International Journal of Cooperative Information Systems
© World Scientific Publishing Company

A SEMANTIC SECURITY FRAMEWORK FOR SYSTEMS OF SYSTEMS

DANIEL TRIVELLATO
Eindhoven University of Technology
Eindhoven, The Netherlands

NICOLA ZANNONE
Eindhoven University of Technology
Eindhoven, The Netherlands

MAURICE GLAUNDRUP
Thales Nederland
Hengelo, The Netherlands

JACEK SKOWRONEK
Thales Nederland
Hengelo, The Netherlands

SANDRO ETALLE
Eindhoven University of Technology
Eindhoven, The Netherlands
University of Twente
Enschede, The Netherlands

Received (8 March 2012)
Revised (8 March 2012)

Systems of systems (SoS) are dynamic coalitions of distributed, autonomous and heterogeneous systems that collaborate to achieve a common goal. While offering several advantages in terms of scalability and flexibility, the SoS paradigm has a strong impact on systems interoperability and on the security requirements of the collaborating parties. In this paper we introduce a service-oriented security framework that protects the information exchanged among the parties in an SoS, while preserving parties' autonomy and interoperability. Confidentiality and integrity of information are protected by combining context-aware access control with trust management. Autonomy and interoperability among parties are enabled by the use of ontology-based services. More precisely, parties may refer to different ontologies to define the semantics of the terms used in their security policies and to describe domain knowledge and context information; a semantic alignment technique is then employed to map concepts from different ontologies and align the parties' vocabularies. We demonstrate the applicability of our solution by deploying a prototype implementation of the framework in an SoS in the maritime safety and security domain.

Keywords: systems of systems; information confidentiality and integrity; systems interoperability.

1. Introduction

Systems of systems (SoS)[30] are coalitions of autonomous systems and services that collaborate to achieve a common goal. These coalitions are dynamic, with systems joining and leaving, and involve parties employing different protocols, vocabularies, data models and organizational structures. Examples of SoS include web services, mobile ad-hoc networks (MANETs),[10] socio-technical systems,[49] etc.

Despite offering a high degree of operational flexibility, the SoS paradigm has a strong impact on systems interoperability as well as on the security requirements of the collaborating parties. Sharing sensitive information with other parties in an SoS might be required for the success of the coalition; nevertheless, such information should be accessed exclusively by authorized parties, which may vary depending on the context (e.g., the time of access, the criticality of a situation). Therefore, along with the development of SoS comes the demand for a flexible security framework that faces the related security challenges, providing usable and trusted tools to support parties in their security management.

In particular, to deal with the dynamic nature of SoS, a security framework should be able to take into account security-relevant contextual information available at the time an access request is made, and incorporate it in the access control decision.[4] Contextual information may consist of "basic" environmental conditions (e.g., the location of the requester, the time of access), or more complex conditions derived from the basic ones (e.g, an emergency situation due to the collision between two vessels). Context-aware access control models[4,14,43] can be employed to serve this purpose.

In addition, contrarily to centralized systems where users and resources belong to a single, trusted domain, in an SoS parties often do not know each other beforehand; it is therefore not possible to rely on identity-based approaches to regulate the access to local resources. Trust Management[5] (TM) has been proposed as a solution to this problem. TM is an approach to access control in distributed systems where access decisions are based on the attributes of a requester, which are certified by means of digital credentials. Credentials are certificates attesting that a subject has a certain attribute, and are digitally signed to ensure their authenticity and integrity.

The problem of most TM frameworks proposed in the literature (e.g., see Ref. 1, 29, 32) is that they assume a complete agreement among the parties in a distributed system on the vocabulary used to denote subjects' attributes and to describe the concepts and relationships that characterize a given application domain. When heterogeneous systems form dynamic coalitions that transgress the boundaries between organizational and cultural units, however, this assumption is unrealistic. More likely, parties will "speak" different languages and employ different organizational models; nevertheless, they must be able to collaborate to achieve the coalition's goal. As a first step towards enabling mutual understanding and thus interoperability among parties in an SoS, semantic approaches have been adopted for policy specification.[25,48] In particular, ontologies have been largely used in the Semantic Web to assign a precise structure and semantics to information and to define domain knowledge. Accordingly, parties can refer to ontologies to provide a semantics to the terms used to specify their policies and to describe the application domain.

The use of ontologies alone, however, is not enough to achieve interoperability. In fact, parties might refer to different ontologies to denote the same (or similar) concepts in the domain; semantic alignment techniques[16,22] need thus to be employed to map concepts from different ontologies, i.e., to align different vocabularies and organizational models. A major drawback of existing semantic alignment techniques is that they require complete knowledge of the ontologies to be aligned. In many SoS, however, this requirement might not be satisfied since parties do not know each other beforehand or might want to keep part of their knowledge base confidential. Therefore, a solution that is also effective when working with partial knowledge needs to be devised.

In this paper we present the security framework for SoS that we are developing in the context of the Poseidon project[a], a joint project involving a number of industrial (Thales Nederland[b] and Noldus[c]) and academic partners. The proposed framework protects the confidentiality and integrity of information, while preserving autonomy and interoperability among parties in dynamic, inter-organizational coalitions of systems and services. In particular, it combines context-aware access control with TM to protect information from unauthorized access and improper modification. Autonomy and interoperability are enabled by the use of ontology-based services. More precisely, parties may refer to (possibly) different ontologies in the specification of their policies and to describe domain knowledge and context information; this allows each party in the coalition to employ the organizational model and terminology that they consider more appropriate within their system. The semantic alignment technique presented in Ref. 44, which is based on the notion of similarity between ontology concepts, is then employed to align their vocabularies, enabling mutual understanding. To overcome the problem of partial knowledge of the parties' ontologies, the proposed alignment technique considers concepts' similarity "estimates" (since they are computed based on partial knowledge) issued by different parties in the SoS, and combines these estimates into a single similarity value weighing them based on the "reliability" of their issuer.

We also present a prototype implementation of the security framework. The prototype has been deployed into an SoS in the maritime safety and security domain and is employed by each party in the SoS to protect local resources. The framework's

[a]http://www.esi.nl/poseidon
[b]http://www.thalesgroup.com/netherlands
[c]http://www.noldus.com

architecture, inspired by XACML,[36] consists of a set of core security components (e.g., the access control and trust management components) complemented by the ontology-based services. All components and services have been implemented following the service-oriented architecture paradigm[38] to facilitate their integration and deployment into existing SoS. The modularity of the framework allows for the integration of additional services to support the evaluation of policies and provide additional functionalities (e.g., a signature verification module). Since usability is a major concern for the deployability of a security framework,[11] we have also developed a policy editor that assists security administrators in the specification and management of security policies.

The remainder of the paper is organized as follows. Section 2 presents a use case scenario for an SoS in the maritime safety and security (MSS) domain, and elicits a set of basic requirements that a security framework for SoS should satisfy. Section 3 discusses related work. An ontology-based policy language and semantic alignment technique are introduced in Section 4. Section 5 describes the architecture of our security framework; a prototype implementation of the framework is then presented in Section 6. Finally, Section 7 concludes the paper, providing directions for future work.

2. Requirements Elicitation

In this section we first introduce a scenario for SoS in the maritime safety and security (MSS) domain, which is the application domain of the Poseidon project. Then, based on this scenario and our experience in the Poseidon and TAS3[d] projects,[6] we identify some key requirements that a security framework for SoS should satisfy.

2.1. *Case Study: EU NAVFOR*

SoS in the MSS domain have the task of monitoring a given maritime area and taking appropriate actions in case that unwanted events take place. Typically, monitoring involves various sensory inputs (e.g., radar, AIS[e]) as well as reference data available on the Internet. Maritime SoS can be used in a static context (e.g., border patrol), or in response to special circumstances in a maritime area (e.g., search and rescue missions).

An example of SoS for the latter case is the anti-piracy operation headed by the European Union that is currently taking place in the Horn of Africa. The objective of this operation is to prevent, deter and repress criminal activities and acts of piracy and armed robbery against ships of the World Food Programme, of the African Union Military Mission in Somalia (AMISOM), and other vulnerable vessels transiting off the Somali coast. Here, the SoS consists of a Maritime Security Center

[d]http://www.tas3.eu
[e]The Automatic Identification System (AIS) is a short range coastal tracking system used for identifying and locating vessels.

(MSC) and the EU Naval Force (EU NAVFOR) vessels belonging to the different countries involved in the operation. In addition, search and rescue (SAR) vessels of those countries may temporarily join the coalition in case of emergencies.

In the remainder of the section, we present a scenario centered on the monitoring of terrorist activities. We point out that all the names and facts introduced in this scenario are purely fictitious. The scenario involves the following actors:

- The EU, which determines the countries that are taking part to the operation and their tasks.
- The MSC, located in Northwood (UK), which coordinates the activities of the EU NAVFOR vessels present in the operation area.
- IT-1, a vessel of the Italian navy with patrolling tasks off the Somali coast.
- DK-1, a frigate of the Danish navy in command of operations off the Somali coast.
- Blue Star, a cargo ship transiting off the Somali coast and heading to Copenhagen.
- Black Pearl, a high speed craft.
- CG-1, a Dutch coastguard vessel.

The scenario is divided into two parts. The first part is set off the coast of Somalia, while the second part of the scenario takes places off the Dutch coast. The first part of the scenario consists of the following steps:

(1) Vessel IT-1 is patrolling a maritime area south-east of the Horn of Africa. Vessel DK-1 is patrolling the Gulf of Aden.
(2) An operator of IT-1 notices on his diplay that two ships, named Blue Star and Black Pearl, are suspiciously approaching each other at a nearby location. The operator of IT-1 requests to DK-1 (which is in command of operations) whether more information about those two ships is available; furthermore, he requests whether IT-1 should intervene to perform a closer investigation of the activities in which the two ships are engaged.
(3) The cargo ship Blue Star is already under investigation by the Danish navy because it is suspected of being involved in terrorist activities. The Danish navy has infiltrated agents who are investigating the evolution of the events. Since IT-1 has patrolling tasks and is not assigned to the investigation of terrorist activities, the Danish navy does not provide to IT-1 the extra information it gathered about Blue Star; furthermore, IT-1 is ordered not to intervene.
(4) After navigating next to each other for some time, Blue Star and Black Pearl split and proceed in opposite directions. Black Pearl proceeds in east direction, while Blue Start continues its travel towards Copenhagen.

The second part of the scenario consists of the following steps:

(6) In the proximity of the Dutch coast the cargo ship Blue Star gets into trouble due to a storm and starts drifting.

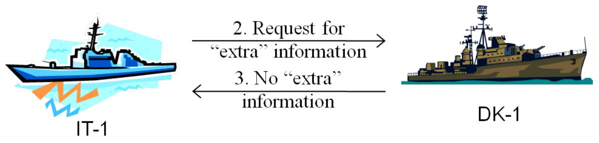
Table 1. Policies of the Parties in the Scenario

Party	**Policy ID**	**Security Policy**
EU	EU_1	The Italian navy is a member of the EU NAVFOR.
	EU_2	The Danish navy is a member of the EU NAVFOR.
	EU_3	The Dutch navy is a member of the EU NAVFOR.
MSC	MSC_1	Operators on vessels of the EU NAVFOR can access public information about the ships transiting in the operation area.
	MSC_2	Operators on vessels of the EU NAVFOR which are assigned to the prevention of criminal activities (or similar tasks) can access additional "off the record" information about ships which has been gathered during the operation.
	MSC_3	Operators on SAR vessels certified by EU NAVFOR members can access all the information about a ship in case of emergency.
Italian Navy	IT_1	IT-1 is a patrol vessel assigned to the EU NAVFOR.
Danish Navy	DK_1	DK-1 is a frigate assigned to the EU NAVFOR.
	DK_2	Operators on vessels of the EU NAVFOR can access public information about the ships transiting in the operation area.
	DK_3	Operators on vessels of the EU NAVFOR which are assigned to the prevention of criminal activities (or similar tasks) can access additional "off the record" information about ships which has been gathered during the operation.
Dutch Navy	NL_1	All SAR vessels of the Dutch coastguard are certified as SAR vessels by the Dutch navy.
Dutch Coastguard	CG_1	CG-1 is a SAR lifeboat of the Dutch coastguard.

