POLIPO: Policies & OntoLogies for Interoperability, Portability, and autOnomy

Daniel Trivellato
Ei/Ψ Group, TU/e
d.trivellato@tue.nl

Fred Spiessens
Ei/Ψ Group, TU/e
a.o.d.spiessens@tue.nl

Nicola Zannone
Ei/Ψ Group, TU/e
n.zannone@tue.nl

Sandro Etalle
Ei/Ψ Group, TU/e
s.etalle@tue.nl

***Abstract*—In this paper we identify the requirements for the definition of a security framework for distributed access control in dynamic coalitions of heterogeneous systems. Based on the elicited requirements, we introduce the POLIPO framework that combines distributed access control with ontologies to give a globally understandable semantics to policies, enabling interoperability among heterogeneous systems.**

I. Introduction

In the context of the POSEIDON project[1] we are developing a security framework for distributed access control in a System of Systems (SoS) [1]. The SoS model is suitable to represent and analyze the Maritime Safety and Security (MSS) domain which is the focus of POSEIDON. The MSS domain is characterized by the interaction and collaboration between autonomous and heterogeneous systems sharing information, processes, and resources. These collaborations are not fixed a priori, but can dynamically change over time as new parties join, leave, or change their responsibilities and objectives. An example of MSS coalition is the task force created by the European Union to protect commercial shipping off the Somali coast against piracy.[2]

The SoS model eliminates the physical boundaries of organizations and increases operational flexibility, but affects interoperability and security of collaborating parties. Sharing sensitive information with other parties may be required for the success of the coalition. For example, a commercial ship having an engine failure off Somali coast can communicate its position to the EU task force for aid. EU surveillance vessels, however, may employ different communication models and thus are not able to interpret the message. Moreover, the message can be intercepted by pirates that can exploit this emergency situation to attack the ship.

The development of authorization mechanisms for distributed systems has become a major research challenge. Trust management (TM) is an approach to this problem in which access control decisions are based on credentials [2], [3]. However, TM does not support any control on the data after their disclosure. Therefore, distributed access control needs to be complemented with portable policies (i.e., *sticky policies*) [4]. Nevertheless, TM and sticky policies do not provide a solution to the problem of interoperability.

[1]http://www.esi.nl/poseidon/

[2]http://www.consilium.europa.eu/showPage.aspx?id=1518&lang=en

When heterogeneous systems form coalitions that transgress the traditional boundaries among organizational, cultural, and legal units, vocabulary alignment is required to enable mutual understanding among parties. Ontologies provide a means for establishing common vocabularies by providing a precise semantics to concepts and relationships in a domain. This has spurred researchers to use ontologies for the specification of security policies [5]. However, the expressive power of these policies is restricted to the one of ontology languages. To overcome this limitation, ontologies have been extended with rules [6], but this extension often causes ontology reasoning to become undecidable [7].

This paper proposes the POLIPO (Policies & OntoLogies for Interoperability, Portability, and autOnomy) framework to address the problem of distributed access control in an SoS. In particular, we make the following contributions:

- We identify the requirements for the definition of a security framework for distributed access control in dynamic coalitions of heterogeneous systems.
- We propose a policy language that combines access control and trust management for the specification of security policies in an SoS.
- We enable interoperability between heterogeneous systems, by applying ontology-based vocabulary alignment to distributed access control. The attributes certified in a credential and the actions characterizing a permission are defined in a shared ontology.
- We combine policy rules with ontologies to improve the query answering support to the knowledge base. This makes it possible to exploit the reasoning services offered by ontologies. However, we do not extend ontologies with rules to avoid undecidability problems. In the POLIPO framework, ontologies are used as remote oracles to infer domain knowledge.
- We discuss an implementation of the framework.

II. Requirements

A practical policy framework should support autonomy in policy specification as well as interoperability among parties and policy portability. In this section, we clarify and discuss the requirements that have driven the design of POLIPO.

