2008 International Conference on Business Information Systems

Towards the Aggregation of Security Requirements in Cross-Organisational Service Compositions

Michael Menzel[1], Christian Wolter[2], and Christoph Meinel[1]

[1] Hasso-Plattner-Institute, University of Potsdam, Germany
{michael.menzel,meinel}@hpi.uni-potsdam.de
[2] SAP Research, CEC Karlsruhe, Germany
{christian.wolter}@sap.com

Abstract. The seamless composition of independent services is one of the success factors of Service-oriented Architectures (SOA). Services are orchestrated to service compositions across organisational boundaries to enable a faster reaction to changing business needs. Each orchestrated service might demand the provision of specific user information and requires particular security mechanisms. To enable a dynamic selection of services provided by foreign organisations, a central management of static security policies is not appropriate. Instead, each service should express its own security requirements as policies that stipulate explicitly the requirements of the composition. In this paper we address the problem of aggregating security requirements from orchestrated services. Such an aggregation is not just the combination of all security requirements, since dependencies and conflicts between these requirements might exist. We provide a classification of these dependencies and introduce a conceptional security model enabling a classification of security requirements to reveal conflicts. Finally, we propose an approach to determine an aggregation of security requirements in cross organisational service compositions.

Key words: SOA, Service Composition, Security, Security Policy

1 Introduction

Service-oriented Architectures (SOA) facilitate the interoperable and seamless interaction of service consumer and service provider to meet the consumer's needs by the service's capabilities. The standard set of Web Service technologies, such as WSDL, UDDI, and SOAP provides the means to describe, locate, and invoke a Web Service based on XML. The independent nature of the services, with respect to operating systems and system architectures, facilitate a composition of different services. In fact, service composition is one of the success factors of Service-oriented Architectures to enable the flexible integration of services provided by independent business partners.

However, the seamless and straightforward integration of cross-organisational services conflicts with the need to secure and control the access to provided ser-

vices. Each service may support different security mechanisms and may require different pieces of user information for access control. Since the user of the composed service might be unknown to the provider of the orchestrated services, the establishment of an identity federation is a basic necessity to provide required information across domain borders. In an identity federation all parties are willing to rely on assertions representing claims about users. These claims are issued by a trusted identity provider that manages a digital identity of the service user. For instance, a credit card company can assert the user's name and his credit card information in an encrypted security token (e.g. SAML [12]). A user can request this token on demand of a service's requirements and include this token in a SOAP message to invoke a service. Windows CardSpace [3] is one example for a client technology to manage digital identities from various identity providers. Based on the requirements of a service, CardSpace acts as an identity selector and enables a user to choose from a set of identity providers that can assert the required set of claims. Since this technology is founded on the WS-* protocols, WS-Policy [5] is used to express the requirements.

Security policies expressing requirements for the users of a service are typically generated when the service is deployed, and are assigned to the service statically. This approach is useful in a single domain leveraging a central policy management, but it is not feasible in the context of service compositions containing services from independent organisations. The security requirements of the composed service do not only depend on local security requirements, but also on security policies and service level agreements specified by the orchestrated services. In consequence, the security policy of the composed service might have to be adapted when a service from another service provider is mapped to the composition. This is especially an issue in service compositions that dynamically select services based on non-functional properties. Moreover, it must be considered that there might be dependencies between different security requirements. Requirements might interact, contradict or conflict, so that a simple combination of all requirements is not sufficient.

Current research approaches [2, 1] are based on a semantic matching of service's security preconditions to create a service composition that can be executed with the user's security capabilities. These approaches are focused on enabling an automatic composition under the restriction that the user must state in advance which security credentials and mechanisms he is able to support. This solution disregards privacy concerns, since it conflicts with the conception that the user should control the usage of his identity information by selecting an appropriate digital identity. Moreover, former approaches did not consider dependencies between security requirements that may result in an insecure policy of the service provider.