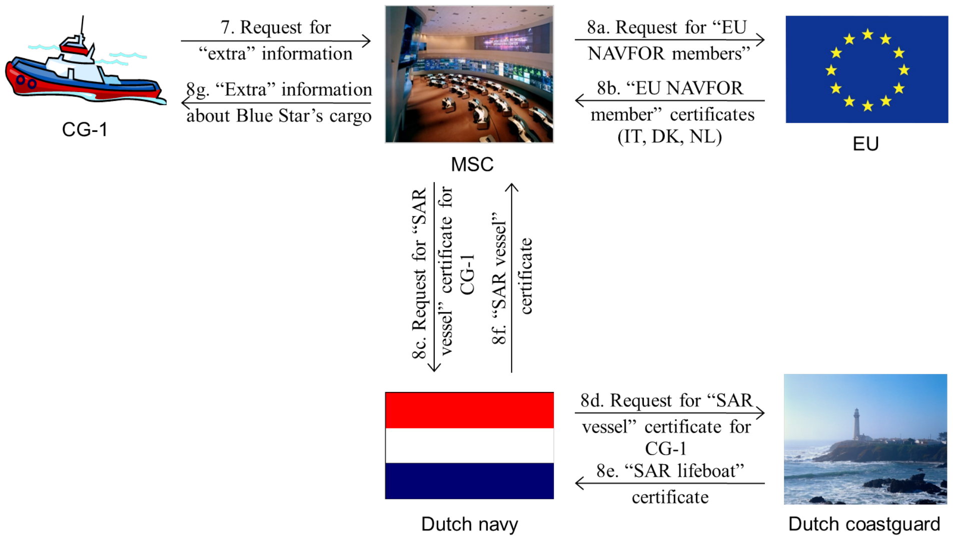
(7) The coastguard vessel CG-1 is nearby and prepares to intervene to give assistance to Blue Star's crew. In order to prepare the intervention, CG-1 needs to have information about the cargo transported by Blue Star. By checking the port from which Blue Star departed, CG-1 infers that the cargo ship has transited off the Somali coast. Therefore, CG-1 sends a request for extra information about Blue Star also to the MSC in Northwood.

(8) Due to the emergency situation, and since CG-1 is a vessel certified for SAR operations by the Dutch navy, which is also part of the EU NAVFOR, the MSC provides extra information about Blue Star's cargo to CG-1. Through this information CG-1's operators find out that Blue Star's cargo contains Anthrax that was possibly meant to be distributed to terroristic cells in Europe. The rescuers must use protective clothes and other ships must keep at a safe distance of at least 500 meters.

Table 1 presents the security policies governing the scenario; to each policy we assign a unique identifier (column *Policy ID*) that we use to refer to the policy in the remainder of the paper. Note that the vocabulary used by the different parties to specify their security policies is not always consistent. For example, policy NL_1 states that the Dutch navy certifies all the "SAR vessels" of the Dutch coastguard. CG-1, however, is certified as a "SAR lifeboat" by the Dutch coastguard (policy CG_1). In this case, an alignment of the vocabularies of the Dutch navy and coast-



(a) Interactions in the First Part of the Scenario (Somali Coast)


CG-1
7. Request for "extra" information
8g. "Extra" information about Blue Star's cargo
MSC
8a. Request for "EU NAVFOR members"
8b. "EU NAVFOR member" certificates (IT, DK, NL)
EU
8c. Request for "SAR vessel" certificate for CG-1
8f. "SAR vessel" certificate
Dutch navy
8d. Request for "SAR vessel" certificate for CG-1
8e. "SAR lifeboat" certificate
Dutch coastguard

(b) Interactions in the Second Part of the Scenario (Dutch Coast)

Fig. 1. Interactions among Parties in the Scenario

guard would be required to allow for a certification of CG-1 by the Dutch navy. The alignment should take into consideration the affinity between the terms "SAR vessel" and "SAR lifeboat". Since a lifeboat is a type of SAR vessel, and thus the affinity between the two terms is high, in our scenario we expect the Dutch navy to certify CG-1 as a "SAR vessel".

The interactions between the actors in the scenario are summarized in Figure 1. Each interaction is labeled with the step of the scenario in which it is described; in case that a step involves multiple interactions, we order them by adding a letter to the label. Since security policies are often deemed to be confidential (see Section 2.2 for a more comprehensive discussion), in our scenario we assume that no party has access to the security policies of the other parties. Therefore, whenever a party requires a certificate issued by another party, an explicit request needs to be made.

2.2. *Security Requirements*

As a first step towards the elicitation of our security objectives, we derive the characteristics of SoS which are relevant for the design of a security framework. As can be evinced from the scenario above, the distuinguishing features of SoS are the following:

- *Dynamicity*: SoS are constantly evolving. Systems may leave an SoS at any time while new systems may join the coalition, depending on the context or the progress towards the goal. Similarly, the information that systems need to exchange may be context-dependent. For instance, additional parties (such as the coastguard vessel CG-1 in our scenario) may be required to join a coalition in emergency situations, and these parties may be given access to information that they normally would not be authorized to access.
- *Distribution*: contrarily to centralized systems where users and resources belong to a single, trusted domain, SoS are characterized by the absence of a central point of control. Each system in an SoS is an independent, complex system which belongs to a (possibly) different security domain and is governed by a different authority (e.g., the Italian and Danish vessels). Furthermore, SoS are open systems in which parties may not know each other before joining the coalition. For example, in principle the MSC does not know whether CG-1 is actually a SAR vessel of an EU NAVFOR member country; for this reason, it requests a certification of CG-1 from a trusted party, i.e., the Dutch navy.
- *Heterogeneity*: as a consequence of the independence of the systems in an SoS, each system may adopt different data and organizational structures, and a different vocabulary to define the concepts and relationships in an application domain. In policy CG_1, for instance, the Dutch coastguard refers to CG-1 as a "SAR lifeboat"; the Dutch navy, however, certifies only "SAR vessels" as Dutch SAR vessels.

These features impose serious challenges on the design of a security framework. Here, we identify the following set of core security requirements that a security framework for SoS should satisfy:

(1) *Protection of information confidentiality and integrity*: protecting sensitive data exchanged among the parties in the SoS from unauthorized access and improper modification is a key requirement for every security framework. In our scenario, for instance, if terrorists would be able to access the information gathered by the EU NAVFOR, they would know that the Danish navy is investigating the activities of the cargo ship Blue Star. Security policies, however, may also contain sensitive information.[7,40,53] In particular, the disclosure of a policy may reveal information that can be used to exploit vulnerable points of a system:[41] by knowing the security policies protecting the intelligence gathered by the EU NAVFOR, for instance, terrorists would know who are the parties that might be aware of their activities, and in which circumstances. Furthermore, by ac-

cessing a policy an adversary would know what credentials he needs to forge to illegitimately gain access to a resource.[20] Therefore, next to the disclosure of data, protecting the disclosure of security policies is highly desirable.
The distributed and dynamic nature of SoS introduces two further challenges to the confidentiality and integrity requirements. More precisely, policies that regulate the access to information need to:

- take into account that parties may not know each other beforehand. Therefore, authorizations cannot (always) be defined based on the identity of the requester; rather, they should be based on his certified attributes (e.g., nationality, rank).
- be flexible and adaptable to different circumstances. In this respect, the evaluation of access requests should take the current context into consideration (e.g., the criticality of a situation).

Whereas in some cases parties might explicitly enumerate the authorities they trust for the certification of a certain attribute, in other circumstances they might rely on *chains of trust*. In policy MSC_3 in Table 1, for instance, the MSC does not state explicitly which authorities it trusts to certify SAR vessel operators; on the contrary, it simply requires the certifying authority to be a member of the EU NAVFOR. Hence, when evaluating the policy, to determine whether to accept a credential certifying a SAR vessel operator, the MSC has to verify whether the credential issuer is certified as a member of the EU NAVFOR by the EU. In other words, the MSC attempts to construct a chain of trust that determines the validity of a credential. This problem is similar to the problem of validating digital certificates in a public key infrastructure.

(2) *Autonomy of the parties involved*: the dynamicity of SoS implies that cooperations are often short-lived and the systems involved constantly change over time. In addition, the parties in an SoS are autonomous and have individual objectives next to the ones of the coalition, and may be involved in more than one SoS at a time. In this setting, we cannot expect the parties in an SoS to employ common data and organizational models and vocabularies for the specification of their security policies; rather, parties should be able to employ different models and vocabularies.

(3) *Interoperability among parties*: despite the heterogeneity in their data and organizational structures and vocabularies, parties must be able to understand each other for the success of the coalition. In our scenario, for example, it is evident that CG-1 should be certified by the Dutch navy as a "SAR vessel", even though it is defined as "SAR lifeboat" by the Dutch coastguard (policy CG_1 in Table 1). Standards are often employed to enable interoperability in distributed systems (e.g., XACML[36] and SAML[35]). Although representing a valid solution as communication protocols, however, the use of standards does not solve the problem of semantic misalignment of the parties' vocabularies.

(4) *Ease of use and deployment into existing systems*: the services and functional-

ities offered by the systems in an SoS have strong implications on the design of the security components. On the one hand, the functional components of a system should be designed and implemented as independently as possible from its security components. On the other hand, however, a security framework for SoS should be easy to integrate into existing systems and should easily interface with the system's functionalities to protect the system's confidential information. In addition, the framework should be flexible in order to facilitate its extension with additional security components that may become relevant during the lifetime of an SoS (e.g., a reputation system for service selection). Further to being easy to deploy, a security framework for SoS should be easy to use.[11] In particular, it should be easy to configure by the security administrator with the appropriate security policies and security parameters, which may change rapidly due to the dynamicity of the domain. Moreover, the framework should be as automated as possible in order to require minimal human intervention.[31]

Clearly, these requirements do not cover all the security aspects that are relevant for a distributed system. In the context of the Poseidon project, however, the focus is mainly on designing a solution that satisfies the requirements which are characteristic for SoS. A more complete list of requirements for security frameworks for distributed systems is provided in Ref. 18, 31. Nevertheless, many of these requirements can be seen as security solutions rather than desiderata. The call for an access control system and a mechanism for establishing trust among parties,[18] for example, represent possible solutions that can be employed to protect the confidentiality of information. Similarly, the need for a semantically rich framework,[31] for a mapping between semantics and openness to standards[18] can be seen as solutions to the autonomy and interoperability requirements. Other requirements proposed by the authors, on the other hand, are out of the scope of this paper. Examples of such requirements are the protection from network attacks (e.g., denial of service, eavesdropping, identity spoofing), network communications security, and the guarantee of service availability.

In the next section we review related work on security frameworks for SoS, and show that none of the existing frameworks satisfies the four requirements identified in this section. Then, in Sections 4 to 6 we present our solution.

3. Related Work

Interest in the development of security frameworks for SoS has increased in the last years, as more sensitive information needs to be exchanged among distributed parties (e.g., over the Internet). In this section we review some of the security frameworks presented in the literature, dividing them into two categories: TM frameworks and semantic frameworks. For each framework, we discuss which of the requirements presented in Section 2.2 it satisfies. Notice that, since the deployability of a framework is strictly dependent on its implementation and no implementation details are

Table 2. Requirements Satisfied by the Existing Security Frameworks for Distributed Systems

Framework		Requirement			
		P	A	I	E
TM	RT[29]	-	X	√	
	Cassandra[1]	-	X	√	
	PeerTrust[32]	-	X	√	
	Tulip[13]	-	-	-	
	X-GTRBAC[3]	√	-	-	
	TrustBuilder2[27]	-	√	-	
Semantic	ROWLBAC[19]	-	√	√	-
	REI[25]	-	√	√	-
	KAOS[48]	-	√	√	-
	Kolter et al.[26]	-	√	√	√

P = Protection of Data and Policies; A = Autonomy; I = Interoperability;
E = Ease of Use and Deployment

√ = Satisfied; "-" = Partially Satisfied; "X" = Not Satisfied

provided for most of the frameworks in the literature, a discussion on the ease of integration of the security frameworks into existing SoS is mostly omitted. For the same reason, no observation on the usability of the frameworks can be made. A summary of the result of our study is given in Table 2.

