approach accurately captures the knowledge underpinning each policy kind, whilst it disentangles the representation of policies from the actual code employed for enforcing them. At the same time, it lends itself to, and therefore paves the way for, a series of *correctness* checks that are performed automatically by a *policy validator*. Although our focus in this work is on the policy kinds supported by the PaaSword project, namely Data Encryption policies, DFD policies and ABAC policies, the proposed approach is equally applicable to any other kind of security policy.

As depicted in Figure 1, our approach comprises two main phases. In the first phase, the ontological framework of concepts and properties offered by USDL-SEC, Linked USDL's security profile [10], is *reified* giving rise to three distinct *profiles*, one for each kind of security policy considered, namely the *Encryption profile*, the *DFD profile* and the *ABAC profile*. In the second phase, each profile is *extended* by incorporating a customised version of the *ontological meta-model* depicted in Figure 1. This meta-model forms essentially an abstract template that can be suitably extended (see Section 3.2 for more details) in order to capture the semantics of a particular policy kind.

The extension that takes place in the second phase aims at encriching each profile devised in the first phase with a suitable framework of concepts and properties that accurately captures the semantics of the corresponding policy kind. In this respect, it essentially transforms each profile into an *ontological template* capable of accommodating all those knowledge artefacts that are required for describing policies of the particular kind. For instance, regarding Data Encryption policies, the profile would be extended with concepts and properties for accommodating such knowledge artefacts as the particular encryption kind used (e.g. symmetric, asymmetric, hybrid), the particular encryption algorithm applied, the encryption strength, etc. Regarding DFD policies, the profile would be extended with concepts and properties for accommodating such knowledge artefacts as the particular data fragmentation scheme used (e.g. vertical or horizontal fragmentation), as well as the particular data distribution scheme adopted (e.g. preferred or fobidden data storage locations, preferred of forbidden data storage providers, etc.). Finally, regarding ABAC policies, the profile would be extended with concepts and properties for accommocdating such knowledge artefacts as the identity of the subject requesting the access, the actual action requested (e.g. read/write), the data object on which the action is requested, the location of access, the time of access, etc.

The reifications and extensions that take place in the two phases of the proposed approach are further elaborated in Section 3. A brief discussion that motivates our choice to adopt Linked USDL as the basis of our approach, as well as an outline of the concepts and properties of the ontological framework offered by USDL-SEC are, however, first in order.

2.1 Linked USDL

Linked USDL [10] is a remodelled version of USDL [1] which draws upon numerous research efforts in the area of Semantic Web Services and business ontologies [2], [3]. It capitalises on the principles of Linked Data in order to support its use in a 'web of data'. Specifications are expressed in an RDF vocabulary [16] that provides sufficient support for the generic description of cloud services. Linked USDL consists of a Core schema and a number of extension schemata, or profiles, that address diverse aspects of a cloud service. In our work, we are interested in the Security profile (USDL-SEC) which offers the foundation for the generic representation of a broad range of security policies. Our reliance on Linked USDL brings about the following benefits [3]. (i) Linked USDL utilises widely-used RDF vocabularies, such as GoodRelations [8], SKOS [18], and FOAF [20] in order to promote the sharing of knowledge whilst increasing the interoperability, hence the reusability, of our policies. (ii) By incorporating a set of different profiles, Linked USDL offers a holistic and generic solution that conveniently captures a wide range of business details that are required for accurately describing our policies. (iii) Linked USDL is designed to be easily extensible through linking to further existing, or new, ontologies. This crucially facilitates the seamless integration of our ontological templates with the vocabularies devised in [13] for describing the concepts involved in the definition of the PaaSword security policies (see Section 3 for more details).

2.2 USDL-SEC

As depicted in Figure 2, USDL-SEC defines the following top-level classes: *Security Profile*, *Security Goal*, *Security Mechanism*, *Security Technology* and *Security Realization Type*. The Security Profile class, as its name suggests, introduces a set of security profiles to which a cloud application potentially adheres. A security profile is invariably related to a security goal. This brings about the Security Goal concept which comprises a set of sub-concepts that capture particular security goals. A complete list of all such security goals is depicted in Figure 2. A security goal is tied to the security mechanism in terms of which it is realised. This introduces the