In this paper we propose a method to aggregate and verify security requirements for cross-organisational federated service compositions. Therefore, we provide:

- a classification of dependencies and effects between requirements (*requirement interactions*) that must be resolved when aggregating a consistent set of security requirements.
- a conceptual security model that describes entities (e.g. security goals, policies and security mechanisms) and their relationships regarding security requirements independently from technical aspects.
- an approach to determine a consistent aggregation of security requirements for a service composition

The rest of this paper is organised as follows. In Section 2 we present a travel agency scenario to illustrate dependencies of requirements in a service composition and a definition of interaction classes between security requirements. In Section 3 we introduce our conceptional security model to distinguish different types of security requirements. Finally, we describe in Section 4 how our model can be used to determine the interaction classes of security requirements and how security requirements can be aggregated in a service composition. Section 5 provides an overview about related work, while the final section concludes this paper.

2 Interactions of Security Requirements

In this section we present an example to clarify the aggregation of security requirements in a service composition. Moreover, we provide a complete list of possible interactions between these requirements that have to be considered to compute a secure aggregation.

2.1 Travel Agency Service Scenario

Consider the example of a composed travel agency service that is capable to perform a hotel and a flight reservation as shown in Figure 1. We assume that the request to this service has to include all needed functional parameters, e.g. preferred hotel, room types, arrival and departure time, and preferred type of flight. Based on this information, the service is able to book the preferred combination of hotel and flights. The service composition contains four independent services whereas the services a_2, a_3, and a_4 are provided by business partners of the service a_1. The first service a_1 validates and verifies the functional service parameters and invokes the service a_2, which performs the reservation of the desired hotel. Finally, depending on the preferred type of flight, either the service a_3 is called to book a budget flight or the service a_4 is called to book a regular flight.

In addition to these functional parameters, identity related information must be provided to complete the booking process successfully. The services a_2, a_3, and a_4 require the name and address of the user as well as his credit card information for payment. To secure the exchanged information, different security mechanisms are needed to ensure confidentiality and integrity. While the services

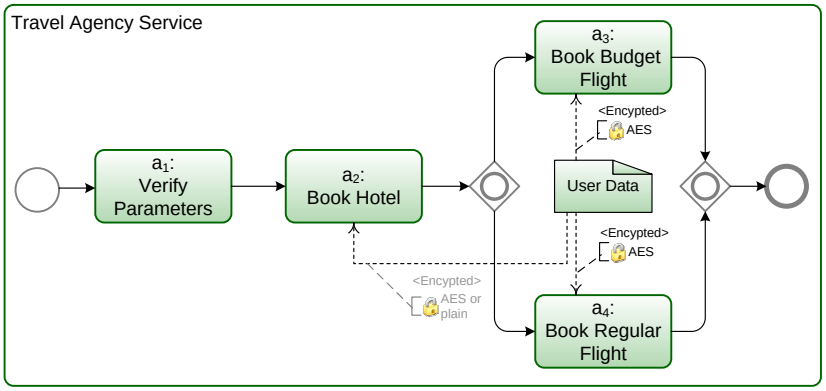

Fig. 1. Travel Agency Service Composition

a_3 and a_4 require the user to decrypt the information using the AES algorithm, it is optional for service a_2. WS-Policy can be used to express the requirements of the individual services.

Although the policies of the basic services are well defined, it is challenging to generate a secure policy for the service composition. The collection and simple combination of all security assertions is not sufficient, since independent requirements might interact. In our example, the simple combination of the basic policies would allow that two security tokens conveying the same pieces of identity information are added to the service request with different requirements for encryption. One security token with the users name and address would be encrypted with AES, while the other token with the same information can be added to the same message without encryption. It is obvious that this is a security breach, but what if service a_2 supports a weaker encryption mechanism (e.g. DES) or AES with a shorter key length? The security level regarding the identity information would depend on the weakest security mechanism. Another possibility is that different services require different mechanisms to guarantee integrity. How can be ensured that they provide the same level of security?