Requirement 1 (Autonomy): Every party shall be able to design and express its policy autonomously.

978-0-7695-3742-9/09 $25.00 © 2009 IEEE
DOI 10.1109/POLICY.2009.19

(1)	SeniorOfficer $\sqsubseteq$ Officer JuniorOfficer $\sqsubseteq$ Officer
(2)	TemporaryOfficer $\sqsubseteq$ SeniorOfficer TemporaryOfficer $\sqsubseteq$ JuniorOfficer
(3)	JuniorOfficer $\sqsubseteq \neg$ SeniorOfficer
(4)	StarNavy $\sqsubseteq$ AlliedNavy

Figure 1. Sample POSEIDON Ontology

This means that every party shall be able to specify its policies regardless of which parties have already joined the coalition. While this requirement may seem obvious and trivial to implement, we state it explicitly because it has implications that can easily conflict with other requirements.

Requirement 2 (Interoperability): Parties shall be able to interact with each other unambiguously.

As security policies have to be understood by all parties, the problem of semantic interoperability emerges. To tackle this problem, we turn to the established practice of ontologies. An ontology is a formal representation of a domain in terms of concepts and relationships, each with a precise semantics.

Some ontologies support non-monotonic constraints (e.g., disjointness of concepts). This implies that extending an ontology can introduce inconsistencies. Therefore, when using a single shared ontology as common vocabulary, we cannot simply allow parties to it by adding their own local concepts and constraints, because this would introduce limitations that are based on the order in which the parties join the coalition.

Example 1: Take the ontology in part 1 of Fig. 1 and suppose two parties want to join the coalition and add their constraints to this ontology. One party wants to add part 2 while the other party wants to add part 3. Whatever party joins the coalition first will be able to make its adaptations without making the ontology inconsistent. But thereafter, extending the ontology for the other party would be illegal.

The example above shows that demanding a shared ontology to be consistent would conflict with autonomy. For this reason, we allow local extensions to the shared ontology. When a party joins the coalition, it imports the POSEIDON ontology into its local ontology, and possibly reuses and extends some global concepts. Every party should keep its local ontology consistent. However, the extensions made locally to the imported ontology remain inside the ontology of a certain party and, therefore, cannot affect the consistency of the ontologies of other parties.

Requirement 3 (Portability): Remote evaluation of policies shall preserve the interpretation of the policy owner.

Policies can be transferred to other parties, with the intention of restricting the usage and redistribution of data. Typically, sticky policies are attached to the data they protect, following them from system to system, and are evaluated and enforced remotely at the side of a trusted data recipient. When an access request is processed remotely, the data owner wants to be sure that the sticky policy is evaluated according to his own interpretation. Thereby, the data owner has to be careful when designing a policy, to preclude illegitimate flows of information. In this setting, the use of credentials allows the data owner to maintain more control over remote access to data.

Example 2: The Green Star Navy allows senior officers of allied navies to access some sensitive information, where allied navies include Star navies according to the POSEIDON ontology (part 4 of Fig. 1). Senior officers of the Blue Star Navy are thus allowed to access the information. However, the Blue Star Navy also includes the Grey Cross Navy among its allies in its local ontology. If the policy is evaluated locally within the Blue Star Navy, senior officers of the Grey Cross Navy are granted permission to access the information, against the intention of the Green Star Navy which does not have the Grey Cross Navy among its allies. To prevent this disclosure of information, the Green Star Navy can specify that only senior officers of navies having an 'AlliedNavy' credential signed by the Green Star Navy itself are entitled to access the information.

Data recipients need to avail of all necessary information to evaluate a policy correctly. Therefore, the data owner must make sure that sticky policies are self-contained as far as their evaluation is concerned.

