Context-aware Security Models for PaaS-enabled Access Control

Simeon Veloudis[1], Yiannis Verginadis[2], Ioannis Patiniotakis[2], Iraklis Paraskakis[1] and Gregoris Mentzas[2]

[1]*South East European Research Centre (SEERC), International Faculty of the University of Sheffield, City College, 24Prox. Koromila St., 54622, Thessaloniki, Greece*

[2]*Institute of Communications and Computer Systems, National Technical University of Athens, Athens, Greece*

{sveloudis, iparaskakis}@seerc.org, {jverg, gmentzas}@mail.ntua.gr

Keywords: Context-aware security, Ontologies, Access Control, Data privacy, Security by design

Abstract: Enterprises are embracing cloud computing in order to reduce costs and increase agility in their everyday business operations. Nevertheless, due mainly to confidentiality, privacy and integrity concerns, many are still reluctant to migrate their sensitive data to the cloud. In this paper, firstly, we outline the construction of a suitable Context-aware Security Model, for enhancing security in cloud applications. Secondly, we outline the construction of an extensible and declarative formalism for representing policy-related knowledge, one which disentangles the definition of a policy from the code employed for enforcing it. Both of them will be employed for supporting innovative PaaS-enabled access control mechanisms.

1. INTRODUCTION

Adopting the cloud computing paradigm means that an enterprise's IT environment is eventually transformed into a matrix of interwoven infrastructure, platform and application services which are delivered from diverse service providers (NIST, 2011). The cloud services that an enterprise will come to depend on will span not only different technologies and geographies, but most importantly, entirely different domains of ownership and control, making the strategic and operational management of the enterprise cloud environment a particularly challenging assignment. Nevertheless, enterprises increasingly recognize the compelling economic and operational benefits of cloud computing (Micro, 2010). Virtualizing and pooling IT resources in the cloud enables organisations to realize significant cost savings and accelerates deployment of new applications, simultaneously transforming business and government at an unprecedented pace (Group, 2013). Regardless of the differences in the figures reported with respect to the size of the cloud computing market or its future prospects, analysts agree on the view that the adoption of cloud computing is advancing at an ever-increasing pace (Cisco, 2011) and that it introduces a new economy-based paradigm (Vaquero et al., 2008). At the same time, however, it creates new security vulnerabilities stemming mainly from the fact that corporate data reside in externally controlled servers or untrusted cloud providers. Exploiting these vulnerabilities may result in data confidentiality and integrity breaches (CSA, 2013).

Evidently, these valuable business benefits cannot be realised without addressing the data security challenges introduced by cloud computing (Verginadis et al., 2015a). A promising approach to alleviating the security concerns associated with cloud computing is to assist application developers in defining effective security controls for the sensitive data of their cloud applications. To this end, in (Verginadis et al., 2015a) we proposed a generic security-by-design framework, essentially a PaaS solution that includes capabilities for guiding developers through the process of defining appropriate access control policies for safeguarding their sensitive data. In order to provide such

This paper has been published in *Jorge Cardoso, Donald Ferguson, Víctor Méndez Muñoz, and Markus Helfert (Eds.) Proceedings 6th International Conference on Cloud Computing and Services Science (CLOSER 2016) Vol. 1 and 2, Rome, Italy, April 23-25* (pp. 202–212). SCITEPRESS - Science and Technology Publications. DOI: https://doi.org/10.5220/0005918602020212

capabilities, such a generic framework bears two seminal characteristics. Firstly, it hinges upon an adequate access control scheme, one that takes into account the inherently dynamic and heterogeneous nature of cloud environments. Secondly, it captures the knowledge that lurks behind such a scheme (e.g. actions, subjects, locations, environmental attributes, etc.) using a generic and extensible formalism, one which can be tailored to the particular needs of different cloud applications. The first characteristic calls for the incorporation of the notion of context in access control policies, i.e. the consideration of dynamically-changing contextual attributes that may characterise data accesses. It therefore involves the development of a re-usable and generic Context-aware Security Model which goes beyond the traditional context-insensitive security (e.g. DAC, MAC, RBAC (Ferrari, 2010)). The second characteristic calls for the adoption of a declarative approach to modelling policy-related knowledge, one which is orthogonal to the code of any particular cloud application and which can be easily adapted to suit the needs of any such application.

The aim of this paper is twofold. On the one hand, it outlines the construction of a suitable Context-aware Security Model, one which essentially supports an Attribute-based Access control (ABAC) model (Hu et al., 2014). On the other hand, it outlines the construction of an extensible and declarative formalism for representing policy-related knowledge, one which disentangles the definition of a policy from the code employed for enforcing it, bringing about the following advantages: (i) it allows the policy-related knowledge to be extended and instantiated to suit the needs of a particular application, independently of the code employed by the application; (ii) it forms an adequate basis for reasoning generically about the correctness and consistency of the security policies, hence about the effectiveness of the security controls that these policies give rise to.