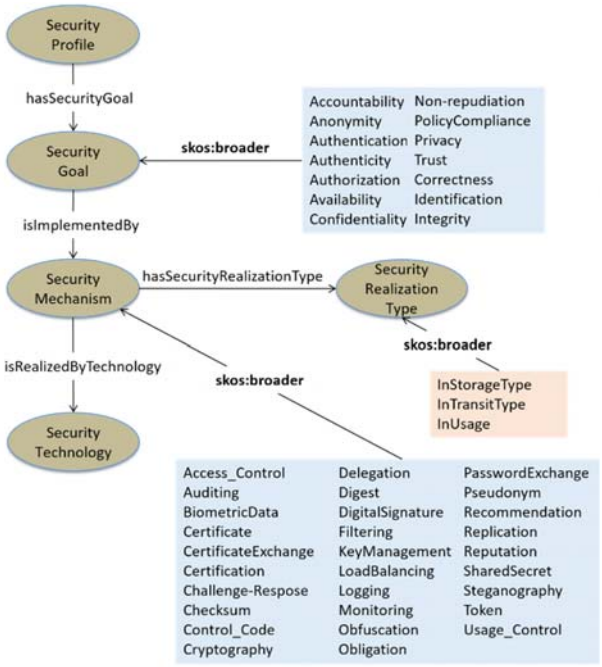

Figure 2. USDL-SEC

Security Mechanism class which incorporates sub-concepts for capturing different types of security mechanisms – the list of all the security mechanism types supported by USDL-SEC is illustrated in Figure 2. A security mechanism is in turn tied to the security technology that implements it, giving rise to the Security Technology concept. Furthermore, a security mechanism is related to the layer of the ISO/OSI protocol stack at which it is realised (for example, the Application or Network layer). This brings about the Security Realization Type class that specifies this layer.

The concepts and their relations defined above are declaratively captured in terms of ontological classes and object properties respectively[1]. More specifically, the following object properties are introduced: `hasSecurityGoal` that interrelates a security profile with its security goal; `isImplementedBy` that interrelates a security goal with the mechanism that realises it; `isRealizedByTechnology` that interrelates a security mechanism with the underlying technology that implements it; `hasSecurityRealizationType` that interrelates a security mechanism with the particular ISO/OSI layer at which it operates. The fact that a concept constitutes a sub-concept of another concept is expressed through the use of the property `skos:broader` [18].

The above framework of concepts and their associations lays the foundations for constructing ontological templates that are suitable for the semantic characterisation of the security policies that the PaaSword project aspires to sustain. Section 3 elaborates on how this framework is reified in order to give rise to an ontological template for the expression of DFD policies. Analogous accounts apply to the other two kinds of policy, namely Data Encryption policies and ABAC policies; the interested reader is referred to [14] for more details.

3. CONSTRUCTING THE DFD ONTOLOGICAL TEMPLATE

As outlined at the beginning of Section 2, the DFD ontological template is constructed through a two-phase process. In the first phase, the framework of classes and properties offered by USDL-SEC is reified to give rise to the DFD profile. In the second phase, the DFD profile is extended with additional concepts and properties that capture those knowledge artefacts that are required for accurately expressing DFD policies. These reifications and extensions are further elaborated below.

3.1 Reifying USDL-SEC: The DFD Profile

The reification of the concepts and properties offered by USDL-SEC proceeds either by using existing classes that already appear in the USDL-SEC vocabulary (e.g. the *Security Goal* concept is reified in terms of its narrower *Privacy* concept – see Figure 3), or by introducing new concepts along with their associations.

The DFD profile is itself represented as an instance of the class `SecurityProfile`, namely the instance `pdfd:DFDProfile`[2] (see Figure 3). The security goal of this profile is *privacy*. As depicted in Figure 3, this is modelled by reifying the `SecurityGoal` class in terms of the USDL-SEC `Privacy` class and defining an instance of this class, namely `pdfd:DFDGoal`, to represent the particular privacy goal. This instance is then associated, through the property `hasSecurityGoal`, with the instance `pdfd:DFDProfile`.

The privacy goal is implemented by a security mechanism, namely the *DFD mechanism*. This is captured by reifying the `SecurityMechanism` class in terms of the class `pdfd:DFD` and defining an instance in this class, namely `pdfd:DFDMechanism`, which represents the actual DFD mechanism. This instance is then associated, through the property `isImplementedBy`, with the instance `pdfd:DFDGoal` which, as described above, represents the security goal. Note that the class `pdfd:DFD` represents a new concept specifically introduced into USDL-SEC for modelling DFD policies.