TM frameworks provide a way of establishing trust among unknown parties based on their attributes, certified by means of credentials. TM frameworks focus mainly on the protection of data, possibly allowing for the specification of context-dependent access rules; the confidentiality of security policies, however, is mostly disregarded. Tulip[13] and RT,[29] for instance, rely on a centralized policy evaluation strategy which requires the policies of all parties to be collected in a single location for evaluating access requests and deriving chains of trust. On the contrary, Cassandra[1] and PeerTrust[32] address policy confidentiality by employing a fully distributed policy evaluation algorithm. Cassandra, however, does not guarantee the termination of the computation; this is not acceptable for an access control system, which should always take a (positive or negative) decision. PeerTrust, on the other hand, requires the dependencies among the policies of all parties to be known to the policy evaluator, making the policy confidentiality requirement only partially fulfilled. In addition, similarly to most TM languages (e.g., see Ref. 7, 51), Cassandra and PeerTrust assume a complete agreement among the parties on the vocabulary used for the specification of policies. Consequently, they ensure confidentiality of information and interoperability among parties at the cost of autonomy.

TrustBuilder2[27] is a fully-configurable TM framework whose components can be added to the system as plug-ins; in this way, each party can employ the policy language and policy evaluation algorithm he considers more appropriate. Even though this increases the autonomy of parties, their interoperability cannot be guaranteed, as it entirely depends on the plug-ins chosen by each party. To enhance interoperability, X-GTRBAC[3] and Tulip employ an XML-based exchange syntax for policies, which facilitates their automatic processing. The use of XML also enables parties to refer to different vocabularies for the specification of their policies. However, this approach is purely syntactical and does not provide the semantics necessary to guarantee mutual understanding between parties.

In contrast, semantic frameworks enable both autonomy and interoperability by providing a semantics to the terms used in security policies. Most of these systems rely on ontologies for the specification of policies (e.g., ROWLBAC,[19] REI,[25] KAOS[48]). An ontology is a characterization of a domain in terms of concepts and relationships, each with a precise semantics; therefore, an ontology can be used to represent the knowledge base of a party. Each party can refer to a different ontology for the specification of his policies; semantic alignment techniques[16,22,33,34] can then be employed to align their vocabularies, enabling mutual understanding among collaborating parties. Most of the existing semantic alignment techniques compute the degree of semantic resemblance between concepts in different ontologies based on the structure of the ontologies where those concepts are defined. These techniques, however, either assume parties to employ a common ontology schema[22] (thus limiting autonomy), or require complete knowledge of the ontologies to be aligned[16,33,34] (relinquishing confidentiality). None of these assumptions is acceptable for our scenario, where parties do not know each other a priori and might want their knowledge base (or part of it) to remain private. Furthermore, the expressive power offered by semantic frameworks is limited by the underlying ontology language (e.g., OWL[52]), and does not allow the specification of several types of security constraints, as for instance separation of duty. Since such constraints are common to many application domains for SoS (e.g., MSS, business-to-business), these languages do not provide a valid solution for our scenario. To overcome this limitation, ontology languages have been extended with rules,[23] but this extension causes the policy reasoning to become undecidable.[39] On the other hand, the use of ontology languages for policy specification allows for an easy integration of security policies into the semantic web; this can be used to protect data in SoS such as web services.

The work that is closest to our proposal is the security framework for web services introduced by Kolter et al.[26] This framework uses an attribute-based access control model to protect the confidentiality and integrity of information. A prototype implementation of the framework is presented in Ref. 17; similarly to ours, it is strongly inspired by the XACML framework[36] and uses the XACML and SAML[35] standards for the exchange of messages among parties. The framework employs ontologies to define the semantics of the terms used in policies; an ontology alignment technique enables parties to use different ontologies, guaranteeing both autonomy and interoperability. To perform semantic alignment and policy evaluation, however, the framework requires the ontologies and policies of all parties to be public. Consequently, the confidentiality requirement is partially disregarded. Since it relies on standards for communication and it is built according to the service-oriented architecture paradigm of XACML, the framework should be easy to integrate into existing SoS.

4. Overview of the Solution and Framework's Ingredients

In order to satisfy the requirements introduced in Section 2.2, the security framework proposed in this paper combines models and techniques from the fields of computer security, knowledge representation, and software engineering. In particular, it relies on:

- Context-aware access control and trust management (TM) to protect the confidentiality and integrity of information. Context-aware access control is used to tackle the dynamicity of SoS: by incorporating context information (e.g., the location of the requester, the criticality of the situation) in access decisions, parties can specify flexible policies which adapt to different situations. TM, on the other hand, deals with the distributed nature of SoS. In TM, access decisions are based on the attributes of a requester (e.g., vessels of the EU NAVFOR), which are certified by means of digital credentials issued by an authority (i.e., any party in the SoS). The contribution of this approach is twofold: (a) contrarily to identity-based approaches, grounding an access decision on the certified attributes of a requester allows parties to exchange information with (previously) unknown entities; (b) each party can choose which authority to trust to certify which attributes, and accept only credentials issued by that authority. For example, the Danish navy may trust only the EU for certifying the member countries of the EU NAVFOR.
  As mentioned in Section 2.2, to determine the validity of a credential it might be required to derive a chain of trust among the certifying authorities. To address this problem, we have designed and developed a novel policy evaluation algorithm for TM, called GEM.[45] Contrarily to many of the existing algorithms (e.g., see Ref. 13, 29), GEM retrieves the credentials necessary to construct a chain of trust in a completely distributed way without disclosing the security policies of parties, thereby preserving their confidentiality.
- Ontology-based services to enable autonomy and interoperability among the parties in an SoS. More precisely, parties refer to ontology concepts, relationships, and instances to assign a semantics to the terms used in their policies and to describe their data and organizational structures. This, combined with the use of a semantic alignment techniques that maps concepts and instances from different ontologies, allows parties to use the vocabulary and models they consider more appropriate within their system (thus accommodating parties' heterogeneity), while preserving mutual understanding with the rest of the coalition.
- A service-oriented architecture to allow for an easy integration and deployment into existing systems. Our security framework is implemented as a web service; consequently, it can be easily plugged into existing systems as a proxy server that intercepts all the outgoing and incoming messages of a system. Furthermore, each component of the security framework is also implemented as a service, which is programmed to interface with the functional components and

the data stored within a system. This modular approach promotes the reusability of components and facilitates the extension of the framework with additional security services that may become relevant during the lifetime of an SoS (e.g., a reputation system for service selection, or a key performance indicator service).[6] To enhance the usability of the framework we provide a policy editor that assists security administrators in the specification and management of security policies. The editor provides features like syntax highlighting, syntax check, and auto-completion of terms defined in a party's vocabulary.

In the next two subsections we introduce two of the framework ingredients. In particular, in Section 4.1 we present the policy language that each party can use to specify security policies, which integrates context-aware access control with trust management and ontologies. In Section 4.2 we introduce a semantic alignment technique for mapping ontology concepts, relationships, and instances in an SoS. Then, in Section 5 we present the security framework's architecture and show how these ingredients are combined into a unified security framework; the implementation of the framework is discussed in Section 6.2.

4.1. *Policy Language*

In this section we propose a context-aware, ontology-based policy specification language inspired by constraint Datalog,[28] which has been proposed as a foundation language for TM systems. A *policy* consists of a set of Horn clauses of the form:

$$H \leftarrow B_1, \ldots, B_n, c.$$

where H is an atom called *head*, and $B_1, \ldots, B_n, c$ (with $n \geq 0$) is called *body*, where $B_1, \ldots, B_n$ are literals (i.e., positive or negative atoms) and c is a *constraint*. A clause with an empty body is called *fact* (the "$\leftarrow$" symbol is omitted in facts). Policies are specified using four constructs:

- *Ontology atoms*: are used to query the knowledge base represented by ontologies; they have the form $conceptURI(i)$ or $relationshipURI(i_1, i_2)$, where $conceptURI$ and $relationshipURI$ identify respectively a concept and a relationship in an ontology. $conceptURI(i)$ holds if i is an instance of $conceptURI$; $relationshipURI(i_1, i_2)$ holds if instance i_1 is related to instance i_2 via $relationshipURI$.
- *Credential atoms*: represent digitally signed statements made by an issuer about an attribute of a subject; they have the form $\mathsf{cred}(issuer, att, subject)$, where $issuer$ and $subject$ are unique identifiers of a party (e.g., public keys), and att is a $conceptURI$.
- *Authorization atoms*: denote the permission of a subject to perform an action on an object; they have the form $\mathsf{perm}(subject, action, object)$, where $subject$ is the unique identifier of a party, $action$ is a $conceptURI$, and $object$ is a resource identifier (e.g., a URI).

- *Constraints*: are quantifier-free formulae from some fixed *constraint domain*, which is a language of first order formulae containing at least `true`, `false`, and the identity predicate "=" between terms. To guarantee termination of policy evaluation, we allow only constraints from constraint-compact domains (e.g., equality constraints).[42]

All the information collected by the various sensors (e.g., ships' location) and the knowledge derived by the services within an SoS (e.g., suspicious events in the operation area) is stored in an ontology as an instance of an ontology concept (e.g, a concept *Actor*) or relationship (e.g., a relationship *hasPlace*). Thus, parties can obtain the context and domain information relevant for the evaluation of their policies by querying the appropriate ontology. Notice that the use of *conceptURI* in ontology atoms differs from its use in credential and authorization atoms: while in ontology atoms it is used to access the knowledge base at the specified URI, in credential and authorization atoms it is used to characterize the semantics of attributes and actions, i.e., to denote the vocabulary in which the term is defined.

We distinguish two types of policies: *credential policies* and *authorization policies*, which are sets of *credential clauses* and *authorization clauses* respectively. Intuitively, the type of the clause is determined by the atom in the head. The body of a credential clause can contain credential atoms, positive and negative ontology atoms, and constraints; the body of an authorization clause can contain credential atoms, authorization atoms, positive and negative ontology atoms, and constraints.

We do not allow ontology atoms to appear in the head of clauses, as this may lead to the introduction of inconsistencies and cause the ontology reasoning to become undecidable.[23,39] Another restriction that we pose concerns the use of negation. In particular, we restrict negation to ontology atoms, because negating a credential or authorization atom might lead to undesired situations: it would be sufficient for the requester to "hide" a credential to obtain a permission or a credential that should not be granted to him. Negation in the body of credential clauses, even though allowed, should be used carefully. In fact, a credential can be seen as a token that, once issued, stays valid for a certain period of time; negation should thus be limited to ontology atoms whose truth value is not expected to change in the near future. On the contrary, authorization clauses are evaluated to determine whether a permission should be granted at a certain moment in time, and changes in the truth value of an atom in the body only affect future access decision, without impacting past decisions.