The rest of this paper is organised as follows. In Section 2, we elaborate on a context-aware security model that will be used as an underlying vocabulary for describing access control policies. In Section 3, we introduce a policy model that allows for the semantic description of PaaS-enabled access controls. In Section 4, we briefly discuss relevant work and in Section 5 we conclude the paper by presenting the next steps for the implementation and evaluation of the proposed approach.

2. CONTEXT-AWARE SECURITY MODEL

In this section, we present a context-aware access model, which can be used by the developers in order to annotate database Entities, Data Access Objects (DAO) or any other web endpoints that give access to sensitive data managed by cloud applications. This context model conceptualises the aspects, which must be considered during the selection of a data-access policy. These aspects may be any kind of information which is machine-parsable (Dey 2001); indicatively they may include the user's IP address and location, the type of device that s/he is using in order to interact with the application as well as his/her position in the company. These aspects can be interpreted in different ways during the security policy enforcement. In particular, the context aware access model can set the basis for determining which data is accessible under which circumstances.

2.1 Context-aware Security Meta-Model

In Figure 1, we present a meta-model that captures the main facets of the Context-aware Security Model along with their associations. Specifically, this model comprises of two different kinds of facets that may give rise to:

- Dynamic security controls – These controls grant or deny access to sensitive data on the basis of dynamically-evolving contextual attributes which are associated with the entity requesting the access. The relevant model facets are:
    - *Security Context Element*
    - *Permission*
    - *Context Pattern*
- Static security controls - These controls are independent of any dynamically evolving contextual attributes. They mainly correspond to the distribution and cryptographic protection features that certain data artefacts must have. The relevant model facet is the:
    - *Data Distribution and Encryption Element (DDE)*

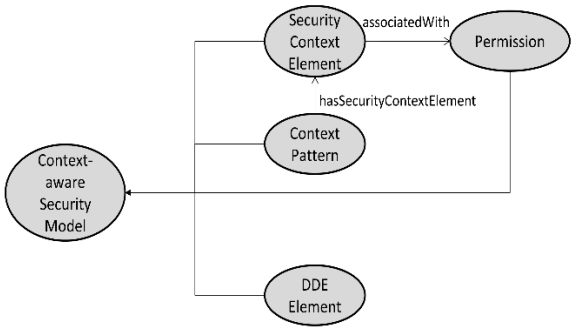

Figure 1: Context-aware security meta-model.

2.2 Context Model Facets

This section provides an elaboration of the initial set of facets that have been included in the part of the model that gives rise to dynamic security controls. We note that all these model facets are focused on the aspects relevant to access control for cloud services.

2.2.1 Security Context Element

The `Security Context Element` refers to the following five top-level concepts:

- `Location` - This class describes a physical and/or a network location where data are stored or from which a particular entity is requesting to access data.
- `DateTime` - This class describes the specific chronological point expressed as either instant or interval that characterises an access request (extends `owl-time:TemporalEntity`).
- `Connectivity` - This class captures the information related to the connection used by the

