D. Hühnlein, H. Roßnagel, C. Schunck, M. Talamo (Eds.): Open Identity Summit 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 181

Aligning ABAC Policies with Information Security Policies using Controlled Vocabulary

Raik Kuhlisch[1] and Sören Bittins[2]

Abstract: Attribute-based Access Control (ABAC) policies are based on mutually processable policy attributes. Assigned permissions in such policies need to be reflected or combined with organisational constraints. Best practice in information security dictates having the operational need to access a particular information artifact independent from the function of the specific application systems. Consequently, any policy regulating the behaviour towards information access must adhere to a minimum degree of mutual semantic expressiveness to be combined and processed with the matching ABAC policy. We show how to detect policy attribute conflicts between ABAC policies and information access policies by means of controlled vocabulary and Semantic Web technologies.

Keywords: policy conflict, access control, attribute-based access control (ABAC), information security, XACML

1 Introduction

An access control model defines how to evaluate authorisation decision or how to decide resource access requests in distributed computing environments. A security context anchors a user's roles and potential approvals to access protected resources [Be06, p. 18]. The paradigm of Attribute-based Access Control (ABAC) contributes to the appraisal and decision of requests by injecting additional attributes from all relevant information systems such as subject, object or environment conditions into the decision process. Consequently, a security context becomes more complex compared to other paradigms such as the user-centric Role-based Access Control (RBAC) [FK92] especially regarding the semantical expressivness, computability, and interoperability when policies of higher granularity must be evaluated for deciding on the legitimacy of a resource access request.

From the information security management point of view (e.g. ISO 27001 [In14]), information access is defined in or derived from an overarching information security policy [Ku13]. The specific access-regulating rules are implemented in more precise policies that capture the specifics of application systems and access contexts. Therefore, ABAC policy attributes need to be correlated with attributes from a variety of information security policies. Due to the use of atomic values in ABAC policies, interoperability issues may arise. This paper proposes a policy system that separates information access from access control rules in accordance with an information security management. Without proper means to relate ABAC policy attributes with other policies, inconsistencies or conflicts could occur.

[1] The University of Rostock, Faculty of Computer Science and Electrical Engineering,
Albert-Einstein-Str. 22, 18059 Rostock, Germany, raik.kuhlisch@uni-rostock.de
[2] Fraunhofer FOKUS, ESPRI – Collaborative Safety and Security
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany, soeren.bittins@fokus.fraunhofer.de

An integrated policy organisation mechanism is presented that addresses ABAC policy conflict identification through controlled vocabularies in conjunction with a conflict detection algorithm, includes a security policy poly-hierarchy with fitting-in ABAC policies.

Organisation. This paper is structured as follows: Section 2 discusses relevant previous work. Section 3 introduces background on policy-based information management and Semantic Web technology-based vocabulary management. The proposed ABAC policy conflict detection is outlined in Section 4, while Section 5 motivates a policy system that supports a policy alignment. Section 6 concludes and addresses future research.

2 Related Work

ABAC policies can be represented as eXtensible Access Control Markup Language (XACML) [Mo05] policies. Dersingh et al. [DLJ08] use knowledge representation and retrieval techniques to enhance XACML with context attributes. They extended the XACML framework with an ontology administration point that holds inferred knowledge about domain information associated with subjects, resources, actions, or environments. This work is aimed at a refinement of policies with semantic context attributes. However, their focus is rather on the evaluation of context-aware authorisation decisions than on policy conflict handling.

Conflict detection with ABAC policy has been investigated comprehensively. Zoo and Zhang [ZZ14] propose a conflict detection algorithm based on a binary sort tree. It fits into the XACML framework. Additional policies are registered through a policy administration point, a core component of the XACML framework. The detection algorithm is limited to XACML policies and does not take other kinds of security policies into account.

Shu, Yang and Arenas [SYA09] put forward conflict detection and handling among ABAC policy rules. Their notion of semantically-equivalent policies allows for matching policies with similar tuples (subject, object, action). However, an assumption is that these policies rely on fully comparable attribute predicates. This is true if policies share a common set of policy attribute values. External policy attributes cannot be readily compared.

We represent XACML policies in Semantic Web technologies in order to infer conflicts with other policies and propose to align ABAC policies with information security policies.