Table 3 presents how the policies in Table 1 can be specified in the proposed language. Clauses MSC_1, MSC_2, MSC_3, DK_2, and DK_3 are authorization clauses, while clauses EU_1, EU_2, EU_3, IT_1, DK_1, NL_1, and CG_1 are credential clauses. The concepts and relationships used in the clauses are defined in different ontologies; in particular, *eu* and *msc* represent the ontology of the European Union (EU) and of the MSC respectively, *sem* refers to the Simple Event Model (SEM) ontology,[50] and *dk* represents the ontology of the Danish navy. Following the XML conven-

Table 3. Policies in Table 1 Specified in the Policy Language

EU_1	cred('EU',eu:NAVFOR_Member,'IT').
EU_2	cred('EU',eu:NAVFOR_Member,'DK').
EU_3	cred('EU',eu:NAVFOR_Member,'NL').
MSC_1	perm(X,msc:Read,Y) $\leftarrow$ cred('EU',eu:NAVFOR_Member,Z), cred(Z,eu:NAVFOR_Vessel,X), not(msc:hasClassification(Y,'secret')).
MSC_2	perm(X,msc:Read,Y) $\leftarrow$ cred('EU',eu:NAVFOR_Member,Z), cred(Z,eu:NAVFOR_Vessel,X), eu:hasTask(X,eu:law_enforcement).
MSC_3	perm(X,msc:Read,Y) $\leftarrow$ cred('EU',eu:NAVFOR_Member,Z), cred(Z,eu:SAR_Vessel,X), sem:hasActor(Y,W), sem:hasAnomalyFactor(W,F), $F > 0.7$.
IT_1	cred('IT',eu:NAVFOR_Vessel,'IT-1').
DK_1	cred('DK',eu:NAVFOR_Vessel,'DK-1').
DK_2	perm(X,dk:Read,Y) $\leftarrow$ cred('EU',eu:NAVFOR_Member,Z), cred(Z,eu:NAVFOR_Vessel,X), not(dk:SecretInformation(Y)).
DK_3	perm(X,msc:Read,Y) $\leftarrow$ cred('EU',eu:NAVFOR_Member,Z), cred(Z,eu:NAVFOR_Vessel,X), eu:hasTask(X,eu:law_enforcement).
NL_1	cred('NL',eu:SAR_Vessel,X) $\leftarrow$ cred('CG',eu:SAR_Vessel,X).
CG_1	cred('CG',eu:SAR_Lifeboat,'CG-1').

tion, prefixes denote namespaces (i.e., URIs); for instance, prefix *sem:* stands for *http://semanticweb.cs.vu.nl/2009/11/sem/*.

In the authorization clauses in Table 3, the object parameter of the head atom represents the unique identifier of the data item which the requester wants to access (e.g., details about the cargo of a ship). Clause MSC_3 is an example of clause combining all four language constructs. This clause states that a subject X is authorized to read an object Y if: X is a *SAR_Operator* certified by a *NAVFOR_Member* Z, object Y refers to an actor W (e.g., a ship), and an anomalous behavior has been registered for this actor, characterized by an "anomaly factor" greater than 0.7 (e.g., the ship is drifting). Concepts *NAVFOR_Member* and *SAR_Vessel* are defined in the EU ontology, concept *Read* in the MSC ontology, and relationships *hasActor* and *hasAnomalyFactor* are defined in the SEM ontology.

4.2. *Semantic Alignment*

The policy language introduced in Section 4.1 enables parties to provide a precise semantics to the terms used in their policies by referring to ontologies. Nevertheless, this is not enough to guarantee interoperability in an SoS, as the use of different vocabularies (i.e., different ontologies) might still cause problems of mutual understanding. For example, in the policies in Table 3, the Dutch navy and the Dutch coastguard use different terms to denote similar concepts (i.e., *eu:SAR_Vessel* and *eu:SAR_Lifeboat* respectively).

A possible solution to the problem of interoperability is the definition of a complete and precise semantic alignment between the vocabularies employed by all parties in the SoS. This solution, however, is unrealistic since in many cases it might not be possible to find perfectly matching terms in two parties' vocabularies. Furthermore, in short- and mid-term cooperations this solution would be too costly and time-consuming. In Ref. 44, we propose an alternative solution to the problem of semantic alignment that allows parties to use different vocabularies for the specification of their policies while preserving mutual understanding. Here, we provide an overview of the solution, and refer to Ref. 44 for a more detailed description.

The proposed semantic alignment technique is based on the concept of *similarity*, which expresses the degree of semantic resemblance between two ontology concepts.[33,34] In our approach, similarity is asserted by means of *similarity credentials*, which may be issued by any principal in the SoS and are represented in the policy language introduced in Section 4.1 by the following construct:

- *Similarity credential atoms*: express the degree of similarity between two ontology concepts (or instances) in the view of the similarity credential's issuer; they have the form $\mathsf{sim}(issuer, att_1, att_2, degree)$, where *issuer* is the unique identifier of a party, att_1 and att_2 are *conceptURI*s (or *instanceURI*s), and *degree* is a value in the range $[0, 1]$, where 0 indicates complete dissimilarity between att_1 and att_2 and 1 indicates semantic equivalence.

Every party may issue a similarity credential for any pair of concepts. Since the similarity degree computed by each party is based on partial knowledge (part of the ontologies to be aligned might not be public) and might be computed using different similarity metrics (e.g., see Ref. 12, 15, 33, 34), the similarity credentials issued by different parties may report different similarity degrees. In other words, similarity credentials represent estimations rather than precise measures. Furthermore, in an SoS not all the parties are equally reliable for the estimation of similarity. Therefore, to better approximate the similarity between two attributes, parties could collect and combine the statements issued by several parties, weighing them by the reliability of their issuer.

In many cases, however, the reliability of a party in an SoS cannot be assessed in advance as parties may not know each other beforehand. The question that we need to address is thus how to combine similarity statements when the reliability of the information source is unknown. Our solution estimates the reliability of a party by computing the *accuracy* of his statements. The accuracy of (the statements issued by) a party is computed by comparing all the statements issued by the party with the similarity credentials about the same attributes issued by the other parties in the SoS; the smaller the difference between the degrees indicated by the party and the degrees reported by the other parties, the higher his accuracy.

Initial accuracy values can be tuned to bias the similarity computation, giving more weight to the statements of some issuers. This could be useful, for instance, in an SoS characterized by the presence of domain vocabulary experts (e.g., special-

Table 4. Example of Similarity Credentials and Policy Using Similarity Constraints

S_1	sim('EU',eu:law_enforcement,eu:patrolling,0.8).
S_2	sim('DK',eu:law_enforcement,eu:patrolling,0.9).
$MSC_{2'}$	perm(X,msc:Read,Y) ← cred('EU',eu:NAVFOR_Member,Z), cred(Z,eu:NAVFOR_Vessel,X), eu:hasTask(X,T), similar(T,eu:law_enforcement) ≥ 0.85.
S_3	sim('EU',eu:SAR_Vessel,eu:SAR_Lifeboat,0.9).
S_4	sim('CG',eu:SAR_Vessel,eu:SAR_Lifeboat,0.7).
NL_1	cred('NL',eu:SAR_Vessel,X) ← cred('CG',Y,X), similar(Y,eu:SAR_Vessel) ≥ 0.75.

ized maritime vocabulary alignment services), which are expected to provide more accurate statements than the other parties in the SoS.

Finally, we introduce the function $\mathsf{similar}(att_1, att_2)$ that computes the degree of similarity between att_1 and att_2 by combining the similarity credentials about the two attributes issued by several parties, weighing the similarity degrees indicated in these credentials based on the accuracy of their issuer. The similarity function can be used to express *similarity constraints* of the form:

$$\mathsf{similar}(att_1, att_2) \geq \mathit{threshold} \tag{1}$$

where *threshold* is the minimum required degree of similarity between attributes att_1 and att_2. Using this constraint, a party enables interoperability with parties using different vocabulary and increases the flexibility of his policies by accepting credentials about possibly unknown attributes, provided that they are similar to a known attribute for at least a certain degree. The threshold value can be tuned depending on the sensitivity of the information protected by a policy and on how broad the known attribute is. In general, higher threshold values should be used for critical information and very specific attributes; for less critical data or broad concepts we can rely on a lower threshold. For instance, the term "SAR vessel" employed by the Dutch navy in its policy (policy NL_1 in Table 3) is quite broad. A not too high threshold could thus be employed to enable the certification of distinct types of SAR vessels (e.g., lifeboats). Setting the right threshold is a challenging task, and is based on heuristics and user expertise rather than on precise rules. Intuitively, the higher the threshold, the lower the risk of disclosing information to unauthorized parties. The accuracy of the threshold value can be assessed through an inspection of the authorized and denied accesses. In case of a high number of false positives (i.e., undesired authorizations granted), the threshold value should be increased; on the contrary, the presence of false negatives (i.e., wrongly denied accesses) indicates that the current threshold might be too high.

Examples of similarity credentials and policies using similarity constraints are shown in Table 4. Consider the first two similarity credentials S_1 and S_2, issued by the EU and the Danish navy respectively, and policy $MSC_{2'}$ specified by the MSC. Assume that the MSC wants to determine whether the Italian navy, which

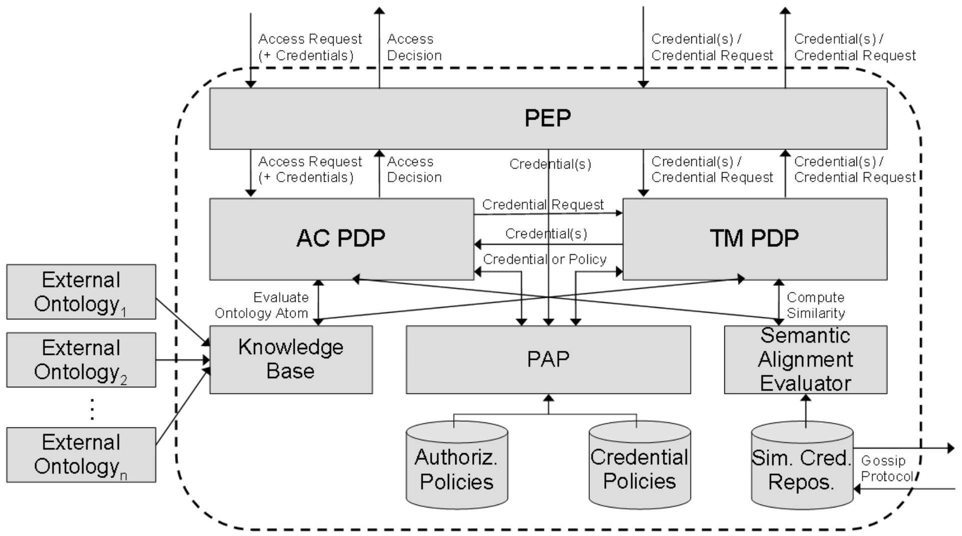


Fig. 2. Security Framework Architecture (Ref. 47)

has patrolling tasks in the EU NAVFOR (see scenario in Section 2.1), should be authorized to access "off the record" information about vessels. Then, the MSC needs to determine whether concepts *eu:patrolling* and *eu:law_enforcement* have a degree of similarity of at least 0.85. To compute the similarity between the two concepts, the MSC has to first calculate the accuracy of the EU and the Danish navy, possibly setting different initial accuracy values depending on whose statements the MSC wants to give more weight to. In our scenario, for example, if the computation results in the same accuracy values for the EU and the Danish navy, the Italian navy would be authorized by the MSC to access "off the record" information about vessels; in fact, the similarity degree returned by function `similar` would be 0.85, which is equal to the threshold. On the contrary, if the accuracy value of the EU were higher than that of the Danish navy (e.g., because the MSC assigns a higher initial accuracy value to the EU), then the Italian navy would not be authorized to access the information, since the similarity function would return a value which is lower than 0.85.

5. The Security Framework

This section presents the security framework that can be employed by each party in an SoS to protect the local resources. An overview of the framework's architecture is shown in Figure 2, where the dashed line separates the local components (i.e., the trusted environment of a party) from the external world. The framework consists of a set of core components (i.e., *Policy Enforcement Point*, *Access Control* and *Trust Management Policy Decision Points*, and *Policy Administration Point*, respectively denoted as PEP, AC PDP, TM PDP, and PAP in Figure 2) and a number of

```
<samlp:AttributeQuery
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:eu="http://www.example.org/ontologies/eu">
   <saml:Issuer>MSC</saml:Issuer>
   <saml:Attribute Name="sem:Role">
      <saml:AttributeValue xsi:type="xs:string">
         eu:NAVFOR_Member
      </saml:AttributeValue>
   </saml:Attribute>
</samlp:AttributeQuery>
```

Fig. 3. Credential Request in SAML

complementary specialized services used to assist the policy evaluation process (i.e., *Knowledge Base* and *Semantic Alignment Evaluator*). To facilitate the integration of the framework into existing systems, all the components and services have been implemented following the service-oriented architecture paradigm.[38] The set of core components includes the main components of the XACML architecture,[36] with the addition of the TM PDP for the evaluation of TM policies. The combination of these components with the specialized services ensures protection of information, while preserving autonomy and interoperability among parties in the SoS. In the next paragraphs we discuss each component and service in detail and present some examples of how it can be used in practice, referring to the scenario introduced in Section 2.1.