2.2 Classification of Requirement Interactions

A secure service complies with a set of security goals $G_i = \{g_1, \ldots, g_m\}$ (e.g. authorisation or confidentiality) requiring a certain behaviour or information from a client that wants to access the service. These expectations are expressed as a set of requirements $R_i = \{r_{i1}, \ldots, r_{im_i}\}$ associated with this service. To guarantee a consistent and valid set of requirements, interactions (e.g. conflicts) between these requirements must be considered. A *requirement interaction* is the effect that two requirements have on each other. We introduce a classification of security requirement interactions in this section based on former work on QoS interactions described by Wohlstadter et el. [13]. A requirement has a positive or negative impact that increases or decreases the service's security. This impact is

expressed by the function Ψ. Based on the comparison of the individual security impact of two requirements r_1 and r_2 with the combined one, eight classes of interaction can be revealed:

- *Independent* - $\Psi(\{r_1, r_2\}) = \Psi(\{r_1\}) + \Psi(\{r_2\})$
  The requirements are totally independent and do not interfere with each other. The combined security impact is the sum of the individual ones.
- *Equivalent* - $\Psi(\{r_1, r_2\}) = \Psi(\{r_1\}) = \Psi(\{r_2\})$
  Two requirements will be equivalent, if they have the same effect providing the same level of security. It is optional to use both requirements together, although it might be advantageous to gain a greater flexibility by offering equivalent alternatives to the service consumer.
- *Prevent* - $\Psi(\{r_1, r_2\}) = \Psi(\{r_1\})$
  The requirement r_1 will prevent r_2, if r_2 has no impact on the security. Consider the aforementioned travel agency scenario describing a service invocation that contain the same pieces of information in an encrypted and a plain style. The unsecure requirement prevents the high security requirement to effect the communication.
- *Restrict* - $\Psi(\{r_1, r_2\}) < \Psi(\{r_1\}) + \Psi(\{r_2\})$
  Both requirements lower the impact of the other one. The combined impact is usually greater than the individual ones, but less than the sum of those.
- *Complements* - $\Psi(\{r_1, r_2\}) > \Psi(\{r_1\}) + \Psi(\{r_2\})$
  The requirements will be complementary, if the combined security impact is larger than the individual security level. It is more secure to use both requirements together.
- *Require* - $\Psi(\{r_1, r_2\}) > 0, \Psi(\{r_1\}) \leq 0)$
  A requirement r_1 will require r_2, if the combined security impact is positive, while the individual impact of r_1 is negative. Both requirements must be used together.
- *Conflict* - $\Psi(\{r_1, r_2\}) < 0, \Psi(\{r_1\}) \geq 0 \wedge \Psi(\{r_2\}) \geq 0$
  Two requirements will conflict, if their combined impact is negative, although their individual impacts are positive. These requirements must not be deployed together.
- *Exclude* - $\Psi(\{r_1, r_2\}) = \Psi(\{r_1\}) \vee \Psi(\{r_1, r_2\}) = \Psi(\{r_2\})$
  The requirements r_1 and r_2 represent an excluding alternative that may be selected in dependency on functional or non-functional parameters of a service.

Although the possibility of interacting security requirements for a single service has to be considered, this problem is much more serious in terms of service compositions. In a service composition there are multiple sets of security requirements $R_i = \{r_{i1}, \ldots, r_{im_i}\}$, $m_i \in \mathbb{N}$ involved that are associated with each service i. Each service can have its own understanding of security and states its own requirements regarding the same security goals. Therefore, if R_{comp} is the set of consistent security requirements for the service composition satisfying a set of predefined security goals, then $R_{comp} \subseteq \bigcup_{i=1\ldots n} R_i$. Due to possible interactions between security requirements, it is likely that R_{comp} is unequal to the union of all requirements.