The DFD mechanism operates at the Application layer of the ISO/OSI protocol. This is captured by refining the `SecurityRealizationType` class in terms of the USDL-SEC `InUsageType` class and introducing an instance of this class, namely `pdfd:DFDType`. This instance is then associated, through the property `hasSecurityRealizationType` with the instance `pdfd:DFDMechanism` that represents the DFD mechanism.

Finally, the DFD mechanism is realised in terms of a particular technology, one that relies on a set of DFD policies. In order to model such a security technology, a number of new concepts, along with their associations, are introduced. These are the `pdfd:PaaSwordDFD` concept, which accommodates the security technology, and the `pdfd:DFDPolicySet` concept, which bundles together the DFD policies on which the security technology relies. In particular, the security technology takes the form of the instance `pdfd:DFDTechnology` (see Figure 3) and is associated with the DFD mechanism through the USDL-SEC property `isRealizedByTechnology`. The `pdfd:DFDPolicySet` concept is associated with the security technology through the property `pdfd:hasDFDPolicySet`.

[1] USDL-SEC classes and properties are defined under the `usdl-sec` namespace that is omitted here to reduce notational clutter.

[2] The namespace `pdfd` (stands for "PaaSword DFD") is where all DFD-related concepts and properties are defined.

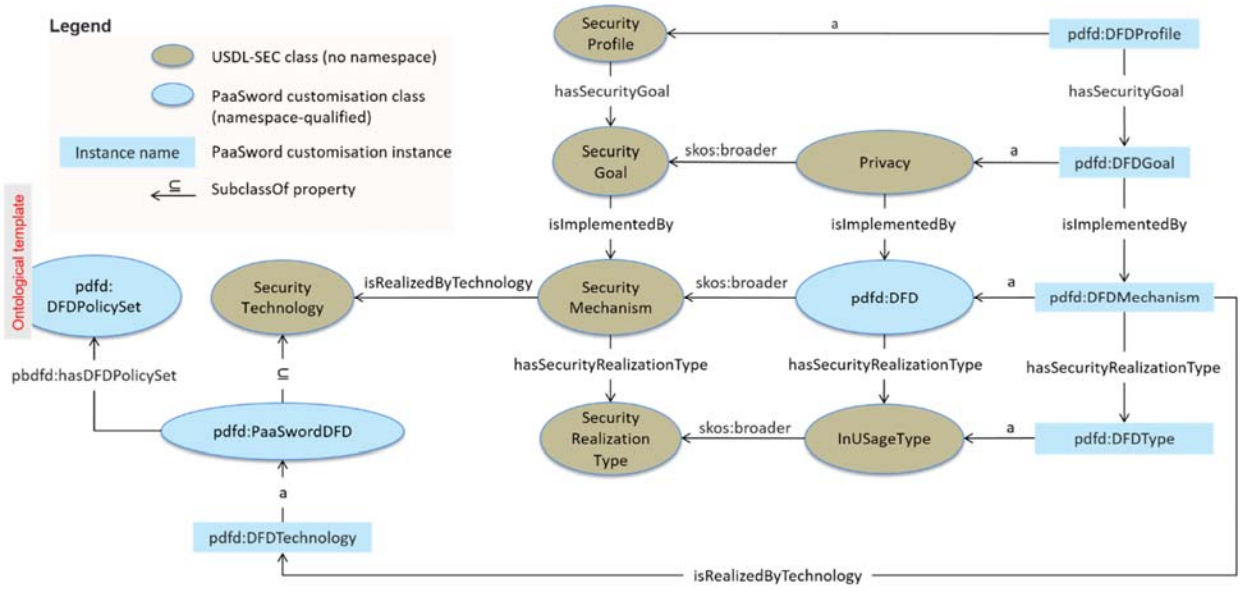

Figure 3. The DFD Profile

This concept essentially *extends* the DFD profile with a generic *ontological template* capable of accommodating all those knowledge artefacts that are required for describing DFD policies. This ontological template is further elaborated in Section 3.2 below.

3.2 Extending the DFD Profile: The DFD Ontological Template

The DFD ontological template is a customised version of the ontological meta-model depicted in Figure 1. In fact, in our approach, any ontological template that aspires to semantically describe a particular kind of security policy forms a customised version of this meta-model. Section 3.2.1 below outlines the concepts and properties that make up this meta-model; Sections 3.2.2, 3.2.3 and 3.2.4 describe how this meta-model is customised in order to give rise to the DFD ontological template. Analogous accounts apply for the ontological templates of the other two kinds of PaaSword policy, namely the Data Encryption ontological template and the ABAC ontological template depicted in Figure 1; the interested reader is referred to [14] for more details.