Policy Enforcement Point. The PEP is the interface of a party with the external world, and has three main tasks: (1) intercepting incoming requests for local resources, (2) contacting the appropriate PDP to evaluate those requests, and (3) enforcing the decision of the PDP. We consider two types of requests: *access requests* and *credential requests*, which are specified in XACML and SAML[35] respectively. We rely on XACML and SAML for the specification of request and response messages for two reasons: first, the use of standards for communication enhances the interoperability among the parties in an SoS; second, it allows for an integration of the security framework into existing systems employing those standards, such as for instance the TAS3 system.[6]

Access requests are requests for data controlled by the local party (e.g., message 2 in Figure 1), and are expressed as XACML `AuthzDecisionQuery`. Access requests can be accompanied by credentials (hereafter called *supporting* credentials) that are relevant for the evaluation of the request; in this case, the PEP sends these credentials to the PAP, which adds them to the local *credential policies*. Supporting credentials are incorporated into access requests by means of the SAML profile of XACML.[37]

Credential requests are requests for credentials issued by the local party, and are expressed as SAML `AttributeQuery`. A drawback of SAML is that it requires

```
<samlp:Response
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:eu="http://www.example.org/ontologies/eu">
  <saml:Issuer>EU</saml:Issuer>
  <samlp:Status>
    <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </samlp:Status>
  <saml:Assertion ID="c17">
    <saml:Issuer>EU</saml:Issuer>
    <saml:Subject>
      <saml:NameID>IT</saml:NameID>
    </saml:Subject>
    <saml:Subject>
      <saml:NameID>DK</saml:NameID>
    </saml:Subject>
    <saml:Subject>
      <saml:NameID>NL</saml:NameID>
    </saml:Subject>
    <saml:AttributeStatement>
      <saml:Attribute Name="sem:Role">
        <saml:AttributeValue xsi:type="xs:string">
          eu:NAVFOR_Member
        </saml:AttributeValue>
      </saml:Attribute>
    </saml:AttributeStatement>
  </saml:Assertion>
</samlp:Response>
```

Fig. 4. Credential Response in SAML

the *saml:Subject* XML element to be always specified in a credential request, hence preventing parties to request the certificates of all the subject having a given attribute (e.g., all the member countries of the EU NAVFOR). To overcome this limitation, we have modified SAML to accept also requests where the *saml:Subject* XML element is not specified. An example of such credential request is presented in Figure 3: it represents the request made by the Maritime Security Center (MSC) to the EU for the credentials certifying the EU NAVFOR members (i.e., message 8a in Figure 1). Notice that the XML element *saml:Issuer* refers to the party who is issuing the request, and not to the issuer of the requested credential. In SAML, the credential issuer is always the party receiving the request.

Upon receiving a request, the PEP parses it and forwards it to the appropriate PDP. In particular, access requests are processed by the AC PDP, while credential requests by the TM PDP. The decision on a request may depend on credentials which are not locally available and, consequently, need to be retrieved from some other party. The PEP is responsible of forwarding requests for missing credentials to the appropriate parties (either the initial requester or a third party), and feeding the response back to the TM PDP.

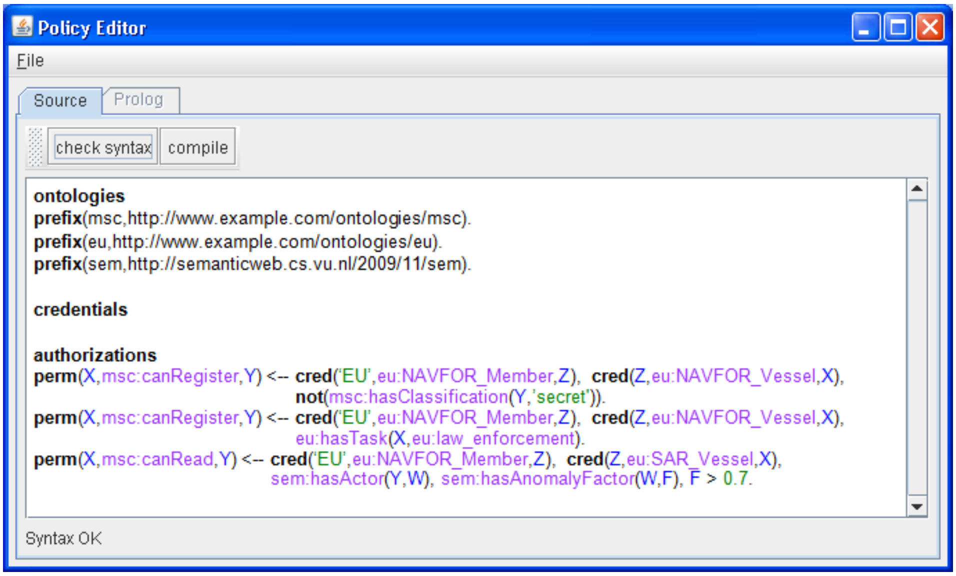
Finally, the PEP enforces the decision of the PDP. If the decision is positive, the PEP grants access to the requested resource (in case of an access request) or issues the requested credential(s) (in case of a credential request); otherwise, a "deny" response is sent to the requester. Response messages are expressed as XACML `Response` and SAML `Response`. To enable parties to send a list of all the subjects having a given attribute, SAML has been modified to accept multiple *saml:Subject* XML elements. Figure 4 presents the response sent by the EU to the MSC (message 8b in Figure 1).

Access Control Policy Decision Point. The AC PDP is responsible for the evaluation of access requests. When receiving an access request, the AC PDP fetches the relevant authorization clauses through the PAP, and checks whether they depend on some credentials. If this is the case, the AC PDP requests the credentials to the TM PDP, which takes over the responsibility of retrieving them. Then, depending on whether all the necessary credentials have been successfully retrieved and the other conditions in the authorization clauses are satisfied, the AC PDP determines whether the access request should be authorized or denied. The evaluation of ontology atoms (i.e., queries to the knowledge base) and similarity functions in the authorization clauses are resolved by invoking the Knowledge Base service and the Semantic Alignment Evaluator respectively.

Trust Management Policy Decision Point. The TM PDP is responsible for the evaluation of credential requests. Similarly to the AC PDP, the TM PDP fetches the applicable credential clauses through the PAP. The policy evaluation algorithm within the TM PDP defines the procedure to compute the answers to a credential request. In our framework we employ GEM[45] as policy evaluation algorithm. GEM evaluates credential requests in a completely distributed way without disclosing the policies of parties, thereby preserving their confidentiality. The algorithm combines a tabling strategy[9] with a deadlock-free "wait" mechanism to detect loops and determine when the computation has terminated (i.e., when all the answers to a credential request have been collected). In particular, tabling is used to keep track for each credential of the previously received requests for that credential, the answers computed so far, and the status of the evaluation; request identifiers are used to detect cyclic dependencies among credentials (i.e., loops). The wait mechanism allows parties to hold until a "maximal" set of answers have been computed before returning them to the requester. In this way, the requester knows that he will receive a response only when all the answers to his request have been collected, simplifying termination detection. As for the AC PDP, ontology atoms and similarity functions in credential policies are resolved by contacting the appropriate service.

Policy Administration Point. The PAP takes care of the management of authorization and credential policies, and mediates the access of the AC PDP and TM PDP to the repositories where these policies are stored. When contacted by one of the PDPs, the PAP loads the policies from the appropriate repository and returns them to the invoking PDP.

To improve the usability of the framework, the PAP provides security administrators with a graphical user interface that allows the editing, modification and



```
ontologies
prefix(msc,http://www.example.com/ontologies/msc).
prefix(eu,http://www.example.com/ontologies/eu).
prefix(sem,http://semanticweb.cs.vu.nl/2009/11/sem).

credentials

authorizations
perm(X,msc:canRegister,Y) <-- cred('EU',eu:NAVFOR_Member,Z),  cred(Z,eu:NAVFOR_Vessel,X),
                              not(msc:hasClassification(Y,'secret')).
perm(X,msc:canRegister,Y) <-- cred('EU',eu:NAVFOR_Member,Z),  cred(Z,eu:NAVFOR_Vessel,X),
                              eu:hasTask(X,eu:law_enforcement).
perm(X,msc:canRead,Y) <-- cred('EU',eu:NAVFOR_Member,Z),  cred(Z,eu:SAR_Vessel,X),
                          sem:hasActor(Y,W), sem:hasAnomalyFactor(W,F), F > 0.7.
```

Syntax OK

Fig. 5. Policy Editor

deletion of policies. Figure 5 presents a screenshot of the policy editor; the displayed clauses represent the policy of the MSC in Table 3. The policy is divided into three parts: *ontologies*, which imports the relevant ontologies into the local Knowledge Base and defines their prefixes, *credentials*, which contains credential clauses, and *authorizations*, which contains authorization clauses. The editor supports security administrators during policy editing by providing features like syntax highlighting, syntax check, and auto-completion of ontology concepts, relationships, and instances. Syntax highlighting and syntax check allow the security administrator to easily identify errors in the policy (e.g., a negated credential atom) by coloring the text that has caused the error in red. The auto-completion function assists administrators in the typing of *conceptURI*, *relationshipURI*, and *instanceURI* by accessing the knowledge base and displaying the list of concepts, relationships, and instances that match the (partially) typed text. Finally, the editor provides automatic conversion of policies into Prolog programs.

Knowledge Base. The Knowledge Base service is used by a party to retrieve the context and domain information that is relevant for an access decision. The Knowledge Base consists of a set of ontologies that define the concepts and relationship employed in the party's policies, the structure of the information he controls (i.e., metadata annotating the local resources), and all the domain and context information (e.g., the current location of a vessel) collected and derived within the SoS. A party can enlarge its Knowledge Base by importing ontologies defined and published

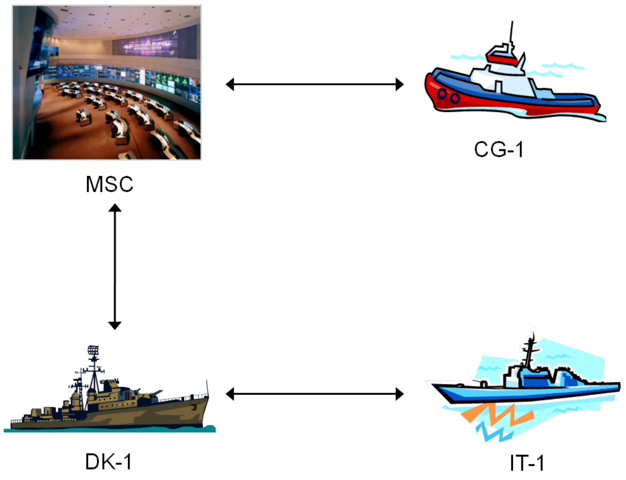


Fig. 6. The Poseidon SoS

by other parties or institutions.

Semantic Alignment Evaluator. This service computes the similarity degree between ontology concepts and instances. It implements the function `similar` presented in Section 4.2 using the similarity credentials available in the *Similarity Credential Repository* (SCR) and the accuracy of their issuers. New similarity credentials can be collected, for instance, using a gossip protocol[21]. The accuracy of parties is recomputed only when new credentials are acquired, to reduce the overhead of the computation. In Figure 2, the Semantic Alignment Evaluator is depicted as a service directly controlled by the local party; however, parties in a coalition may rely on (possibly shared) external semantic alignment services.

6. Framework Validation

We have deployed a prototype implementation of the security framework introduced in Section 5 into an SoS developed within the Poseidon project.[46,47] In this section, we first present the Poseidon SoS, then we discuss the prototype implementation of the security framework, and finally we demonstrate the application of the resulting system to the scenario in Section 2.1.