3 Security Model

Security requirements are expressed by security policies, usually in a very technical and policy language dependent way. To determine the interaction type of two requirements, we need a security model that abstracts from technical details. The model must reveal all security aspects in an SOA landscape and the relationship among affected entities. Therefore, our conceptual security model describes basic security goals and outlines the relationship to specific security attributes and mechanisms.

3.1 Specifying Security Goals

The abstract concept of security can be defined precisely by specifying a set of security goals [11]. Although these goals can be further specialised, subdivided or combined, we will focus solely on basic goals in this paper:

1. *Confidentiality* provides protection against the unauthorised notice of stored, processed, or transferred information.
2. *Integrity* ensures the properness (intactness, correctness, and completeness) of information (data integrity). Transferred, processed, or stored data must not be modified with proper rights and - in economic terms - modifications must correspond to business values and expectations.
3. *Authentication* ensures the credibility of information - such as a claimed identity - by confirming this information as authentic.
4. *Authorisation* is the process of granting rights to participants to perform an interaction, for instance to access a resource.
5. *Traceability and Auditing* provide verifiability regarding all performed actions in an information processing system. This can be related to simple logging mechanisms, but also to monitoring as real-time auditing e.g. in intrusion detection systems.
6. *Availability* ensures that data, resources and services, which are needed for the proper functioning of a system, are available at each point in time regarding the requested quality of service.

These goals can be related to various entities in a Service-oriented Architecture. The relations among security goals and affected entities are typically described by *Constraints* that are composed in a security *Policy* as indicated by Figure 2. A subset of constraints can be exposed as requirements to service user. Since some constraints regulate the internal functioning of the service, they are not revealed to the service user.

The basic entity in such a model is an *Object*. We define an object as an entity that is capable of participating in an *Interaction* with other objects. This interaction will always lead to an *Effect*, which can comprise the provision of information or the change of state in a system. The effect can, but does not need to be, related to the object that initiated the interaction. For example, one object could be an application and another object could be a service to store

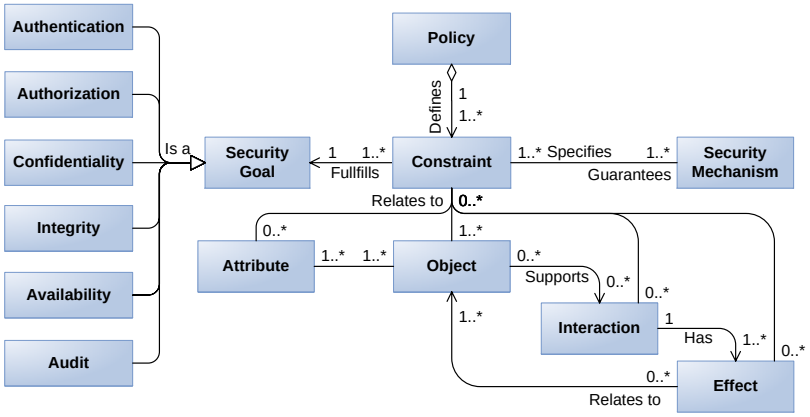

Fig. 2. Security Policy Model

data. The process of accessing this service would be the interaction resulting in the effect that data is stored or some information is returned to the application. This conception is close to the model that is described in the OASIS reference model for SOA [10] and enables a straight mapping to this model.

Each object is related to a set of attributes describing its meta information. For instance, if the object represents a user, attributes, such as name, email address, age, etc. will be assigned. Altogether, policy constraints always refer to a set of objects, a particular set of objects' attributes, and optionally a set of interactions and effects that are related to the objects. Based on these relations, specific constraints for particular security goals can be defined. These specific constraints define requirements for associations between the entities with regard to the particular security goals.

As shown in Figure 2, constraints specify security mechanisms that enforce or guarantee the defined constraint. For instance, a confidentiality policy usually specifies an algorithm (e.g. DES) that must be used to guarantee this requirement.