3.2.1 Ontological Meta-Model

Following an approach inspired by the XACML standard [7], three levels of structural elements are discerned for the ontological meta-model: *Rules*, *Policies* and *Policy sets* (see Figure 1). A rule is the most elementary structural element and the basic building block of policies. In this respect, rules are the carriers of the core logic of policies. They are defined as instances of the class `pwd:Rule`[3].

A policy comprises one or more rules and is represented as an instance of the class `pwd:Policy` (see Figure 1). A policy is associated with its constituent rules through the object property `pwd:hasRule` (see Figure 1). Policies are also grouped into *policy sets*. A policy set is modelled as an instance of the class `pwd:PolicySet`. A policy is associated with its containing policy set through the object property `pwd:belongsToPolicySet` (see Figure 1). A policy set may exhibit a hierarchical structure and comprise one or more other policy sets. In order to capture such a recursive inclusion of policy sets, the `pwd:belongsToPolicySet` property is rendered applicable to policy sets as well (i.e. in addition to policies).

3.2.2 Ontological Representation of DFD Rules

A DFD rule is abstractly described by the *template* illustrated in Table 1.

Table 1. DFD rule template

[*controlled object*] is fragmented with [*fragmentation scheme*] and distributed with [*distribution scheme*]

This template specifies a generic structure, in terms of relevant attributes, to which all DFD rules adhere. It comprises the attributes *controlled object*, *fragmentation scheme* and *distribution scheme*. The first attribute identifies the sensitive data object which is to be fragmented and distributed. Its values are drawn from the class `pcm:Object` of the Security Context Element model – a vocabulary of relevant classes and properties that has been devised as part of the PaaSword project for describing the various contextual attributes that may appear in a policy rule. For reasons of space, this vocabulary is omitted here; the interested reader is referred to [23], [13] for more details.

The second and third attributes identify, respectively, the data fragmentation and distribution schemes that are to be applied to the controlled object. They draw their values from the classes `pdm:DataFragmentation` and `pdm:DataDistribution` respectively of the Data Distribution and Encryption model defined in [13]. These classes encompass a framework of concepts and properties for accomodating relevant knowledge artefacts such as the particular fragmentation scheme used (e.g. vertical or horizontal fragmentation) and the particular data distribution scheme adopted (e.g. preferred or fobidden data storage locations, preferred of forbidden data storage providers). The works in [23], [13] provide

[3] The namespace `pwd` is where all concepts and properties related to the ontological meta-model are defined.

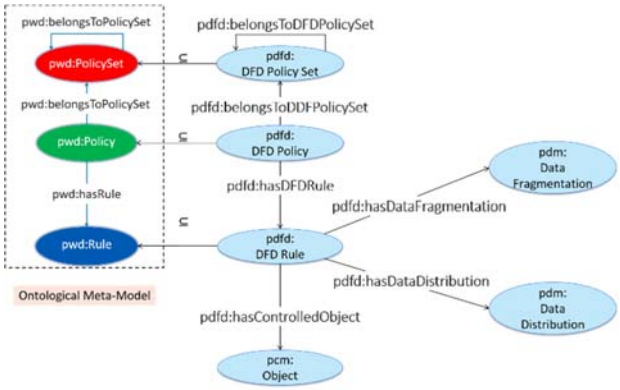

Figure 4. The DFD Ontological Template

more details on the concepts and properties that make up the Data Distribution and Encryption model.

The rule template of Table 1 is expressed ontologically as a customised version of the `Rule` concept of the ontological meta-model of Section 3.2.1. More specifically, the template itself is represented as an instance of the class `pdfd:DFDRule` which is defined as a subclass of the class `pwd:Rule` (see Figure 4). The template is associated with its constituent attributes through the object properties `pdfd:hasControlledObject`, `pdfd:hasDataFragmentation` and `pdfd:hasDataDistribution` (see Figure 4).

3.2.3 Ontological Representation of DFD Policies

Similar to DFD rules, DFD policies take the form of customised versions of the `pwd:Policy` concept of the ontological meta-model of Section 3.2.1. More specifically, individual DFD policies take the form of instances of the class `pdfd:DFDPolicy` which is defined as a subclass of the class `pwd:Policy` (see Figure 4). A DFD policy is tied to the rules that it comprises through the object property `pdfd:hasDFDRule`. This property is a sub-property of the generic property `pwd:hasRule` introduced by the policy meta-model[4].