6.1. *The Poseidon System of Systems*

The Poseidon SoS is a flexible, adaptable, and evolvable SoS in which parties gather information from their surroundings, analyze it, and communicate with each other to achieve situational awareness in the MSS domain. An overview of the Poseidon SoS is given in Figure 6. The SoS consists of four main systems: a patrol vessel of the Italian navy (IT-1), a frigate of the Danish navy (DK-1), the Maritime Security

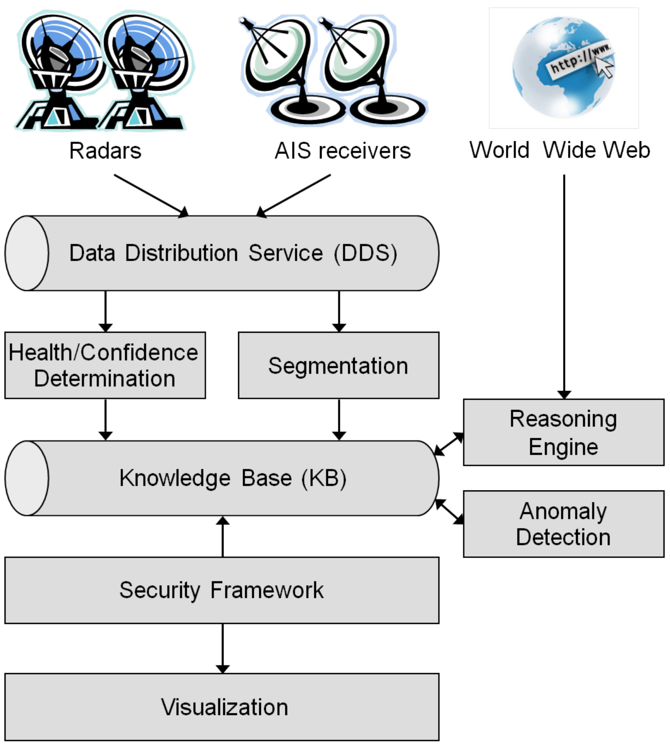


Fig. 7. SoS Systems Architecture

Center (MSC), and a SAR vessel of the Dutch coastguard (CG-1). Each of these systems employs sensors to gather information from its surroundings; in particular, radars and AIS receivers mounted on the vessels are employed to obtain information about the maritime traffic in a given area. AIS receivers are sensors that capture AIS messages broadcasted by the ships transiting in their coverage area and send those messages to the controlling system for further processing. Sensors data are integrated with further information from the Internet and intelligence gathered by the parties in the SoS to obtain more comprehensive information and derive new knowledge about the current context. This information is then presented to the vessels' operators, which monitor the maritime traffic to accomplish their task in the mission.

The process of transforming raw sensor data into comprehensive information that allows the operators of the coalition's vessels to take informed decisions involves several system components. The architecture of each system in the SoS is designed to satisfy four main requirements: scalability, availability, modularity, and robustness to changes in the context (e.g., detecting and reacting to emergency situations) and in the organizational structure of the SoS (e.g., parties joining or leaving the coalition). Figure 7 illustrates the architecture of the systems in detail. The data

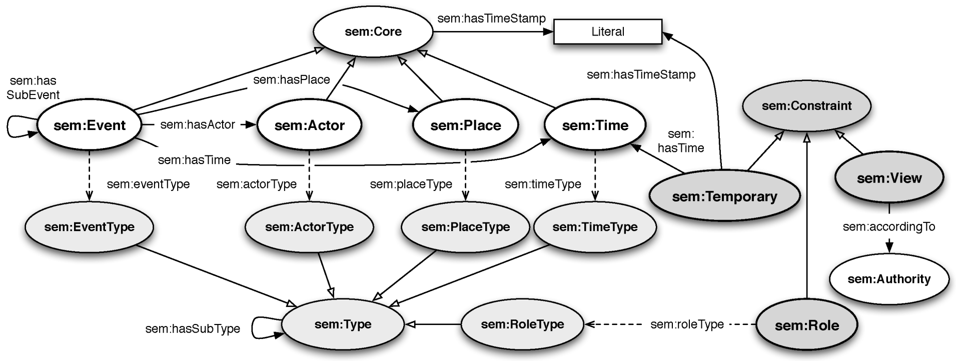


Fig. 8. The Simple Event Model (SEM)

streaming from the various receivers are integrated and made available to the other components of the system through a *data distribution service* (DDS). The DDS is a networking middleware for real-time systems; it implements a publish/subscribe model for sending and receiving data, events, and commands among the components of a system. Components that produce information (publishers) create “topics” (e.g., AIS messages, weather conditions) and publish “samples”. The DDS takes care of delivering the sample to all the subscribers that declare an interest in that topic.

Two components have direct access to the DDS: the *sensor health monitor* and the *AIS message segmenter*. The first component analyzes the AIS messages received from the different sensors with the objective of detecting faulty information sources (e.g., a “stuttering” receiver); the output of this analysis is a confidence value associated to the source of each AIS message. The segmenter performs a compression of AIS messages into segments of consistent behavior, to reduce computation time and memory requirements of higher level components. The resulting “trajectories” are added to the *knowledge base* (KB) together with the confidence value produced by the health monitor.

The KB is structured according to the schema defined by the SEM ontology[50] (Figure 8). The most important concept in this ontology is concept *sem:Event*, which describes events in the maritime domain in terms of “*who* did *what*, *where* and *when*”. Each event is assigned a *sem:EventType* (left part of Figure 8), and is linked to the actor responsible of the event, and to the place and time at which it occurs. A segment is an example of an event; the event actor is the ship to which the segment refers, and the event type is the ship’s behavior characterizing the segment (e.g., slowing down). Security-related concepts and relationships are also represented in the SEM ontology (lower part of Figure 8). In particular, every issuer of a credential is an instance of concept *sem:Authority*, and the roles of the operators of the coalition vessels are instances of concept *sem:Role*.

Only a limited number of interesting ship behaviors can be detected on the sole basis of the information contained in the segments. Richer behavior definition requires the combined knowledge about the ship position, the characteristics of this position (e.g., geographic type), movement features and the ship type. For instance, it is possible to classify a ship as a ferry by observing that it moves back and forth between two ports at regular time intervals. To this end, the *reasoner* component retrieves from the Web more information about ships and geography, and applies deduction rules to derive further knowledge. The outcome of the reasoning process is added to the KB to enrich the sensors data. Similarly, the *anomaly detection* service derives suspicious events in the maritime traffic (e.g., a ship stopping in a traffic lane); this results in an anomaly factor associated to each segment, which is added to the KB as an annotation to the segments data. The resulting KB contains exhaustive information about all the events occurring in a maritime area. More details about the segmenter, the SEM ontology, and the reasoner components can be found in Ref. 50.

Every request to access information contained in the KB of a party (both coming from the party itself or from another system in the SoS) passes through our security framework, whose response depends on the security policy and the attributes of the requester. A *visualization* service is employed by the operators of the various systems to monitor the maritime traffic.

6.2. *Security Framework Implementation*

In the Poseidon SoS Google Earth is used as visualization software. Access to the information contained in the KB and communication among parties (e.g., access and credential requests) is via HTTP. In this setting, the PEP of the security framework acts as a web proxy that intercepts all the HTTP requests to the KB, loads the data, and returns an HTTP response in the appropriate format, based on the policy of the party controlling the data. In particular, response messages for the visualization service are in Keyhole Markup Language (KML) format, while responses to requests for extra information about a vessel are in HTML format.

Accordingly, the PEP has been divided into two modules: an interface module and a services module; the latter consists of a set of responders, one for each service provided by the local party. The PEP interface module waits for incoming access requests, parses them (from XACML and SAML to an internal format), and passes them to the appropriate responder which takes care of processing the request and generating a response. The PEP of each system in the Poseidon SoS has three responders: a KML responder, an HTML responder, and a GEM responder, which processes credential requests. This architectural choice enhances the flexibility of the SoS, as it allows new services to be included in the SoS by simply adding the relative responders to the PEP component.

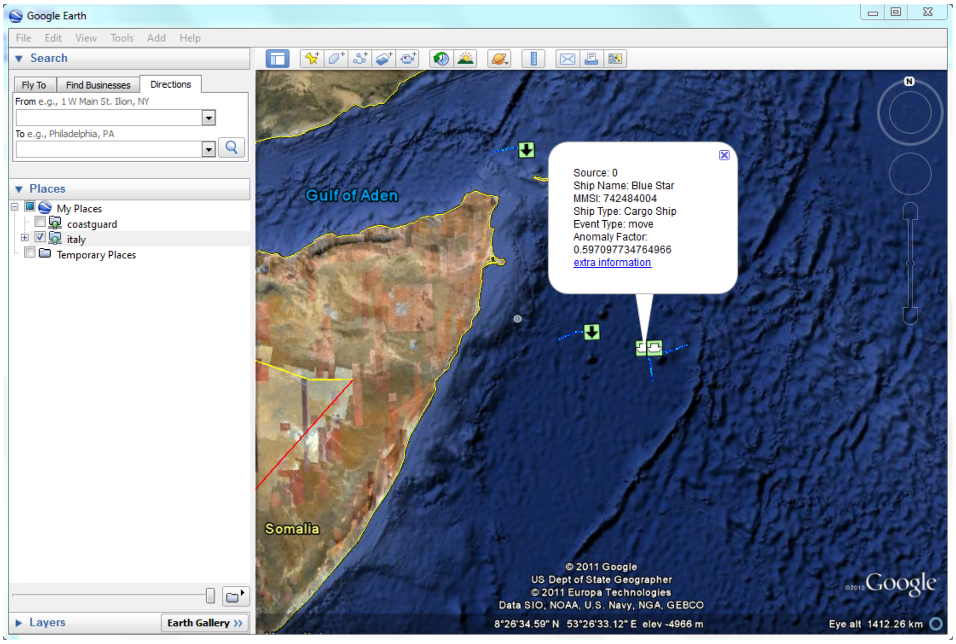
Most of the security framework components have been implemented in Java, except for the AC PDP, the KB, and the similarity credentials repository. In particular, the AC PDP is based on SWI-Prolog. We employ the JPL (Java to Prolog) library as an interface between the PEP and the AC PDP, while the communication between the AC PDP and the semantic alignment evaluator is via HTTP, through the SWI-Prolog HTTP library. Ontology atoms in authorization policies are resolved by requesting their evaluation to the KB by means of the SWI-Prolog Semantic Web library, which consists of packages for reading, querying and storing RDF documents (an ontology can be represented as a set of RDF triples). The TM PDP accesses the KB through the Jena Semantic Web framework for Java, and communicates with the semantic alignment evaluator and the other parties in the SoS by means of an `HttpURLConnection`. Finally, the similarity credentials repository is a MySQL database consisting of two tables: one containing the similarity credentials collected by a party, and the other containing the accuracy of the issuers of those credentials.

6.3. *EU NAVFOR*

The Poseidon SoS has been employed in a demonstrator based on the scenario introduced in Section 2.1. Due to the lack of availability of AIS data off the Somali coast, however, in this paper we display synthetic data generated using the PRESTO software.[24] Figures 9 and 10 present the output of the visualization service and the response to requests for extra information of the operators of IT-1 and CG-1 respectively, based on the policies in Table 3.

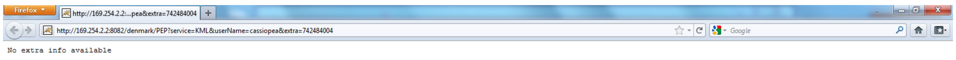
In the visualization, the color of a segment reflects the anomaly factor associated to that segment; the color scale goes from blue to red as the anomaly factor increases. The current position of each ship is represented by an icon, whose type corresponds to the type of the vessel it represents: in particular, icons depicting an arrow pointing downwards represent vessels which are (or become) part of the EU NAVFOR; all the other vessel types are represented by an icon depicting a white boat. By clicking on an icon or a segment the operator can see the name, maritime mobile service identity (MMSI), and type of a ship, as well as the event type (e.g., the ship is slowing down, has stopped, etc.) and anomaly factor corresponding to the last segment relative to the ship and the source that provided the information (i.e., the identifier of the AIS receiver who published the AIS messages from which the segment is computed). For instance, in Figure 9(a), information about the ship Blue Star is displayed. The information box shows that Blue Star is a cargo ship which is currently moving, and has an associated anomaly factor of approximately 0.6. The source of the information is the AIS receiver with identifier “0”.