III. The POLIPO framework

In this section we present POLIPO (Policies & OntoLogies for Interoperability, Portability, and autOnomy), a security framework that combines distributed access control with ontologies to enable interoperability, portability, and autonomy in dynamic coalitions of heterogeneous systems. In particular, we introduce a logic-, ontology-based language for the specification of distributed access control policies.

POLIPO policies are specified using four constructs:

- *ontology atoms*: are used to query the knowledge base represented by ontologies. Each concept in an ontology is identified by a $conceptURI$. $conceptURI(a)$ holds if a is an instance of $conceptURI$. Each relationship in an ontology is identified by a $relationshipURI$. $relationshipURI(a_1, a_2)$ holds if instance a_1 is related to instance a_2 via $relationshipURI$. Examples of ontology atoms are given in part 1 of Fig. 2: $psd{:}SeniorOfficer$ is a $conceptURI$ and $psd{:}worksFor$ is a $relationshipURI$. These atoms check whether '$John$' is a senior officer and if he works for the Green Star Navy (GS is the unique identifier of the Green Star Navy) according to the definitions given in the POSEIDON ontology. Consistently with the XML (and OWL) convention, we use prefixes followed by symbol ":" to substitute base URIs (e.g., *psd:* stands for *www.example.net/poseidon/PSD-Ontology/*).
- *credential atoms*: represent digitally signed statements made by an issuer about an attribute of a subject. A

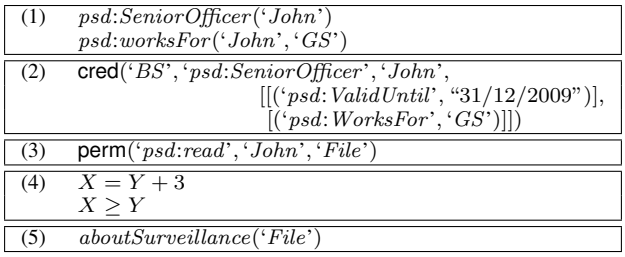

Figure 2. Examples of POLIPO Atoms

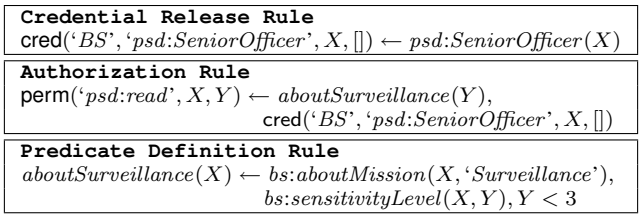

Figure 3. Examples of POLIPO Rules

credential atom has the form:

$$\mathsf{cred}(issuer, att, subject, [[c_1, \ldots, c_n], [a_1, \ldots, a_n]])$$

where $issuer$ is the unique name of the entity signing the credential; att is a $conceptURI$ specifying the attribute for which the credential is issued; $subject$ is the unique name of the entity to whom the credential refers; $[c_1, \ldots, c_n]$ and $[a_1, \ldots, a_n]$ are lists of optional properties. Each c_i (respectively a_i) is a pair $(property, value)$, where $property$ is a $relationshipURI$ related to the the concept credential[3] (resp. to att), and $value$ bounds the range of the relationship. Part 2 of Fig. 2 shows a credential issued by the Blue Star Navy (BS) and certifying the fact that '$John$' is a senior officer of the Green Star Navy. The credential has validity period as an optional property.

- *authorization atoms*: denote the permission of a subject to perform an action on an object. They have form:

$$\mathsf{perm}(action, subject, object)$$

where $action$ is a $relationshipURI$ specifying the action that $subject$ is allowed to perform on $object$; $subject$ is the unique name of the entity to whom the permission is granted; $object$ represents the target of the permission. In the authorization atom in Fig. 2 part 3, '$John$' is given the permission to read '$File$'.

- *constraints*: are specified using Constraints Logic Programming (CLP) [8] constraints (e.g., $=$, $>$, $<$, etc.) or user-defined predicates (resp. parts 4 and 5 in Fig. 2).