3.2.4 Ontological Representation of DFD Policy Sets

Similar to DFD policies, DFD policy sets take the form of customised versions of the `pwd:PolicySet` concept of the ontological meta-model of Section 3.2.1. More specifically, individual DFD policy sets take the form of instances of the class `pdfd:DFDPolicySet` which forms a subclass of the class `pwd:PolicySet` class (see Figure 4). A DFD policy set is associated with the DFD policies that it comprises through the object property `pdfd:belongsToDFDPolicySet`. This property is a sub-property of the generic property `pwd:belongsToPolicySet` introduced by the ontological meta-model. Note that a DFD policy set may exhibit a hierarchical structure and encompass nested DFD policy sets. In order to capture such a recursive inclusion, the `pdfd:belongsToDFDPolicySet` property is also applicable to DFD policy sets (i.e. in addition to policies).

4. RELATED WORK

A number of different approaches to policy representation have been proposed [6], [7], [17], [25], [21], [9], [11]. [6] proposes PONDER – a domain-specific language for modelling security and management policies; [7], [17], [25] propose markup languages for articulating access control policies. Although these formalisms promote a separation of concerns by unravelling the representation of policies from the code employed for enforcing them, they lack semantic agreement outside the confines of the organisations that created the policies. Any interoperability thus hinges on ad-hoc vocabularies that are shared by the various stakeholders that participate in an interaction. This inevitably entails the following shortcomings: (i) it restricts the portability of policies as well as their reusability; (ii) it hinders the determination of inter-policy relations; (iii) it leads to ad-hoc reasoning about policy compliance, one which is tangled with the particular vocabularies that are utilised for articulating the rules according to which the reasoning takes place; (iv) it impedes the performance of rule-based policy governance.

In an attempt to overcome these shortcomings, a number of semantically-rich approaches for the representation of policies have been proposed [21], [9], [11]. These embrace OWL for capturing the knowledge that underlie the policies. More specifically, [21] presents KAoS – a layered framework for articulating, enforcing and managing policies. KAoS comprises: (i) a human interface layer for the expression of policies; (ii) a policy management layer that ontologically captures the knowledge that resides in policies; (iii) a monitoring and enforcement layer that encodes this knowledge in a programmatic format suitable for enforcing the policies. In [9] Rei is proposed – an ontology that captures a broad range of policies through the provision of a suitable abstraction for the representation of a set of desirable behaviours that are exhibited by autonomous entities. In [11], the authors propose POLICYTAB for facilitating trust negotiation in Semantic Web environments. POLICYTAB adopts ontologies for the representation of policies that guide a trust negotiation process ultimately aiming at granting, or denying, access to sensitive Web resources. These policies essentially specify the credentials that an entity must possess in order to carry out an action on a sensitive resource that is under the ownership of another entity.

Although the aforementioned semantically-rich approaches succeed in capturing the knowledge artefacts that are reflected in a policy, their reliance on the standards semantics of OWL [12], hence on the Open World Assumption that OWL is based upon, hinders the expression of *constraints* regarding the content and structure of a policy [19]. This impedes the construction of a policy validator that evaluates the correctness of policies by assessing their conformance with such constraints. In contrast, the reliance of our approach on RDF for the expression of policies raises this limitation.

[4] It is to be noted here that we could alternatively devise a model whereby each DFD policy refers to exactly one controlled object and to exactly one DFD scheme and hence comprises exactly one rule. This means that we could completely dispense with the concept of DFD policies and, instead, rely entirely on the concept of DFD rules. Nevertheless, we refrain from doing so for two main reasons: (i) We would like to retain a uniform ontological structure, one which abides by the generic ontological meta-model of Section 3.2.1. (ii) For increased generality, we would like to retain the possibility of a single multi-rule DFD policy being capable of protecting a multitude of sensitive objects with possibly different DFD schemes.

5. CONCLUSIONS AND FUTURE WORK

This work has proposed an approach to the construction of ontological templates for the representation of security policies in cloud environments. The templates aim at facilitating the definition of appropriate security policies that give rise to effective security controls for protecting sensitive data in the cloud. An advantage of our approach is that the proposed templates are formulated using a generic and extensible RDF vocabulary that lends itself to a series of correctness checks that can be performed automatically and orthogonally to the code employed for enforcing the policies. These checks aim at assessing the validity of a policy with respect to a higher-level ontology that articulates all those attributes whose values a policy may, or may not, restrict. These checks crucially increase our degree of assurance on the effectiveness of the security policies.