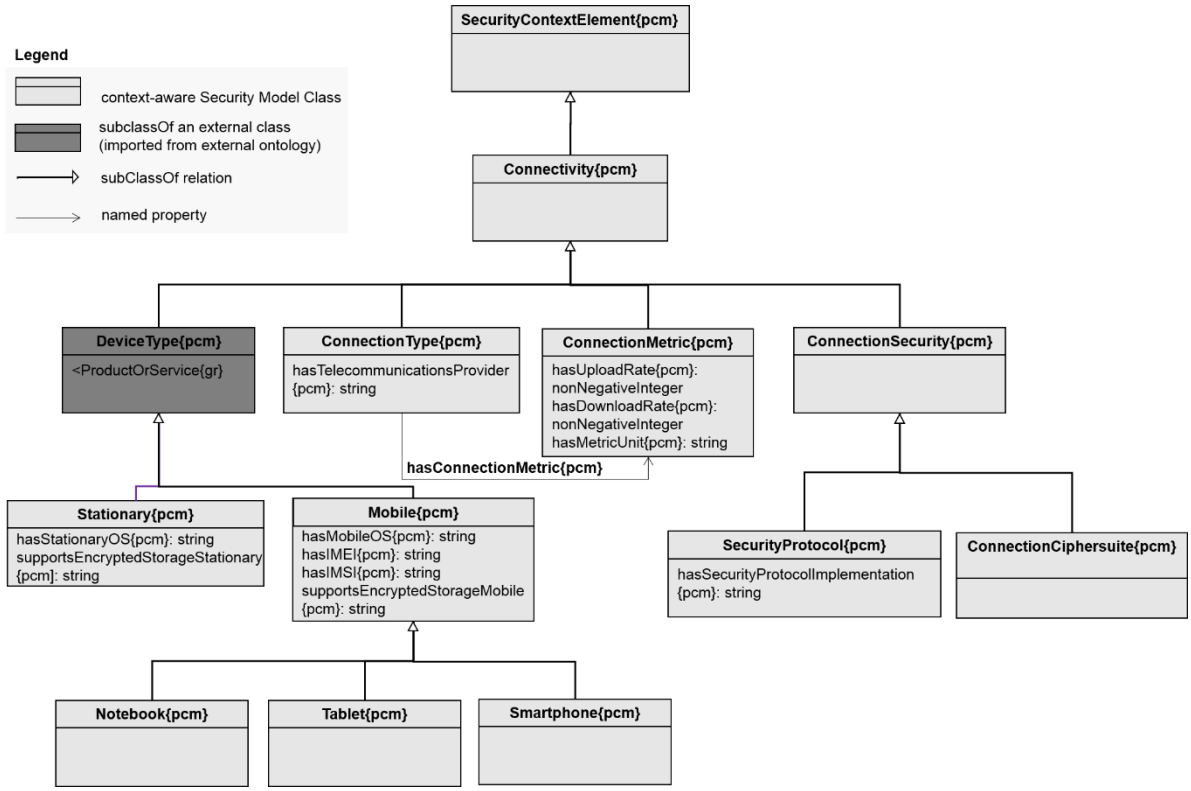

Figure 2: UML Class diagram for the Connectivity context element.

According to this meta-model, instances of these aforementioned facets formulate the `Context-aware Security Model`. Furthermore, `Context Pattern` elements are directly associated to `Security Context Elements` (through the `hasSecurityContextElement` property) in order to be defined, while the latter can be associated with certain `Permission` elements. Due to space limitations we discuss only the context model facets that are relevant to access control.

Subject for accessing sensitive data (see Figure 2).

- Object - This class refers to any kind of artefacts that should be protected based on their sensitivity levels. These artefacts may refer to (non-) relational data, files, software artefacts that manage sensitive data or even infrastructure artefacts used.
- Subject - An instance of this class represents the agent seeking access to a particular data artefact. This can be an organization, a person, a group or a service (extends foaf:Agent, goodrelations:BusinessEntity, goodrelations:ProductOrService).

In Figure 2, we provide further details regarding the Connectivity top level concept that include subclasses, imported or extended external classes, data and object properties. The identifier pcm (stands for PaaS Control Model) recognises the namespace underlying the classes and properties of the proposed vocabulary. Due to space limitations the details of all the top level concepts are not explained in this paper but they are available in the following URL: *http://imu.ntua.gr/software/context-aware-security-model*.

2.2.2 Context Pattern

The next facet of this model is the Context Pattern model that includes the following top-level concepts:

- Location pattern - It refers to recurring motives of data accesses that are recognized with respect to the Location context element.
- DateTime pattern - It refers to recurring motives of data accesses that are recognized with respect to the DateTime context element.
- Connectivity pattern - It refers to recurring motives of data accesses that are recognized with respect to the Connectivity context element.
- Object pattern - It refers to recurring motives of data accesses that are recognized with respect to the Object context element.
- Permission pattern - It refers to recurring motives of data accesses that are recognized with respect to the Permission element.
- Access Sequence Pattern - It refers to data accesses that are recognized by any preceding access actions made by a particular Subject (extends Kaos:AccessAction).

For the above vocabulary we use the identifier pcpm (stands for PaaS Context Pattern Model) for recognising the respective namespace of underlying classes and properties.

2.2.3 Permission

Another important facet is the Permission model that involves the following top-level concepts:

- Data Permission - This class refers to any action allowed by a Subject upon a data entity (extends schema.org:Action)
- DDL Permission - This class reveals the data definition language (DDL) related actions on a specific Object.

The Data Permission involves four subclasses:

- Datastore Permission – It describes any action allowed by a Subject upon a data entity in a datastore (e.g. Search, List, Select, Insert, etc.)
- File Permission - It describes any action allowed by a Subject upon a file (e.g. Read, ChDir, Move, Delete, etc.)
- WebEndpoint Permission – It describes any web endpoint related action that is allowed upon a data artefact (e.g. Get, Put, Post, Delete).
- Volume Permission - It refers to any access permission to a dedicated infrastructure artefact.

The DDL Permission involves two subclasses:

- Datastore DDL Permission – It describes any DDL related permission on a datastore (e.g. Create, Alter, Drop).
- File System Structure Permission - It describes any DDL related permission on a file (e.g. CreateDir, RenameDir, CopyDir, DeepCopyDir, ChOwner, etc.).

For the above vocabulary we use the identifier ppm (stands for PaaS PaaS Permission Model) for recognising the respective namespace of underlying classes and properties.

In Section 3, we demonstrate the way that these contextual elements that give rise to dynamic security controls, can set the basis for developing a policy model for paas-enabled access control.

3. POLICY MODEL FOR PAAS-ENABLED ACCESS CONTROL

Three are the main types of security policy that the proposed PaaS solution aims at supporting:

- Data encryption policies. These determine the strength of the cryptographic protection that each sensitive object enjoys for confidentiality reasons. They give rise to security controls enforceable during bootstrapping of a cloud application.
- Data fragmentation and distribution policies. These determine the manner in which sensitive data objects must be fragmented and distributed to different physical servers for privacy reasons. They too give rise to security controls enforceable during application bootstrapping.
- Access control policies. These are essentially ABAC policies that determine when to grant, or deny, access to sensitive data on the basis of dynamically-evolving contextual attributes associated with the entity requesting the access. Context awareness is deemed of utmost importance for leveraging the security of cloud-based applications which by definition operate in dynamic and heterogeneous environments. Access control policies give rise to security controls dynamically enforceable during application execution time.