We formalize POLIPO policies in the logic programming paradigm. POLIPO rules are Horn clauses of the form $h \leftarrow b_1, \ldots, b_n$, where h, called head, is an atom, and $b_1, \ldots, b_n$, called body, are literals (i.e., positive or negative atoms) with $n \geq 0$. Negation is treated as negation-as-failure (denoted by not): if there is no evidence that an atom is true, it is considered to be false. We also assume that each variable occurring in the head of a rule, in a negative literal, as issuer of a credential atom, or in a CLP constraint also occurs in at least one positive literal in the body of the same rule.

We distinguish three types of rules: *credential release rules*, *authorization rules*, and *predicate definition rules*.

[3]cred and perm are formally defined in the POSEIDON ontology.

Definition 1 (Credential Release Rule): A credential release rule is a Horn clause where the head is a credential atom and the body can contain positive credential and ontology atoms, and constraints.

Example 3: The first rule in Fig. 3 states that the Blue Star Navy releases a $psd{:}SeniorOfficer$ credential to entities defined as senior officers in the POSEIDON ontology.

It is worth noting the difference in the use of $psd{:}SeniorOfficer$. When it occurs as the predicate name in the ontology atom, it refers to the knowledge base of the Blue Star Navy. When it occurs in the credential atom, it is the attribute of the subject certified by the navy.

Definition 2 (Authorization Rule): An authorization rule is a Horn clause where the head is an authorization atom and the body can contain positive credential, authorization, and ontology atoms, constraints, and negative ontology and user-defined atoms.

Example 4: The second rule in Fig. 3 states that subject X is authorized to read object Y if Y is about surveillance and X provides a credential issued by the Blue Star Navy stating that he is a senior officer.

We only allow negation-as-failure of ontology and user-defined atoms in the body of authorization rules for two reasons. First, we restrict non-monotonicity to predicates in the knowledge base. This guarantees that POLIPO policies are stratified logic programs, ensuring efficiency and unambiguity. Second, there is a semantic difference between credential release rules and authorization rules. Authorization rules are evaluated to determine whether a permission should be granted or not at a certain instant; changes in the truth value of an atom in the body only affect future access decision, without impacting past decisions. On the contrary, once a credential is issued, it is valid for a period of time.

Definition 3 (Predicate Definition Rule): A predicate definition rule is a Horn clause where the head is a user-defined atom and the body can contain positive ontology atoms and constraints.

Example 5: The third rule in Fig. 3 defines the user-defined predicate $aboutSurveillance$. An object is about surveillance if it concerns mission '$Surveillance$' and has sensitivity level less than 3 according to the Blue Star Navy.

We have limited the occurrence of ontology atoms to the body of POLIPO rules. This is to keep the policy and ontology reasoning separated. Indeed, by allowing a

free interaction of ontologies with rules (i.e., using rules to modify the semantics of ontologies) the ontology reasoning may become undecidable [7].

POLIPO policies consist of sets of POLIPO rules. In particular, we distinguish two types of policies: *credential release policies* and *authorization policies*.

Definition 4 (Credential Release Policy): A credential release policy is a set of credential release rules.

Definition 5 (Authorization Policy): An authorization policy is a set of authorization rules.

We have implemented a prototype of the policy decision engine in SWI-Prolog. The main reason for using SWI-Prolog is that it provides an interface with ontologies, through the Semantic Web Library. This library consists of packages for reading, querying and storing RDF documents, and hence ontologies (every OWL ontology can be represented as RDF triples). For example, the ontology atom $psd{:}SeniorOfficer(`John')$ can be expressed using the built-in predicate *rdf(John,rdf:type,SeniorOfficer)*, where *rdf:* is the prefix of the URI where relationship *type* is defined. The downside of this implementation choice is that SWI-Prolog does not automatically deal with loops. A solution for handling recursive credentials will be the subject of a whole paper devoted to the algorithms of POLIPO.