3 Preliminaries

3.1 Information Security Management Compliant Policy Hierarchy

Access control must ensure that access is constrained (e.g. through classification) to the minimal meaningful amount of permissions required to fulfill a legitimate operation in order to implement restricting security principles such as ”need-to-know”. Additionally, any permissions assigned to users must not be processed separately but assessed by all relevant decision workflows and must be computable by process-specific policies. They may be derived from security requirements as well, for instance, in [Sa15] a method engineering approach is shown that elicits permission sets from business processes. Nevertheless, authors should consider the following when authoring access control policies [Ca09]:

- What tasks support a dedicated information/software system?
- Who is allowed to perform these tasks?
- What kind of data an information/software system processes?
- What operations are defined on this data?

These questions define a specific application purpose and context. Additional transaction rules provide a context to requests for information resources and is more specific to a purpose of use. They describe a certain type of information that is processed in particular situations and implement the concept of functional roles where tasks have been assigned on the basis of existing expertise and routine organization of work. One means to develop access policies in a formal manner is to apply information security management, outlining a comprehensive handling of information disclosure according to an operational need. The most prominent framework is ISO/IEC 27001 [In15]. National equivalents exist (e.g. in Germany with the BSI Standard 100-1 [Bu08]). However, the most important thing is to apply a top-down approach (i.e. to break data activities into system specific permissions).

An adequate information security management system/approach should at least include information security objectives, measures, and controls. Access Control is just one measure in order to manage information security. However, the remainder of this document puts a strong focus on it—paper-based information processing is not examined.

We adapted the policy concepts from [So98, p. 222] as follows:

- An *information security policy* describes fundamental objectives for the protection of information in an organisation. It applies to all electronic and paper-based information processing methods. In order to achieve the fundamental objectives general principles and responsibilities are described.
- The part of an *information security policy* that describes legitimate accesses to protected information is defined as an *information access policy*. This policy defines the legitimate use of information (i.e. regulates the processing of information through purpose of use restrictions and who may access protected data under what circumstances) in terms of compliance, regulatory mandates as well as what data is protected (e.g. record types). Finally, it states employee's characteristics within an organization or organisational unit. It describes the constraints on the processing of data or might determine the purposes for which and the manner in which any data are to be processed.
- A *resource behaviour policy* controls resource access by taking into account the organisation of work and regulating the use of subsystems of an entire information/-software system. It holds a list of software services that realise a clear purpose of use. These services are implemented by automated subsystems of an information system (including belonging hardware, system software, communication equipment and the application system itself). The *resource behaviour policy* concludes with access types defined on the software services.
- A more detailed *ABAC policy* picks up access rights through permissions that are assigned to registered subjects. It may contain actions in the sense of a MUST or MUST NOT (i.e. obligations) or actions that are explicitly forbidden with authorized actions (refrain actions).

3.2 Policy and Policy Attribute Representation

A fundamental prerequisite for identifying policy conflicts among different types of policies is to rely on formal attribute relations. We formalize those relations through controlled vocabularies (also known as code systems or terminologies). One standard is the Common Terminology Services 2 (CTS2) by the Object Management Group (OMG) [Ob15] for defining structural and functional profiles over the entire lifecycle of concepts (i.e. codes).

We represent policy attributes through code systems and manage services, resources, actions etc. by a terminology server. In [Bi13] an implementation is described that runs a CTS2 instance with Semantic Web technologies. So-called Resource Description Framework (RDF) [Ha04] signatures with additional RDF Schema [BG14] axiomatic triples define structural constraints of a domain model with Frame Logic semantics [KLW95]. This ontological approach fits to well-established *denyAll* default behaviors in access control policies and provides means to reason about processing purposes. That means, we adapt the implementation from [Bi13] in order to provide a policy representation with ontologies. This requires the definition of policy signatures in RDF. The UML-like diagram in Figure 1 shows a rough outline of defined RDF classes and properties that represent policy types (multiplicity is derived from the Extended Backus-Naur Form notation). For the sake of clarity, no ABAC policy is shown. String patterns denote the use of code systems.