The first time that the operator of IT-1 (respectively of CG-1) requests to access extra information about a vessel (e.g., the cargo ship Blue Star) from the KB of DK-1 (resp. of the MSC), the security framework of DK-1 (resp. of the MSC) attempts to collect the credentials required to authorize the access. This initiates a credential discovery process involving several parties in the SoS. Then, depending on whether the credential discovery was successful, on the role of the requester


Google Earth
File Edit View Tools Add Help
Search
Fly To
Find Businesses
Directions
From e.g., 1 W Main St. Ilion, NY
To e.g., Philadelphia, PA
Places
My Places
coastguard
italy
Temporary Places
Layers
Earth Gallery
Gulf of Aden
Somalia
© 2011 Google
US Dept of State Geographer
© 2011 Europa Technologies
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
8°26'34.59" N 53°26'33.12" E elev -4966 m
Eye alt 1412.26 km

Source: 0
Ship Name: Blue Star
MMSI: 742484004
Ship Type: Cargo Ship
Event Type: move
Anomaly Factor:
0.597097734764966
extra information

(a) Data View for the Operator of IT-1



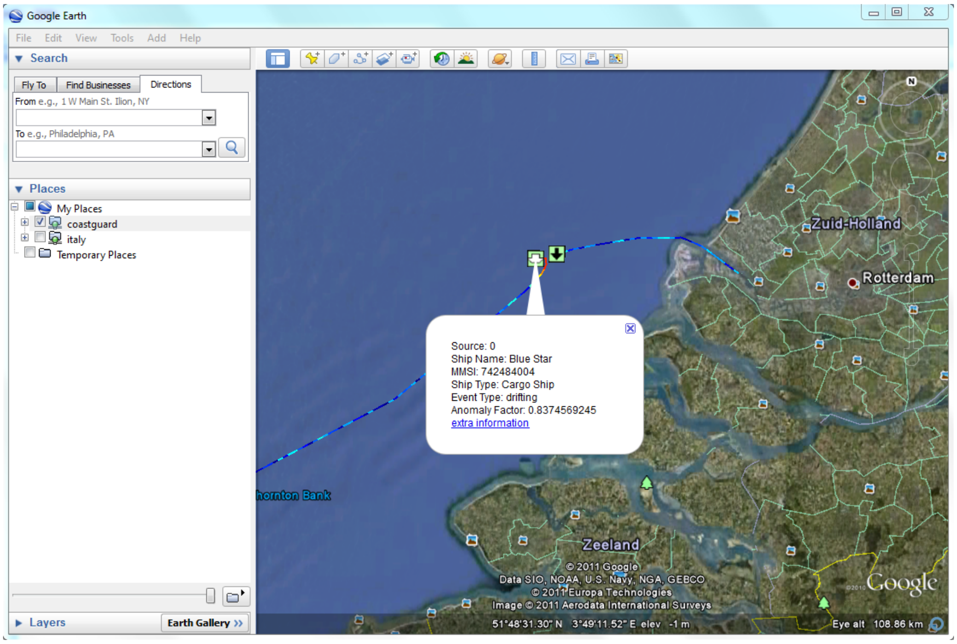
```
http://169.254.2.2:8082/denmark/PEP?service=KML&userName=cassiopea&extra=742484004
No extra info available
```

(b) Extra Information Displayed to the Operator of IT-1

Fig. 9. Data View and Extra Information Displayed to the Operator of IT-1 (Ref. 47)

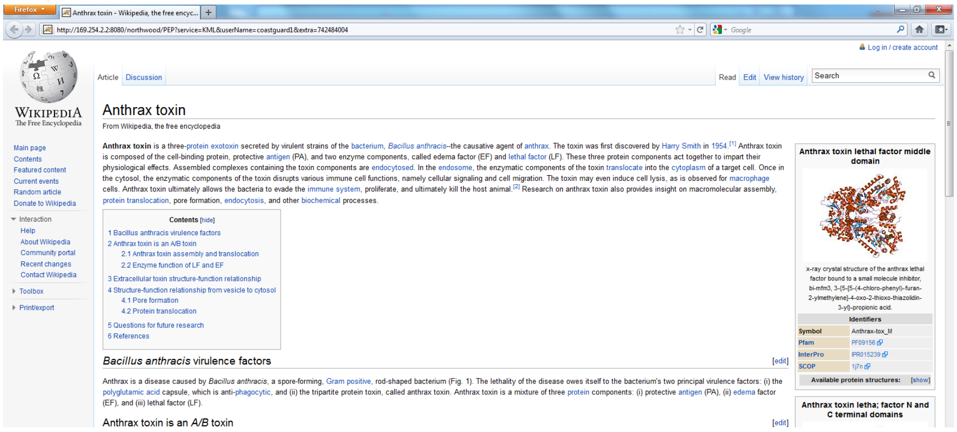
within the coalition, and on the current context, the extra information is provided or denied. For example, since in our scenario IT-1 is not assigned to the prevention of terrorist activites, DK-1 does not provide extra information to the Italian operator (Figure 9(b)). On the contrary, since CG-1 is a SAR vessel certified by the Dutch navy, and Blue Star is in emergency (as can be evinced by the high anomaly factor and the event type - drifting - in Figure 10(a)), extra information about the cargo ship is provided to the coastguard operator. In this case, since Blue Star's cargo contains Anthrax that was meant to be distributed to terroristic cells in Europe, the extra information includes details about the Anthrax toxin and measures to prevent the infection (Figure 10(b)). Notice that the certification of CG-1 as a SAR vessel from the Dutch navy results from the alignment of the vocabularies of the Dutch navy and the Dutch coastguard (see Table 4 for the similarity credentials and policies used for the alignment).

The innovations introduced by our framework with respect to the other existing solutions discussed in Section 3 are represented both by the integration of different


Google Earth
File Edit View Tools Add Help
Search
Fly To
Find Businesses
Directions
From e.g., 1 W Main St. Ilion, NY
To e.g., Philadelphia, PA
Places
My Places
coastguard
italy
Temporary Places
Layers
Earth Gallery
Zuid-Holland
Rotterdam
Thornton Bank
Zeeland
© 2011 Google
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
© 2011 Europa Technologies
Image © 2011 Aerodata International Surveys
51°48'31.30" N 3°49'11.52" E elev -1 m
Eye alt 108.86 km
Google

Source: 0
Ship Name: Blue Star
MMSI: 742484004
Ship Type: Cargo Ship
Event Type: drifting
Anomaly Factor: 0.8374569245
extra information

(a) Data View for Operator of CG-1



http://169.254.2.2:8080/northwood/PEP?service=KML&userName=coastguard1&extra=742484004

Anthrax toxin

From Wikipedia, the free encyclopedia

Anthrax toxin is a three-protein exotoxin secreted by virulent strains of the bacterium, *Bacillus anthracis*--the causative agent of anthrax. The toxin was first discovered by Harry Smith in 1954.[1] Anthrax toxin is composed of the cell-binding protein, protective antigen (PA), and two enzyme components, called edema factor (EF) and lethal factor (LF). These three protein components act together to impart their physiological effects. Assembled complexes containing the toxin components are endocytosed. In the endosome, the enzymatic components of the toxin translocate into the cytoplasm of a target cell. Once in the cytosol, the enzymatic components of the toxin disrupts various immune cell functions, namely cellular signaling and cell migration. The toxin may even induce cell lysis, as is observed for macrophage cells. Anthrax toxin ultimately allows the bacteria to evade the immune system, proliferate, and ultimately kill the host animal.[2] Research on anthrax toxin also provides insight on macromolecular assembly, protein translocation, pore formation, endocytosis, and other biochemical processes.

Contents [hide]

Bacillus anthracis virulence factors [edit]

Anthrax is a disease caused by *Bacillus anthracis*, a spore-forming, Gram positive, rod-shaped bacterium (Fig. 1). The lethality of the disease owes itself to the bacterium's two principal virulence factors: (i) the polyglutamic acid capsule, which is anti-phagocytic, and (ii) the tripartite protein toxin, called anthrax toxin. Anthrax toxin is a mixture of three protein components: (i) protective antigen (PA), (ii) edema factor (EF), and (iii) lethal factor (LF).

Anthrax toxin is an *A/B* toxin [edit]

Anthrax toxin lethal factor middle domain

x-ray crystal structure of the anthrax lethal factor bound to a small molecule inhibitor, bi-mfm3, 3-{5-[5-(4-chloro-phenyl)-furan-2-ylmethylene]-4-oxo-2-thioxo-thiazolidin-3-yl}-propionic acid.

Identifiers	
Symbol	Anthrax-tox_M
Pfam	PF09156
InterPro	IPR015239
SCOP	1j7n
Available protein structures:	[show]

Anthrax toxin letha; factor N and C terminal domains

(b) Extra Information Displayed to the Operator of CG-1

Fig. 10. Data Views and Extra Information Displayed to the Operator of CG-1

techniques, namely context-aware access control, TM, and ontology-based services, and by the techniques themselves. For instance, in our scenario, the authorization of CG-1's operators to access extra information about Blue Star is determined by context information (i.e., emergency situations). Contrarily to other algorithms (e.g., see Ref. 13, 29), GEM, the policy evaluation algorithm used within our framework, derives the credentials required by the operators of IT-1 and CG-1 to access the requested information without disclosing the security policies of the parties in the SoS. Finally, the alignment of the vocabularies of the Dutch navy and the Dutch coastguard is performed in a completely automated way, allowing the Dutch navy to compute the similarity between concepts "SAR vessel" and "SAR lifeboat" on the fly, without requiring knowledge of the ontologies where the two concepts are defined. This is in contrast with the other existing semantic alignment techniques (e.g., see Ref. 16, 33, 34), which require complete knowledge of the ontologies to be aligned. The integration of the proposed techniques results in a comprehensive framework which responds to the needs of an SoS. In addition, from a technical point of view, our solution is characterized by easy deployability into existing SoS. In fact, since the security framework is deployed as a proxy server, parties can connect (or disconnect) the framework to their system by simply redirecting their incoming and outgoing traffic to a different address.

7. Conclusions

Recent years have been characterized by the proliferation of systems of systems (SoS), i.e., coalitions of systems and services that collaborate to achieve a common goal. The security challenges in an SoS are different from those affecting centralized systems. In a dynamic, inter-organizational coalition of systems, parties might not know each other beforehand, might employ different data and organizational models and "speak" different languages. Nevertheless, they must be able to collaborate for the success of the coalition. We have identified four main requirements that a security framework for SoS must satisfy: (1) protection of the *confidentiality* and *integrity* of information, which includes both data and policies; (2) *autonomy* of parties in the choice of data and organizational models and vocabulary used to specify policies and describe the local resources; and (3) *interoperability* among parties. In addition, (4) the security framework must be *easy to use and deploy* into existing systems. A study of the literature has highlighted that none of the existing frameworks satisfies all four requirements.

In this paper we have introduced a security framework for SoS satisfying all the aforementioned requirements. Confidentiality and integrity of information are protected by complementing context-aware access control with trust management (TM); a distributed policy evaluation algorithm guarantees that policies confidentiality is not violated when a credential request is evaluated. Security policies are specified by means of a logic- and ontology-based specification language. The use of ontologies allows parties to provide a semantics to the terms employed in their policies and to describe domain and context information. This, combined with a semantic alignment technique, gives parties the autonomy to adopt the organizational model and vocabulary they consider more appropriate within their system, while preserving interoperability among the parties in the SoS.