IV. Conclusions and Future Work

In recent years, a lot of effort has been invested in the specification of access control policies for distributed systems. Li et al. [3] introduce the trust management framework RT which addresses vocabulary alignment using Application Domain Specification Documents (ADSDs). Similarly to RT, TuLiP [2] uses XML namespaces to avoid name conflicts and facilitate the definition of a common vocabulary. However, these approaches are purely syntactical and do not provide the semantics, expressiveness and reasoning facilities provided by ontologies.

In the attempt to enhance the Semantic Web with security policies, ontologies have been combined with rules [9]. However, integrating ontologies and rules may cause DL reasoning to become undecidable [6]. This limitation has spurred researchers to investigate syntactic restrictions that keep DL with rules decidable [7]. Our approach relates well to [10] which imports in logic programs the knowledge contained in ontologies, without fully integrating the two frameworks. The difference lies in the fact that we do not allow the flow of information from the logic program to the ontology. Eiter et al. use the knowledge inferred by rules to temporarily feed the ontology for further reasoning. This, however, can introduce inconsistencies in the ontology.

This paper has discussed the requirements for the definition of a security framework for distributed access control in the context of the POSEIDON project. Based on the elicited requirements, we have proposed POLIPO, a framework that combines access control, trust management and ontologies to enable interoperability, portability, and autonomy in dynamic coalition of heterogeneous systems. Due to the lack of space, we refer to [11] for a discussion on portable policies.

The work is still in progress to increase the flexibility and expressive power of the language. We are investigating a method to allow parties to specify policies using local vocabulary in such a way that policies can be still understood and evaluated correctly by other parties.

Acknowledgments

We thank Willem R. van Hage for the useful discussions. This work has been carried out as part of the POSEIDON project under the responsibility of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK03021 program. This work is also funded by the EU TAS3 project.

References

[1] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Eng.*, vol. 1, no. 4, pp. 267–284, 1999.

[2] M. Czenko and S. Etalle, "Core TuLiP Logic Programming for Trust Management," in *Proc. of ICLP'07*, ser. LNCS 4670. Springer, 2007, pp. 380–394.

[3] N. Li, J. C. Mitchell, and W. H. Winsborough, "Design of a Role-Based Trust-Management Framework," in *Proc. of Symp. on Security and Privacy*. IEEE Computer Society, 2002, pp. 114–130.

[4] S. Etalle and W. H. Winsborough, "A posteriori compliance control," in *Proc. of SACMAT'07*. ACM, 2007, pp. 11–20.

[5] A. Uszok et al., "KAoS Policy Management for Semantic Web Services," *IEEE Intelligent Systems*, vol. 19, no. 4, pp. 32–41, 2004.

[6] I. Horrocks, P. F. Patel-Schneider, S. Bechhofer, and D. Tsarkov, "OWL rules: A proposal and prototype implementation," *JWS*, vol. 3, no. 1, pp. 23–40, 2005.

[7] R. Rosati, "DL+log: Tight Integration of Description Logics and Disjunctive Datalog," in *Proc. of KR'06*. AAAI Press, 2006, pp. 68–78.

[8] K. Marriott and P. J. Stuckey, *Programming with constraints: an introduction*. MIT Press, 1998.

[9] L. Kagal et al., "Authorization and Privacy for Semantic Web Services," *IEEE Intelligent Systems*, vol. 19, no. 4, pp. 50–56, 2004.

[10] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits, "Well-founded semantics for description logic programs in the semantic web," in *In Proc. of RuleML'04*, ser. LNCS 3323. Springer, 2004, pp. 81–97.

[11] D. Trivellato, F. Spiessens, N. Zannone, and S. Etalle, "POLIPO: Policies & OntoLogies for Interoperability, Portability, and autOnomy," Eindhoven University of Technology, Tech. Rep. CS Report 09/06, 2009.