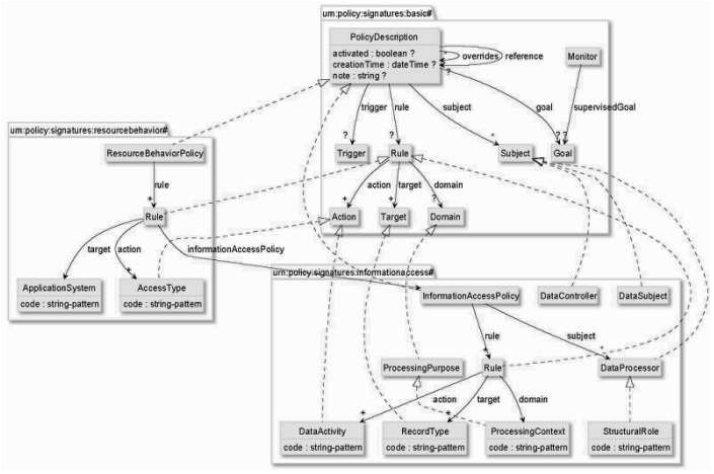

Fig. 1: Policy information model with relations

If all policies are stored within the same triple store, codes or attribute values can easily be referenced with CTS2 class concepts. However, a small list of acceptable codes might be represented as well with regular expressions (notion of facets in Frame Logic semantics). A disadvantage is the tight binding between policy and attribute values which makes a code system modification more difficult (forces an update of a policy signature). The following snippet in RDF Turtle syntax [Be14] shows the application of ”inline codes”.

```
@prefix:      <urn:policy:signatures:informationaccess#> .
@prefix sig:  <urn:negros:signatures#> .
```

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

:StructuralRole
  a rdfs:Class ;
  rdfs:comment """Competence of an employee within an
        organisation or organisation unit. """@en ;
  rdfs:subClassOf :DataProcessor ;
  sig:propertyConstraint [
    sig:onProperty prop:code ;
    sig:range xsd:string;
    sig:min 1 ;
    sig:max 1 ;
    sig:facet '(ApplicationAdministrator)|(OfficeClerkOrAssistant)' ;
  ] .
```

List. 1: RDF signature for structural role values

A a shared triple store utilizing CTS2 terminology layer benefits from:

- strict separation of policy structure and policy attribute maintenance,
- support for any designations of policy attribute values,
- support for domain-specific attribute value bags through value list semantics.

3.3 ABAC Policy Model

An ABAC policy regulates access requests by subjects onto information objects through permissions in an information system. Usually, access control policies express authorisation rules with a quadruple (subjects, objects, actions, authorisation decisions). Other notions of access control policies include obligations and refrain actions. The single difference is the modality of the policy, i.e. which basic statement the policy defines (grant or deny). However, an authorisation decision can rely on non-matching policies as well which should usually cause a deny decision.

ABAC policies can be represented with industry standards such as the declarative XACML [Mo05] that supports Next Generation Access Control [Am13] being capable of handling different policies defined by distributed systems. XACML also supports authorisation statements and obligations but has poor semantic expressiveness due to its binding to XML. Thus, we defined a policy model with corresponding XACML equivalences using the RDF signature means of [Bi13]. This enables us to reason about XACML policies and to relate them with *resource behaviour policies* and *information access policies*.

The conversion of a XACML policy from XML to RDF/XML [Be04] is accomplished with a simple eXtensible Stylesheet Language (XSL) transformation [Ka07]. Although the binding of XACML to XML assumes a tree structure and a graph-based set of RDF triples supports multiple parent elements that may result in not generating a tree sturcture, the proposed conversion is not affected by this limitation.

The use of coded attribute values is originally not supported by XACML. However, the standard provides extensions to register own data types via the `xs:Any` element. For instance, useful data types for interoperability scenarios in healthcare environments are de-

fined in [HL15b]. A coded value for a XACML resource match can be expressed as follows [HL15a]:

```
<ResourceMatch MatchId="urn:hl7-org:v3:function:CV-equal">
  <AttributeValue DataType="urn:hl7-org:v3#CV">
    <hl7:CodedValue code="B05"
              codeSystem="urn:oid:1.2.276.0.76.5.413"/>
  </AttributeValue>
  <ResourceAttributeDesignator
   AttributeId="urn:ihe:iti:xds-b:2007:folder:code"
   DataType="urn:hl7-org:v3#CV"/>
</ResourceMatch>
```

List. 2: Coded XACML resource attribute

The attribute value is now associated with a code system that is managed by a terminology service. However, the use of a particular coded value might conflict with rules of a Resource Behaviour Policy or Information Access Policy. Relations to policies of different types are achieved by class-based inheritance (cf. Fig. 1), which is a basic policy schema defining common policy classes as foundation for more specific policy types. For instance, the `urn:oasis:names:tc:xacml:2.0:policy:schema:os#Action` class is a subclass of the `urn:policy:signatures:basic#Action` class or the `urn:oasis:names:tc:xacml:2.0:policy:schema:os#Subject` is a `urn:policy:signatures:informationaccess#DataProcessor` class, too.

4 ABAC Policy Conflict Detection

Policy conflicts are manifold and basic model is defined by Moffett and Sloman [MS94] and distinguish conflicts between conflict of modalities and conflict of goals. Modality conflicts are defined by inconsistencies between the policy modes (e.g., one policy grants and another one denies access). Goal conflicts may occur when subjects and objects are overlapping in different combinations. Other policy conflicts have a semantical origin, such asdDomain or data conflicts as described in [Ku13]. These conflicts are detectable through of SPARQL queries [PS08] against a policy store (i.e. we do not rely on a rule system). However, semantic conflicts are more difficult to detect because their artifacts need to be mapped, resolved, and correlated by the policy system first.

4.1 Domain Conflict Detection

Formally subsumption relations between the policy attribute values must be defined from different policy types, so that $CodedValue_{PolicyTypeA} \sqsubseteq CodedValue_{PolicyTypeB}$ applies. This does not imply that the code values are organized hierarchically in the same code system but that one code value involves another causing domain conflicts when semantically similar policy attributes are not disjoint. For instance, the following instance is a valid definition of subsumption relations across multiple policy types (cf. Fig. 2). The code value `EXECUTE` of an action type in a *resource behaviour policy* means that it is a valid application with a registered `USE` data activity in an *information access policy*.

To capture codes without a subset of another code, this SPARQL query may be executed:

```
PREFIX conf: <urn:policy:signatures:conflicts#> .
PREFIX prop: <urn:policy:signatures:properties#> .
SELECT ?g1 ?g2 ? superior ? subordinate
  WHERE {
    ?x conf:superior [
      prop:code ?superior
    ].
    ?x conf:subordinate [
      prop:code ?subordinate
    ].
    GRAPH ?g1 {
      ?y prop:action [
        prop:code ?superior
      ].
    }
    GRAPH ?g2 {
      ?z prop:action [
    }
  }
```

List. 3: SPARQL query to detect missing subsumption relations

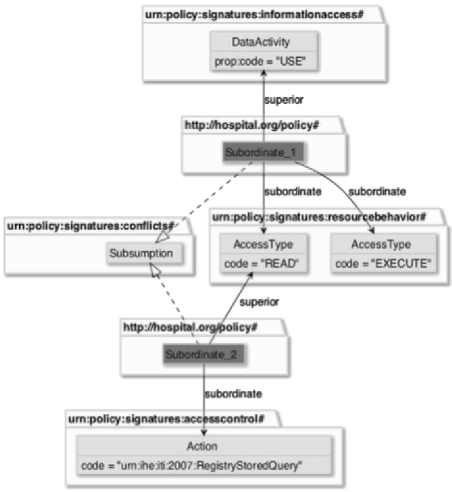

Fig. 2: UML-like policy object model for code-based subsumptions

4.2 Data Conflict Detection

Data conflicts can be either mitigated by a terminology CTS2 service or avoided entirely. Policy attributes with stored code systems or value sets can be linked via the CTS2 Map Services with related codes, in order to organise semantically connatural attributes. Thus, different designators and different scales of a coded value may be used for expressing policies. The challenge is to link the policy model with different identifiers of a coded value. To do so, each SPARQL query has to be checked: All literals with the predicate `prop:code` need to be enlarged by all available designators (SPARQL keyword `OPTIONAL`).

5 Implementation

In general, policy conflict handling may happen statically (”what if . . . ?” analysis) or dynamically during the runtime. The latter can be addressed by setting of priorities or introducing a meta policy (e.g. XACML policy rule combining algorithm). However, this primarily applies to scenarios that implement access control means in order to evaluate an authorisation decision. We focus on solving potential conflicts in policies in order to establish a valid sample space for the specific query dimensions. That is why we rely on a static approach since an automated handling of conflicts (e.g. heuristics) is impractical due to the complex relations between policy attributes based on code systems. Therefore, our conflict resolution policy sets an interventional correction mechanism for the author based on the ”security is a process” principle and is aimed at addressing most issues early through providing guidance for authoring policies.

We implemented a prototype policy management system in Java holding all applicable policies. A policy registration is accompanied by a systemic check against the policy store (RDF store). All services are facilitated with the SPARQL 1.1 Graph Store HTTP Protocol [Og13] and the SPARQL 1.1 Protocol [W313] as Figure 3 shows.

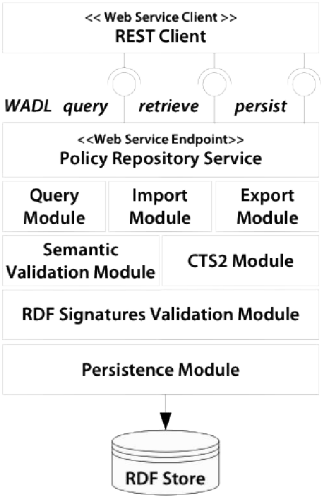

Fig. 3: Architecture policy management

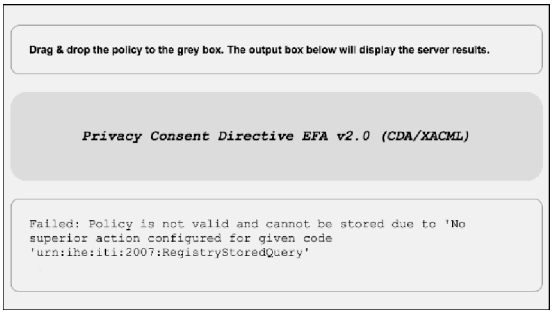

Fig. 4: Policy client interface

The *RDF Semantic Validation Module* checks for inconsistencies at a policy persist operation. It furthermore holds standard *information access policies* as well as *resource behaviour policies* that outline general security properties. Moreover, our prototype consists of a Web-based policy client (cf. Fig. 4) that accepts a patient privacy consent and an enclosed XACML policy. This policy is submitted to a `PERSIST` operation and validated. The sample output in Figure 4 indicates that the XACML action in the given context is not configured and thus not allowed to be used to avoid assigning inflated access rights.

6 Conclusions and Ongoing Work

ABAC policies should be aligned with resource behaviour and information access policies. This ensures that information security constraints are applied to access control policies. These policies may be represented through Semantic Web technologies. The presented approach does not suggest an extension of the XACML language but an in-band conversion of XACML policies to RDF in order to detect conflicts with other policy types. Although our prototype performs acceptably, more significant performance tests with more policy statements need to be executed as an acceptable response time is a prerequisite for a run-time policy decision. One facilitator could be the policy store acting as a policy decision point.

Next, we need to resolve potential conflicts between rule combination algorithms from XACML policies with *information access/resource behaviour policies*. The binary search techniques proposed by Shu et al. [SYA09] may be a good start for that. Finally, to provide a relevant contribution to the field of secure policies in eHealth domains, we will show through an exemplary patient privacy consent that inherently features a multi-dimensional policy (document/resource, external constraints, informational, and data), to be validated against and integrated with access control policies.

References

[Am13] American National Standards Institute: Next Generation Access Control – Functional Architecture (NGAC-FA). Technical Report INCITS 499-2013, ANSI, New York, NY, USA, March 2013.

[Be04] Beckett, Dave: RDF 1.1 XML Syntax. W3C Recommendation, W3C, February 2004. http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/.

[Be06] Benantar, Messaoud: Access Control Systems: Security, Identity Management and Trust Models. Springer Science+Business Media, 2006.