Due to space limitations, in this paper we only consider access control policies.

3.1 Access Control Policy Model

We argue that, in order to aid application developers in defining effective ABAC policies for *any* kind of sensitive data, our PaaS solution must be underpinned by an underlying *ontological model*, one which bears the following characteristics:

- It is founded on a framework of relevant interrelated concepts which capture all those knowledge artefacts that are required for describing an ABAC policy. Such a framework is provided by the vocabulary outlined in Section 2
- It uses an extensible formalism for accommodating the framework of interrelated concepts, hence expressing ABAC policies. Such a representation disentangles the definition of a policy from the code employed for enforcing it, offering the following seminal advantages: (i) It allows the framework of relevant interrelated concepts to be extended and instantiated, independently of the code employed by the application. Such an extension/instantiation aims at customising the framework to the particular needs of a given application. (ii) It forms an adequate basis for reasoning generically about the correctness and consistency of the ABAC policies, hence about the effectiveness of the security controls that these policies give rise to.

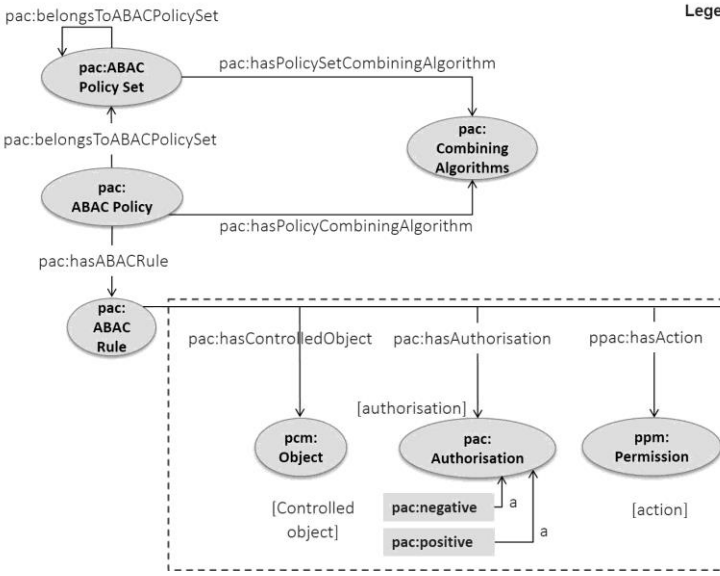

Figure 3: ABAC ontological model

3.1.1 ABAC Policy Rules

Following an approach inspired by the XACML standard (OASIS, 2013), an ABAC policy comprises one or more *rules*. A rule is the most elementary structural element and the basic building block of policies. A generic template for ABAC rules is provided in Table 1:

Table 1: ABAC rule template.

[actor**]** with **[**context expression**]** has **[**authorisation**]** for **[**action**]** on **[**controlled object**]**

The template defines a generic structure, in terms of relevant attributes, to which all ABAC rules in our PaaS framework adhere. It comprises several attributes which are further elaborated below.

- *actor* identifies the subject who may request access to perform an operation on a sensitive object; it draws its values from the

pcm:Subject class of the Security Context Element model defined in Section 2.

- *context expression* is a Boolean expression which identifies the environmental conditions that must hold in order to permit, or deny, the performance of an operation on a sensitive object. Context expressions are further elaborated in Section 3.1.2.
- *authorisation* determines the type of authorisation (positive i.e. 'permit', or negative i.e. 'deny') that is granted.
- *action* identifies the operation that may, or may not, be performed on the protected sensitive object; it draws its values from the ppm:Permission class of the Security Context Element model defined in Section 2.
- *controlled object* identifies the sensitive object on which access is requested; it draws its values from the pcm:Object class of the Security Context Element model defined in Section 2.

In our ontological model, an ABAC rule takes the form of an instance of the class pac:ABACRule (see Figure 3). A number of object properties are attached to this class which are intended to capture the aforementioned attributes. As depicted in Figure 3, these associate the pac:ABACRule class with an appropriate framework of relevant classes from the vocabulary of Section 2 which adequately capture the attributes of the ABAC rule template. The identifier pac (stands for PaaS Access Control) recognises the namespace underlying the classes and properties of the proposed ontological model.

3.1.2 Context Expressions

A context expression takes the form of an instance of the class pac:ContextExpression (see Figure 3). It specifies a number of constraints on the values of one or more instances drawn from the vocabularies pcpm:ContextPattern and pcm:SecurityContextElement defined in Section 2. The class pac:ContextExpression is associated with these vocabularies through the object properties pac:hasPatternParameter and pac:hasParameter respectively depicted in Figure 4. As we would expect, a context expression may combine two or more constraints using logical connectives (conjunction, disjunction, exclusive disjunction, negation). In order to capture such combinations of constraints, the pac:ContextExpression class encompasses a subclass for each logical connective (see Figure 4). A context expression may be defined recursively, in terms of one or more other context expressions. This is captured by associating the pac:ContextExpression class with itself through the properties pac:hasParameter and pac:hasPatternParameter (see Figure 4).