The applicability of the security framework is demonstrated by a prototype implementation for a scenario in the maritime safety and security domain. The prototype is based on a number of standards and techniques (e.g., XACML and SAML), and consists of a set of components implemented following the service-oriented architecture paradigm. This facilitates the deployment of the framework into existing systems, and allows for an easy integration of additional components to support the evaluation of policies and provide additional functionalities. To enhance the usability of the framework we have developed a policy editor that assists security administrators in the specification and management of security policies, by providing features such as syntax highlighting, syntax check, and auto-completion of ontology concepts, relationships, and instances.

In our prototype we have not used encryption to protect the communication between parties, and have not included a signature verification module to verify the authenticity of credentials. Obviously, in a deployed system both features would be necessary, and their addition is part of our future work. Furthermore, we are developing an enforcement mechanism for sticky policies.[8] In the current prototype, parties can attach policies to the data they send to the other parties in the SoS, but have to trust the recipient to enforce these policies. The scheme that we are currently designing combines attribute-based encryption[2] with TM to enable cryptographic enforcement of sticky policies in dynamic SoS. In this way, parties can make their data available within the SoS with the guarantee that only the authorized parties are able to access them.

Acknowledgments

The authors thank the members of the Poseidon project for the useful discussions and valuable comments. 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 has also been funded by the EU-IST-IP-216287 TAS[3] project.

References

1. M. Y. Becker and P. Sewell. Cassandra: Distributed Access Control Policies with Tunable Expressiveness. In *Proceedings of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY'04*, pages 159–168, Washington, DC, USA, 2004. IEEE Computer Society.
2. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy, SP'07*, pages 321–334, Washington, DC, USA, 2007. IEEE Computer Society.
3. R. Bhatti, E. Bertino, and A. Ghafoor. A Trust-based Context-Aware Access Control Model for Web-Services. In *Proceedings of the IEEE International Conference on Web Services, ICWS '04*, pages 184–191, Washington, DC, USA, 2004. IEEE Computer Society.
4. R. Bhatti, E. Bertino, and A. Ghafoor. A Trust-Based Context-Aware Access Control Model for Web-Services. *Distributed and Parallel Databases*, 18(1):83–105, July 2005.

5. M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy, SP'96*, pages 164–173. IEEE Computer Society, 1996.
6. K. Böhm, S. Etalle, J. den Hartog, C. Hütter, S. Trabelsi, D. Trivellato, and N. Zannone. Flexible Architecture for Privacy-Aware Trust Management. *Journal of Theoretical and Applied Electronic Commerce Research*, 5:77–96, August 2010.
7. P. Bonatti and P. Samarati. Regulating service access and information release on the Web. In *Proceedings of the 7th ACM conference on Computer and Communications Security, CCS '00*, pages 134–143, New York, NY, USA, 2000. ACM.
8. D. W. Chadwick and S. F. Lievens. Enforcing "sticky" security policies throughout a distributed application. In *Proceedings of the 2008 workshop on Middleware security, MidSec'08*, pages 1–6, New York, NY, USA, 2008. ACM.
9. W. Chen and D. S. Warren. Tabled Evaluation With Delaying for General Logic Programs. *Journal of the ACM*, 43(1):20–74, January 1996.
10. I. Chlamtac, M. Conti, and J. J. n. Liu. Mobile ad hoc networking: Imperatives and challenges. *Ad Hoc Networks*, 1(1):13–64, 2003.
11. L. F. Cranor and S. Garfinkel, editors. *Security and Usability: Designing Secure Systems That People Can Use*. O'Reilly Media, 2005.
12. R. Culmone, G. Rossi, and E. Merelli. An Ontology Similarity Algorithm for Bioagent. In *Proceedings of the 2002 Workshop on NETwork Tools and Applications in Biology, NETTAB'02*, 2002.
13. M. Czenko, J. Doumen, and S. Etalle. Trust Management in P2P Systems Using Standard TuLiP. In *Proceedings of the 2nd IFIP WG 11.11 International Conference on Trust Management, IFIPTM'08*, volume 263/2008 of *IFIP*, pages 1–16, Boston, 2008. Springer.
14. A. Dersingh, R. Liscano, and A. Jost. 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.
15. H.-H. Do and E. Rahm. COMA: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international conference on Very Large Data Bases, VLDB'02*, pages 610–621. VLDB Endowment, 2002.
16. A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to match ontologies on the Semantic Web. *The VLDB Journal*, 12(4):303–319, November 2003.
17. S. Dürbeck, C. Fritsch, G. Pernul, and R. Schillinger. A Semantic Security Architecture for Web Services. In *Proceedings of the 5th International Conference on Availability, Reliability and Security, ARES'10*, pages 222–227. IEEE Computer Society, 2010.
18. S. Durbeck, R. Schillinger, and J. Kolter. Security Requirements for a Semantic Service-oriented Architecture. In *Proceedings of the The Second International Conference on Availability, Reliability and Security, ARES'07*, pages 366–373, Washington, DC, USA, 2007. IEEE Computer Society.
19. T. W. Finin, A. Joshi, L. Kagal, J. Niu, R. S. Sandhu, W. H. Winsborough, and B. M. Thuraisingham. R*OWL*BAC: representing role based access control in *OWL*. In *Proceedings of the 13th ACM symposium on Access control models and technologies, SACMAT'08*, pages 73–82, New York, NY, USA, 2008. ACM.
20. K. Frikken, M. Atallah, and J. Li. Attribute-Based Access Control with Hidden Policies and Hidden Credentials. *IEEE Transactions on Computers*, 55:1259–1270, October 2006.
21. S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A Survey of Gossiping and Broadcasting in Communication Networks. *Networks*, 18(4):319–349, 1988.

22. S. Heeps, J. Sventek, N. Dulay, A. S. Filho, E. Lupu, M. Sloman, and S. Strowes. Dynamic Ontology Mapping for Interacting Autonomous Systems. In *Proceedings of the 2nd International Workshop on Self-Organizing Systems, IWSOS'07*, volume 4725 of *LNCS*, pages 255–263. Springer, 2007.
23. I. Horrocks, P. F. Patel-Schneider, S. Bechhofer, and D. Tsarkov. OWL rules: A proposal and prototype implementation. *Journal of Web Semantics*, 3(1):23–40, 2005.
24. J. H. Janssens, H. Hiemstra, and E. O. Postma. Creating artificial vessel trajectories with Presto. In *Proceedings of the 22nd Benelux Conference on Artificial Intelligence, BNAIC'10*, 2010.
25. L. Kagal, M. Paolucci, N. Srinivasan, G. Denker, T. Finin, and K. Sycara. Authorization and Privacy for Semantic Web Services. *IEEE Intelligent Systems*, 19(4):50–56, 2004.
26. J. Kolter, R. Schillinger, and G. Pernul. Building a Distributed Semantic-aware Security Architecture. In *Proceedings of the IFIP TC-11 22nd International Information Security Conference, SEC'07*, volume 232 of *IFIP*, pages 397–408. Springer, 2007.
27. A. J. Lee, M. Winslett, and K. J. Perano. TrustBuilder2: A Reconfigurable Framework for Trust Negotiation. In *Proceedings of the 3rd IFIP WG 11.11 International Conference on Trust Management, IFIPTM'09*, volume 300 of *IFIP*. Springer, 2009.
28. N. Li and J. C. Mitchell. Datalog with Constraints: A Foundation for Trust Management Languages. In *Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages, PADL'03*, volume 2562 of *LNCS*, pages 58–73, London, UK, UK, 2003. Springer-Verlag.
29. N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a Role-Based Trust-Management Framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy, SP'02*, pages 114–130, Washington, DC, USA, 2002. IEEE Computer Society.
30. M. W. Maier. Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284, 1998.
31. I. R. Mbaya, A. J. Gerber, and A. J. van der Merwe. Requirements of a security framework for the semantic web. In *Proceedings of the The IASTED International Conference on Software Engineering, SE'09*, pages 106–113. ACTA Press, 2009.
32. W. Nejdl, D. Olmedilla, and M. Winslett. PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. In *Proceedings of the 2004 VLDB Workshop on Secure Data Management, SDM'04*, volume 3178 of *LNCS*, pages 118–132. Springer, 2004.
33. L. D. Ngan, T. M. Hang, and A. E. S. Goh. Semantic Similarity between Concepts from Different OWL Ontologies. In *Proceedings of the 5th IEEE International Conference on Industrial Informatics, INDIN'06*, pages 618–623, Los Alamitos, CA, USA, 2006. IEEE Computer Society.
34. H. A. Nguyen and H. Al-Mubaid. A Combination-based Semantic Similarity Measure using Multiple Information Sources. In *Proceedings of the 2006 IEEE International Conference on Information Reuse and Integration, IRI'06*, pages 617–621. IEEE Systems, Man, and Cybernetics Society, 2006.
35. OASIS. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Technical report, OASIS Standard, 2005.
36. OASIS. eXtensible Access Control Markup Language (XACML) Version 2.0. Technical report, OASIS Standard, 2005.
37. OASIS. SAML 2.0 profile of XACML v2.0. Technical report, OASIS Standard, 2005.
38. OASIS. Reference Model for Service Oriented Architecture 1.0. OASIS Standard, 2006.
39. R. Rosati. DL+log: Tight Integration of Description Logics and Disjunctive Datalog. In *Proceedings of the 10th International Conference on Principles of Knowledge*

Representation and Reasoning, KR'06, pages 68–78. AAAI Press, 2006.
40. K. E. Seamons, M. Winslett, and T. Yu. Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation. In *Proceedings of the Network and Distributed System Security Symposium, NDSS'01*. The Internet Society, 2001.
41. K. Stine, R. Kissel, W. C. Barker, A. Lee, and J. Fahlsing. Guide for Mapping Types of Information and Information Systems to Security Categories. Special Publication SP 800-60 Rev. 1, National Institute of Standards and Technology (NIST), 2008.
42. D. Toman. Memoing Evaluation for Constraint Extensions of Datalog. *Constraints*, 2(3/4):337–359, January 1998.
43. A. Toninelli, R. Montanari, L. Kagal, and O. Lassila. A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. In *Proceedings of the 5th International Semantic Web Conference, ISWC'06*, volume 4273 of *LNCS*, pages 473–486. Springer, 2006.
44. D. Trivellato, F. Spiessens, N. Zannone, and S. Etalle. Reputation-Based Ontology Alignment for Autonomy and Interoperability in Distributed Access Control. In *Proceedings of the 2009 International Conference on Computational Science and Engineering, CSE'09*, volume 3, pages 252–258, Washington, DC, USA, 2009. IEEE Computer Society.
45. D. Trivellato, N. Zannone, and S. Etalle. GEM: a Distributed Goal Evaluation Algorithm for Trust Management. Technical Report CS 10-15, Eindhoven University of Technology, 2010.
46. D. Trivellato, N. Zannone, and S. Etalle. A Security Framework for Systems of Systems. In *Proceedings of the 12th IEEE international conference on Policies for distributed systems and networks, POLICY'11*, Piscataway, NJ, USA, 2011. IEEE Computer Society.
47. D. Trivellato, N. Zannone, and S. Etalle. POSTER: Protecting Information in Systems of Systems. In *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS'11*. ACM, 2011.
48. A. Uszok, J. M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, and S. Aitken. KAoS Policy Management for Semantic Web Services. *IEEE Intelligent Systems*, 19(4):32–41, 2004.
49. B. Whitworth. The Social Requirements of Technical Systems. In *Handbook of Research on Socio-Technical Design and Social Networking Systems*, pages 3–22. IGI Global, 2009.
50. N. Willems, W. R. van Hage, G. de Vries, J. Janssens, and V. Malaisé. An integrated approach for visual analysis of a multisource moving objects knowledge base. *International Journal of Geographical Information Science*, 24(10):1543–1558, 2010.
51. M. Winslett, C. C. Zhang, and P. A. Bonatti. PeerAccess: a logic for distributed authorization. In *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS'05*, pages 168–179. ACM, 2005.
52. World Wide Web Consortium. OWL Web Ontology Language. Technical report, W3C Recommendation, 2004.
53. T. Yu and M. Winslett. A Unified Scheme for Resource Protection in Automated Trust Negotiation. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy, SP'03*, pages 110–122, Washington, DC, USA, 2003. IEEE Computer Society.