In the future, we plan to build the higher-level ontology. As already mentioned, this ontology imposes constraints on the actual RDF tripes that may, or may not, be encountered in a policy. The higher-level ontology will be expressed in the OWL 2 Web Ontology Language which provides the required expressivity for accurately articulating these constraints. A policy validator that parses the higher-level ontology and programmatically represents its contstraints can then be constructed in order to automatically determine whether a policy complies with these constraints. Moreover, we intend to build an editor which will facilitate the formulation of the aforementioned constraints in the higher-level ontology.

6. 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.

7. REFERENCES

[1] Barros, A. and Oberle, D.: Handbook of Service Description: USDL and its Methods, Springer (2012)

[2] Cardoso, J., Pedrinaci, C., Leidig, T., Rupino P. and Leenheer, P.: Foundations of Open Semantic Service Networks. International Journal of Service Science, Management, Engineering, and Technology, vol. 4, no. 2, 1-16 (2013)

[3] Cardoso, J., Pedrinaci, C., Leidig, T.: Linked USDL: a Vocabulary for Web-scale Service Trading. In 11th Extended Semantic Web Conference (ESWC) (2014)

[4] Cloud Computing Reference Architecture. Technical report, NIST (2011)

[5] CloudPassage, "Cloud Security Spotlight Report," LinkedIn, 2015

[6] Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The Ponder Policy Specification Language. In Sloman, M., Lobo, J., Lupu, E. (eds.) Proceedings of the International Workshop on Policies for Distributed Systems and Networks (POLICY '01), pp. 18--38, Springer-Verlag, London (2000)

[7] eXtensible Access Control Markup Language (XACML) Version 3.0. 22 January 2013. OASIS Standard. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[8] GoodRelations: The Professional Web Vocabulary for E-Commerce. http://www.heppnetz.de/projects/goodrelations/

[9] Kagal, L., Finin, T., Joshi, A.: A Policy Language for a Pervasive Computing Environment. In 4th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY '03), pp. 63--74, IEEE Computer Society, Washington, DC (2003)

[10] Linked USDL, http://www.linked-usdl.org/

[11] Nejdl, W., Olmedilla, D., Winslett, M, Zhang. C.C.: Ontology-Based policy specification and management. In Gómez-Pérez, A. and Euzenat, J. (eds.) ESWC'05, pp. 290-302, Springer-Verlag, Berlin, Heidelberg (2005)

[12] OWL 2 Web Ontology Language Primer (2nd Edition), https://www.w3.org/TR/owl2-primer/

[13] PaaSword Deliverable 2.1. https://www.paasword.eu/deliverables/

[14] PaaSword Deliverable 2.2. https://www.paasword.eu/deliverables/

[15] PaaSword project, http://www.paasword.eu/

[16] RDF 1.1 XML Syntax, http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/

[17] Security Assertions Markup Language (SAML) Version 2.0. Technical Overview 25 March 2008. OASIS Standard. https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf (2008)

[18] SKOS Simple Knowledge Organization System. http://www.w3.org/2004/02/skos/

[19] Tao, J., Sirin, E., Bao, J. and McGuinness, D. L.: Integrity Constraints in OWL, In *Proceedings of* the 24th AAAI Conference on Artificial Intelligence (AAAI-10), Atlanta, Georgia, USA, July 11-15 (2010)

[20] The FOAF Project. http://www.foaf-project.org/

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

[22] Veloudis, S., Paraskakis, I., Petsos, C.: Cloud Service Brokerage: Strengthening Service Resilience in Cloud-Based Virtual Enterprises. In Camarinha-Matos et al. (eds.) PRO-VE 2015. LNCS, vol 463, pp. 122--135, Springer, Heidelberg (2015)

[23] Veloudis, S., Verginadis, Y., Patiniotakis, I., Paraskakis, I., Mentzas, G.: Context-aware Security Models for PaaS-enabled Access Control. CLOSER Conference (2016)

[24] What's Hindering the Adoption of Cloud Computing in Europe?, 15 September 2015. [Online]. Available: https://blog.cloudsecurityalliance.org/2015/09/15/whats-hindering-the-adoption-of-cloud-computing-in-europe/

[25] WS-Trust 1.3. 19 March 2007. OASIS Standard. http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.doc (2007)