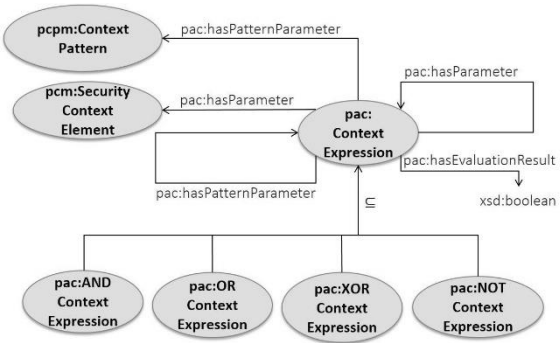

Figure 4: Context expression ontological model.

3.1.3 ABAC Policies and Policy Sets

In our ontological model, an ABAC policy takes the form of an instance of the class pac:ABACPolicy. It is associated with the rules that it comprises through the property pac:hasABACRule. An ABAC policy may comprise a multitude of ABAC rules which potentially evaluate to different (and conflicting) access control decisions. This calls for a *combining algorithm* which reconciles the different decisions and determines an overall decision for the entire policy (OASIS, 2013). An example of a combining algorithm is the 'deny-overrides' algorithm, whereby a policy evaluation resolves to 'deny' if at least one of its constituent rules evaluates to 'deny', or if none of them evaluates to 'permit'. A combining algorithm takes the form of an instance of the class pac:CombiningAlgorithms depicted in Figure 3. A combining algorithm is attached to an ABAC policy through the property pac:hasPolicyCombiningAlgorithm.

Following an approach inspired by the XACML standard (OASIS, 2013), access control policies are grouped into *policy sets*. In our ontological model, a policy set takes the form of an instance of the class pac:ABACPolicySet (see Figure 3). A policy is associated with its enclosing policy set through the property pac:belongsToABACPolicySet. A policy set may exhibit a hierarchical structure and comprise one or more other ABAC policy sets. This recursive inclusion is captured by rendering the pac:belongsToABACPolicySet property applicable to ABAC policy sets too (see Figure 3).

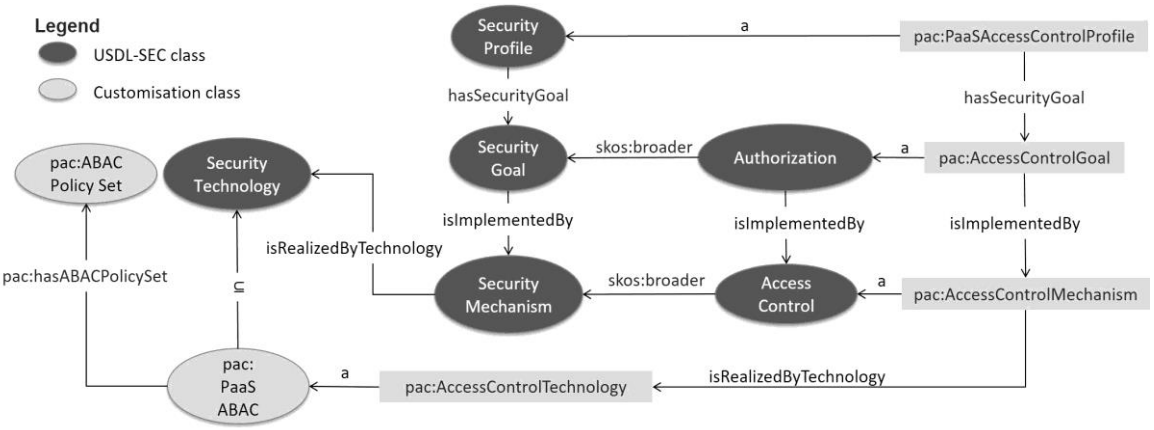

Figure 5: USDL-SEC customisation (only classes and properties used in this paper are depicted).

ABAC policy sets are also associated with combining algorithms. As in the case of policies, these reconcile the potentially different access control decisions to which the policies comprising a policy set may evaluate.

It is to be noted here that analogous policy models have been devised for the rest of the policy types outlined at the beginning of Section 3.