3.2 Enforcing Security Constraints

In our model a *Security Mechanism* is designed to characterise techniques that are used to enforce security constraints (cf. Figure 3). It provides the foundation to specify a comprehensive ontology for security mechanisms, see [4].

Besides security mechanisms, a *Credential* represents another important entity in our model that subsumes evidences used by security mechanisms. A detailed classification of security credentials was presented by Denker *et al.* [4]. In this work they introduced an ontology that divides credentials in simple credentials (e.g. key, login, certificate) and composed credentials (e.g. Smart Card, SAML, WS-Security Token) that contain a set of simple credentials.

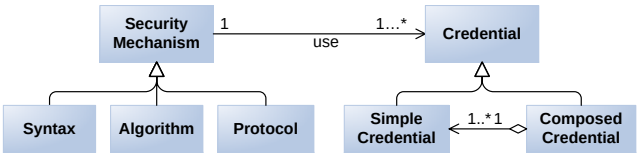

Fig. 3. Security Mechanisms Model

Based on the given security policy model (cf. Figure 2), we defined semantics for specific types of *Constraints*, each guaranteeing one of the security goals listed above. Each constraint is related to a specific set of entities and define rules restricting particular associations between those entities. These rules must be enforced by security mechanisms and credentials. We will not elaborate upon the constraint models during the course of this paper, since these details have only a minor impact on the relationship among security requirements.

4 Aggregation of Security Requirements

In the previous section we introduced a security model that provides semantics for all security aspects related to a service. It has revealed that policy constraints (describing security requirements) are based on two basic ontologies: *mechanisms* to guarantee a certain security goals and *credentials* to provide required information. In this section we will present how this model can be used to determine the interaction class of security requirements. Finally, we will describe an approach to aggregate security requirements concerning a service composition.

4.1 Determining a consistent Aggregation of Security Requirements

Determining the interaction class of multiple requirements demands an evaluation of these requirements regarding the different aspects in our model and their relationships. First of all, it must be considered, if two requirements belong to the same class of security goals. Second, it is important, whether they refer to the same set of entities in terms of objects, attributes, and object interactions.

A requirement interaction will belong to the class

- *Independent*, if the requirements refer to different entities.
- *Prevent, Restrict, or Equivalent*, if the requirements refer to the same security goals and relate to the same entities. It is necessary to compare the security mechanisms and security credentials specified in the requirements to determine the interaction type precisely. A comparison of security mechanisms can be realised similar to the concept of trust indicators, which was proposed by Haller *et al.*[6]. Trust indicators are based on specific metrics to measure and compare security properties concerning service level agreements.
- *Require, Complements, or Conflict*, if the requirements refer to different security goals and relate to the same entities. In general, these requirement

interactions are inherited from dependencies between the associated security goals. The provision of authentication and authorisation information might require confidentiality and integrity to secure this information. Moreover, there might be contracting security goals that lead to conflicting requirements (e.g. monitoring and confidentiality).
– *Exclude*, if there are conditions defined in policy constraints that limit the application to one requirement exclusively, regardless of the related security goals or referred entities.

Security constraints defined in existing standards (e.g. WS-Policy) can be mapped to our model to ensure the policy's consistency. Restricting or preventing requirements can be eliminated if they are optional considering the revealed dependencies. In general, conflicting requirements demand a conflict resolution strategy to decide which requirements should be preferred. For instance, confidentiality requirements could take preference over monitoring requirements.

4.2 Aggregating Security Requirements in a Service Composition

To enable an aggregation of security requirements regarding a service composition, we have to consider that

– each service has its own security requirements that might cause complex dependencies between the requirements of different services.
– the execution of services results in an effect that can satisfy the requirement of a successive service. This requirement does not have to be satisfied by the user.
– the service composition might include xor-splits and -joins depending on conditions based on functional parameters. This might result in different sets of requirements that exclude each other.