[Be14] Beckett, Dave; Berners-Lee, Tim; Prud'hommeaux, Eric; Carothers, Gavin: RDF 1.1 Turtle: Terse RDF Triple Language. W3C Recommendation, W3C, February 2014. http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/.

[BG14] Brickley, Dan; Guha, R. V.: RDF Schema 1.1. W3C Recommendation, World Wide Web Consortium (W3C), February 2014. http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[Bi13] Billig, Andreas: Utilizing Semantic Technologies for a CTS2 Store. Fraunhofer FOKUS, CC E-HEALTH, June 2013. Retrieved 2016-08-01, from http://semantik.fokus.fraunhofer.de/WebCts2LE/main3/cts4omg.pdf.

[Bu08] Bundesamt für Sicherheit in der Informationstechnik: BSI-Standard 100-1: Managementsysteme für Informationssicherheit (ISMS). Technical report, Bundesamt für Sicherheit in der Informationstechnik, 2008. Version 1.5.

[Ca09] Caumanns, Jörg; Kuhlisch, Raik; Pfaff, Oliver; Rode, Olaf: IHE IT-Infrastructure White Paper: Access Control. September 2009. Retrieved 2016-08-13, from http://www.ihe.net/Technical_Framework/upload/IHE_ITI_TF_WhitePaper_AccessControl_2009-09-28.pdf.

[DLJ08] Dersingh, Anand; Liscano, Ramiro; Jost, Allan: Context-aware access control using semantic policies. Ubiquitous Computing And Communication Journal (UBICC) Special Issue on Autonomic Computing Systems and Applications, 3:19–32, 2008.

[FK92] Ferraiolo, David F.; Kuhn, D. Richard: Role-based Access Controls. In: Proceedings, 15th National Computer Security Conference. Baltimore MD, pp. 554–563, 1992.

[Ha04] Hayes, Patrick: RDF Semantics. W3C Recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

[HL15a] HL7 Germany: Example: EFA Digital Consent Document. Wiki Portal for the Interoperability Forum, December 2015. Retrieved 2016-08-01, from http://wiki.hl7.de/index.php?title=Example:_EFA_Digital_Consent_Document.

[HL15b] HL7 Germany: IHE-XACML Binding: Custom Data Types (and related Comparison Functions) used for the bindings. Wiki Portal for the Interoperability Forum, March 2015. Retrieved 2016-08-01, from https://wiki.hl7.de/index.php?title=ihecb:IHE-XACML_Binding#Custom_Data_Types.

[In14] International Organization for Standardization: Information Technology – Security Techniques – Information Security Management Systems – Requirements. Technical Report ISO/IEC 27001:2013 + Cor. 1:2014, International Organization for Standardization, September 2014.

[In15] International Organization for Standardization: Information Technology – Security Techniques – Information Security Management Systems – Requirements. Technical Report ISO/IEC 27001:2013/Cor 2:2015, International Organization for Standardization, July 2015.

[Ka07] Kay, Michael: XSL Transformations (XSLT) Version 2.0. W3C Recommendation, W3C, January 2007. http://www.w3.org/TR/2007/REC-xslt20-20070123/.

[KLW95] Kifer, Michael; Lausen, Georg; Wu, James: Logical Foundations of Object-Oriented and Frame-Based Languages. Journal of ACM, 42:741–843, May 1995.

[Ku13] Kuhlisch, Raik: A Description Model for Policy Conflicts for Managing Access to Health Information. In (Lantow, Birger; Sandkuhl, Kurt; Seigerroth, Ulf, eds): Proceedings, 6th International Workshop on Information Logistics, Knowledge Supply and Ontologies in Information Systems (ILOG). volume 1028 of CEUR Workshop Proceedings, M. Jeusfeld c/o Redaktion Sun SITE, Informatik V, RWTH Aachen, Aachen, pp. 44–55, 2013.

[Mo05] Moses, Tim: eXtensible Access Control Markup Language (XACML) Version 2.0. Technical Report oasis-access_control-xacml-2.0-core-spec-os, OASIS, February 2005.

[MS94] Moffett, Jonathan D.; Sloman, Morris S.: Policy Conflict Analysis in Distributed System Management. Journal of Organizational Computing, 4(1):1–22, 1994.

[Ob15] Object Management Group: Common Terminology Services 2, Version 1.2. Technical Report formal/2015-04-01, OMG, April 2015. http://www.omg.org/spec/CTS2/1.2/.

[Og13] Ogbuji, Chimezie: SPARQL 1.1 Graph Store HTTP Protocol. W3C Recommendation, W3C, March 2013. http://www.w3.org/TR/2013/REC-sparql11-http-rdf-update-20130321/.

[PS08] Prud'hommeaux, Eric; Seaborne, Andy: SPARQL Query Language for RDF. W3C Recommendation, W3C, January 2008. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

[Sa15] Sandkuhl, Kurt; Matulevicius, Raimundas; Ahmed, Naved; Kirikova, Marite: Refining Security Requirement Elicitation from Business Processes Using Method Engineering. In: BIR 2015. volume 1420, pp. 98–109, 2015.

[So98] Solms, Rossouw von: Information Security Management (2): Guidelines to the Management of Information Technology Security (GMITS). Information Management & Computer Security, 6(5):221–223, 1998.

[SYA09] Shu, Cheng-chun; Yang, Erica Y.; Arenas, Alvaro E.: Detecting Conflicts in ABAC Policies with Rule-Reduction and Binary-Search Techniques. IEEE, pp. 182–185, July 2009.

[W313] W3C SPARQL Working Group: SPARQL 1.1 Overview. W3Crecommendation, W3C, March 2013. http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.

[ZZ14] Zou, Jiashun; Zhang, Yongsheng: Research of Policy Conflict Detection and Resolution in ABAC. Journal of Computational Information Systems, 10(12):5237–5244, 2014.