3.2 Access Control Policies in Linked USDL

Section 3.1 outlined a model for the generic representation of ABAC policies. This section demonstrates how this model can be incorporated into the ontological framework provided by Linked USDL (2014), and in particular, into USDL-SEC – Linked USDL's security profile (USDL stands from Unified Service Description Language). By capitalising on USDL-SEC, our approach avoids the use of bespoke, non-standards-based, ontologies for the representation of ABAC policies (see Section 4 for a relevant outline of such ontologies). Instead, it is based on a diffused ontological framework which has recently attracted considerable research interest. In addition, the adoption of Linked USDL brings about the following advantages (Pedrinaci et al., 2014): (i) Linked USDL relies on existing widely-used RDF(S) vocabularies (such as GoodRelations, FOAF and SKOS), whilst it can be easily extended through linking to further existing, or new, RDF(S) ontologies. In this respect, it promotes knowledge sharing whilst it increases the interoperability, reusability and generality of our framework. (ii) By offering a number of different profiles, Linked USDL provides a holistic and generic solution able to adequately capture a wide range of business details. This is important for our work as it allows us to adequately capture the business aspects of the security policies encountered within our framework. (iii) Linked USDL is designed to be easily extensible through linking to further existing, or new, RDF(S) ontologies. This is particularly important for our model as it facilitates seamless integration with the Context-aware security model devised in Section 2. (iv) It provides ample support for modelling, comparing, and trading services and service bundles. It also provides support for specifying, tracking, and reasoning about the involvement of entities in service delivery chains. This is important for our work for it allows comparisons to be drawn between different policy models that may potentially be offered through our framework.

Due to space limitations, an introduction to the classes and properties offered by Linked USDL is omitted here. The interested reader is referred to (Linked USDL, 2014).

3.2.1 Incorporating ABAC Policies into USDL-SEC

USDL-SEC provides a simple vocabulary for describing the security properties of an application. It introduces the classes `SecurityProfile`, `SecurityGoal`, `SecurityMechanism`, and `SecurityTechnology`, along with a number of relevant object properties, as depicted in Figure 5 (to reduce notational clutter, we avoid prefixing the `usdl-sec` namespace to USDL-SEC classes and properties). For a more complete discussion of the classes and properties offered by USDL-SEC the reader is referred to (Linked USDL, 2014).

At the highest level of abstraction, the ABAC policy model forms, essentially, a particular *security profile* to which a cloud application may adhere. In this respect it is modelled as an instance of USDL-SEC's `SecurityProfile` class, namely `pac:PaaSAccessControlProfile`. A security profile is associated, through the object property `hasSecurityGoal`, with one or more *security goals* from the USDL-SEC class `SecurityGoal`. In the case of ABAC policies, the security goal is authorisation. This is modelled in Figure 5 by associating the instance `pac:PaaSAccessControlProfile` with an instance, say `pac:AccessControlGoal`, of the `Authorization` class through the property `hasSecurityGoal`. The `Authorization` class forms a sub-concept of `SecurityGoal`.

The authorisation goal is achieved by means of a suitable *access control mechanism*. USDL-SEC provides a layer of abstraction, namely the concept `SecurityMechanism`, for the specification of such a mechanism. In particular, it provides the class `AccessControl`, a sub-concept of `SecurityMechanism`, an instance of which, say `pac:AccessControlMechanism`, represents the access control mechanism offered by our PaaS framework. This instance is associated with the `pac:AccessControlGoal` instance through the property `isImplementedBy`.

The access control mechanism represented by the instance `pac:AccessControlMechanism` is realised by means of some underlying concrete security technology. USDL-SEC provides a layer of abstraction, namely the concept `SecurityTechnology`, for the specification of such a technology. In our model, the access control mechanism is realised by the access control technology provided by our PaaS framework. This is modelled by introducing the `pac:PaaSABAC` subclass (see Figure 5), along with the instance `pac:AccessControlTechnology` which represents this access control technology. This instance is associated with the access control mechanism through the property `isRealizedByTechnology` (see Figure 5). The `pac:PaaSABAC` subclass is associated, through the property `pac:hasABACPoliceSet`, with the class `pac:ABACPolicySet` (the top concept of the ABAC policy model of Section 3). This essentially captures the fact that the access control mechanism is realised through the policies encompassed in one or more ABAC policy sets.

It is to be noted here that the policy models devised for the rest of the policy types outlined at the beginning of Section 3 are incorporated into USDL-SEC in an analogous manner.

4. RELATED WORK

In the literature, there is a plethora of context models. For example (Strang & Linnhoff-Popien, 2004) and (Bettini et al., 2010) review models of context that range from key-value models, to mark-up schemes, graphical models, object-oriented models, logic-based models and ontology-based models. An interesting context model is the one proposed in (Miele et al., 2009), which was initially developed for mobile devices and later extended for the use in service-based applications in (Bucchiarone et al., 2010). Another example is the one in (Truong et al., 2009) who developed an ontological model of the W4H classification for context. The W4H ontology provides a set of general classes, properties, and relations exploiting the five semantic dimensions: identity (who), location (where), time (when), activity (what) and device profiles (how). Furthermore, authors exploited the concepts of the W4H ontology by including domain-independent common context concepts from existing work; e.g. FOAF, vCard, the OWL-Time Ontology, etc. The five dimensions of context have been also pointed out earlier by Abowd and Mynatt (Abowd and Mynatt, 2000) who stated that context should include the 'five W': Who, What, Where, When, and Why. For example, by 'Who', they mean that it is not enough to identify a person as a customer; the person's past actions and service related background should also be identified for better service provision. 'What' refers to the activities conducted by the people involved in the context and interactions between them. 'Where' represents location data. 'When' is related to time. 'Why' specifies the reason for 'Who' did 'What'. 'Why' represents a complicated notion and acts as the driving force for context sensitive information systems. In addition to that, from the literature review we found interesting efforts that concerned modelling languages, which take context explicitly into account. The first such effort was ContextUML a UML-based modelling language that was specifically designed for Web service development and applies model-driven development principles; see (Sheng, 2005). In a Web-service-based environment, ContextUML considers that context contains any information that can be used by a Web service to adjust its execution and output.

The need for the exploitation of context in the access control mechanisms is quite evident from the state-of-the-art. Nevertheless, we found that even dedicated context-aware extensions to traditional access control models (e.g. Role-based Access Control - RBAC) either do not cover all the contextual elements with a reusable security related context model or are proven hard to maintain in dynamic environments where users often change roles or are not known a priori (Heupel, 2012). On the other hand, pure ontological models (e.g. (Truong et al., 2009), or even Attribute-based Access Control (ABAC) approaches (e.g. (Jung et al., 2014)) they do not seem to cover all the security requirements associated with the lifecycle of a cloud application (i.e. bootstrapping and run-time). Specifically, either they do not cover the full range of contextual elements that are associated with all the security aspects of sensitive data managed by cloud applications or they are based on heavy inferencing that is considered as inefficient for such dynamic environments (Verginadis et al., 2015b).

With respect to policies and policy-based applications, syntactic descriptions promote a declarative approach to policy expression, one which aims at replacing a trend whereby policies are encoded imperatively, as part of the same software that checks for their compliance. Several markup languages have been proposed for the declarative description of policies, some prominent examples being RuleML (2015), XACML (OASIS, 2013), SAML (2008) and WS-Trust (2007). These generally provide XML-based syntaxes for expressing policy rules and sets. Nevertheless, such syntactic descriptions fail to capture the knowledge lurking behind policies. In this respect, they are merely data models that lack any form of semantic agreement beyond the boundaries of the organisation that developed them. Any interoperability relies on the use of vocabularies that are shared among all parties involved in an interaction.

In order to overcome the aforementioned limitations, semantically-rich approaches to the specification of policies have been brought to the attention of the research community. These generally embrace Semantic Web representations for capturing what we term action-oriented policies, i.e. policies which control when a particular actor or subject can perform a specified action on, or through the use of, a particular resource. These approaches typically employ ontologies in order to assign meaning to actors, actions and resources. Several works in the area of semantic policy representation have been reported in the literature (Uszok, 2005; Kagal et al., 2003; Hu et al., 2014). In (Uszok, 2005), the authors presented KAoS – a general-purpose policy management framework which exhibits a three-layered architecture comprising:

- A human interface layer, which provides a graphical interface for policy specification in natural language.
- A policy management layer, which uses OWL (2004) to encode and manage policy-related knowledge.
- A policy monitoring and enforcement layer, which automatically grounds OWL policies to a programmatic format suitable for policy-based monitoring and policy enforcement.

In (Kagal et al., 2003) the authors proposed Rei – a policy specification language expressed in OWL-Lite (2004). It allows the declarative representation of a wide range of policies which control which actions can be performed, and which actions should be performed, by a specific entity. Furthermore, it defines a set of concepts (rights, prohibitions, obligations, and dispenations) for specifying and reasoning about access control rules. In this respect, it provides an abstraction which allows the specification of a desirable set of behaviours which are potentially understandable – hence enforceable – by a wide range of autonomous entities in open and dynamic environments.

In (Hu et al., 2014), the authors recognise that cloud computing, and in particular the concept of multi-tenancy, calls for policy-driven access control mechanisms. They propose an ontology-based framework to capture the common semantics and structure of different types of access control policies (e.g. XACML policies, firewall policies, etc.), and facilitate the process of detecting anomalies in these policies. Their ontology captures the underlying domain concepts involved, the policy structure and the policy attributes. Particular types of access control policies are obtained by appropriately instantiating the ontology.

5. CONCLUSIONS

We have presented suitable vocabularies of concepts and properties, namely the Security Context Element, the Context Pattern and the Permission which adequately captures the knowledge lurking behind ABAC policies. We have also proposed a generic ontological model for the abstract representation of ABAC policies which disentangles the definition of a policy from the actual code employed for enforcing it, bringing about the advantages outlined in Section

3.1. The model is underpinned by the Security Context Element vocabulary, and is incorporated into the ontological framework offered by USDL-SEC (Linked USDL's security profile). Such a model forms the basis of our proposed PaaS solution – essentially a security-by-design framework which aims at aiding cloud application developers in defining effective access control policies for any kind of sensitive data.

Any effective use of the ABAC policy model requires a mechanism through which it can be suitably customised in order to allow for the specification of concrete ABAC policies. Such a customisation amounts to an extension and/or instantiation of the abstract classes and properties presented in Sections 3 and 4. It is the responsibility of such a mechanism to ensure that this extension/instantiation takes place according to a set of predefined *governance policies*. In the future, we intend to investigate the construction of a higher-level ontological framework that will generically accommodate these governance policies and thus pave the way for the construction of a *generic* customisation mechanism that can be easily adapted to the particular needs of the potential adopter of our framework.

ACKNOWLEDGEMENTS

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. The authors would like to thank the partners of the PaaSword project (www.paasword.eu) for their valuable advices and comments.

REFERENCES

Abowd, G., & Mynatt, E., 2000. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction (TOCHI)* - Special issue on human-computer interaction in the new millennium, 29-58.

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., & Riboni, D., 2010. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 161-180.

Bucchiarone, A., Kazhamiakin, R., Cappiello, C., Nitto, E., & Mazza, V., 2010. A context-driven adaptation process for service-based applications. In *ACM Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems (PESOS'10)*, pp. 50-56, Cape Town, South Africa.

Cisco, 2011. Cloud: What an Enterprise Must Know, Cisco White Paper.

CSA, 2013. *The Notorious Nine. Cloud Computing Top Threats in 2013*. Cloud Security Alliance.

Dey, A. K., 2001. *Understanding and Using Context.* In Personal and Ubiquitous Computing Journal, vol. 5, no. 1, p. 4-7.

Ferrari, E., 2010. *Access Control in Data Management Systems.* Synthesis Lectures on Data Management, Morgan & Claypool, Vol. 2, No. 1, p. 1-117.

Group, T. T., 2013. The Notorious Nine. Cloud Computing Top Threats in 2013. Cloud Security Aliance (CSA).

Heupel, M., Fischer, L., Bourimi, M., Kesdogan, D., Scerri, S., Hermann, F., Gimenez, R., 2012. Context-Aware, Trust-Based Access Control for the di.me Userware. In *Proceedings of the 5th International Conference on New Technologies, Mobility and Security (NTMS'12)*, pp. 1-6, Istanbul, Turkey, IEEE Computer Society.

Hu, V. C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller R., and Scarfone K., 2014. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. NIST.

Hu, H., Ahn, G.-J. and Kulkarni, K., 2011. Ontology-based policy anomaly management for autonomic computing. *In 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*.

Jung, C., Eitel, A., Schwarz, R., 2014. Cloud Security with Context-aware Usage Control Policies. In *Proceedings of the INFORMATIK'14 Conference*, pp. 211-222.

Kagal, L., Finin, T. and Joshi, A., 2003. A Policy Language for a Pervasive Computing Environment. In *4th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY '03).*

Linked USDL, 2014. Available online: http://linked-usdl.org/.

Micro, T., 2010. The Need for Cloud Computing Security. Trend Micro.

Miele, A., Quintarelli, E., Tanca, L., 2009. A methodology for preference-based personalization of contextual data. In *ACM Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (EDBT'09)*, pp. 287-298, Saint-Petersburg, Russia.

NIST, 2011. Cloud Computing Reference Architecture, National Institute of Standards and Technology.

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

OWL Web Ontology Language Reference. W3C Recommendation, 2004. Available online: http://www.w3.org/TR/owl-ref/.

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

Specification of Deliberation RuleML 1.01, 2015. Available online:

http://wiki.ruleml.org/index.php/Specification_of_Deliberation_RuleML_1.01.

Security Assertions Markup Language (SAML) Version 2.0. Technical Overview, 2008. Available online: https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf

Sheng, Q., & Benatallah, B., 2005. ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services Development. In *Proceedings of the International Conference on Mobile Business (ICMB'05)*, pp. 206-212, IEEE Computer Society.

Strang, T., Linnhoff-Popien, C., 2004. A Context Modeling Survey. *In Workshop on Advanced Context Modelling, Reasoning and Management, (UbiComp'04)* - The Sixth International Conference on Ubiquitous Computing. Nottingham, England.

Truong, H.-L., Manzoor, A., Dustdar, S., 2009. On modeling, collecting and utilizing context information for disaster responses in pervasive environments. In *ACM Proceedings of the first international workshop on Context-aware software technology and applications (CASTA'09)*, pp. 25-28, Amsterdam, The Netherlands.

Uszok, A., Bradshaw, J., Jeffers, R., Johnson, M., Tate, A., Dalton, J. and Aitken, S., 2005. KAoS Policy Management for Semantic Web Services. *IEEE Intel. Sys.*, vol. 19, no. 4, pp. 32 - 41.

Vaquero, L.M., Rodero-Merino, L., Caceres, J. and Lindner, M., 2008. A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev., vol 39, no 1, pp. 50 — 55*.

Verginadis, Y., Michalas, A., Gouvas, P., Schiefer, G., Hübsch, G., Paraskakis, I., 2015a. PaaSword: A Holistic Data Privacy and Security by Design Framework for Cloud Services. *Proceedings of the 5th International Conference on Cloud Computing and Services Science (CLOSER 2015)*, May 20-22, Lisbon, Portugal.

Verginadis, Y., Mentzas, G., Veloudis, S., Paraskakis, I., 2015b. A Survey on Context Security Policies. In *Proceedings of the 1st International Workshop on Cloud Security and Data Privacy by Design (CloudSPD'15)*, co-located with the 8th IEEE/ACM International Conference on Utility and Cloud Computing, Limassol, Cyprus, December 7-10.

WS-Trust 1.3, 2007. Available online: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.doc.