Results from research work [9] about the generation of semantics of service compositions provide a suitable foundation for the determination of the composition's security requirements. These approaches are based on the aggregation of semantic preconditions of basic services that correspond to our requirements. Meyer [9] introduces a formal workflow model based on petri nets and describes how to express the semantics of services inside this petri net. To calculate the functionality of the aggregation, Meyer introduced graph algorithms that work by recursively defining the state for the markings in the workflow net. Using our security model, and comprising ontologies for *security mechanisms* and *credentials*, we are able to translate policies into semantic preconditions for services. These preconditions can be used with the description of the service composition as input to the aforementioned algorithms to generate the requirements of the service composition expressed as Boolean formula. This output can be checked for consistency as described in the previous section and be mapped to a policy again.

5 Related Work

Our conceptional data model for service based systems is mainly driven by the semantic security annotation approach for web services proposed by Denker *et al.* in [4]. These annotations are used to describe the security capabilities of web services. A reasoning engine is used to perform a security matchmaking between service providers and requesting agents. While providing suitable security ontologies for security mechanisms and credentials, they do not consider the relationship to policy constraints and possible interactions.

Only a few approaches have been published so far that intent to enable the dynamic composition of services regarding security constraints. Carminati *et al.* described an approach to compose web services based on security requirements of web service consumers and web service providers in [2, 1]. Semantic matchmaking is used to create a service composition under security constraints. This approach assumes that service consumers are willing to specify their requirements and capabilities in advance. Dependencies between security requirements are not considered in this work.

Cheikh *et al.* proposed technique for automatic web service compositions in trust-aware communities based on reduction to satisfiability in propositional dynamic logic. Their work is focused on access control and authorisation constraints and do not consider other security goals.

The abstract description of security properties has also been addressed in previous work. However, there is no comprehensive approach that relates all security aspects in a SOA to security goals. Huang presents in [14] a framework for semantic descriptions for web service security constraints. His approach intents to enable a reasoning over non-functional properties and the integration of business rules. Although a general framework to perform a reasoning is described, the paper does not provide a concrete security ontology to describe security constraints and their relationship to security goals. Moreover, an aggregation of security constraints is not considered.

Jürjens presented in [7] the UMLSec extension for UML to express security relevant information within a system specification diagram. The focus of UMLSec lies primarily on the modelling of communication-based security goals, such as confidentiality, for software artefacts rather than a more general approach to modelling a variety of security goals and their relationship.

6 Conclusion

In this paper we presented an approach to aggregate security requirements of services that are composed across organisational boundaries. Each orchestrated service provides its own security policies that stipulate the security requirements of the composed service. Our vision is to facilitate the dynamic composition of services by enabling the calculation of a consistent requirement aggregation that can be communicated to the service user as a security policy. The provision of

such an aggregated security policy enables the service user to securely select an appropriate digital identity representing the required information.

We showed that interactions may occur between security requirements exposed by services from different trust domains. We provided a definition and a complete classification of requirement interactions based on an abstract model. To distinguish these requirements, we introduced a conceptual security model to add semantics to different types of security constraints. Our model revealed the structure of these constraints that describe requirements, their relationship to high level security goals and their dependency to security mechanisms and credentials as basic ontologies. Finally, we described how our model can be used to determine and resolve the interaction classes and introduced an approach to aggregate security requirements based on the computation of preconditions in semantic workflows.

In contrast to current research approaches that are focused on the dynamic composition of services with the intention to meet predefined capabilities of a service user, we introduced a solution that does not require the user to expose all security capabilities in advance. Moreover, the interaction of requirements is not considered by former semantic approaches.

6.1 Future Work

We stated that our security model is an approach to describe requirements and their interactions. There is an ongoing effort to map our security model to WS-Policy and it must be proved that our model can be used to determine the interaction classes of security requirements. Moreover, we presented a basic approach to resolve requirement conflicts in this paper. More complex strategies can be integrated as it has been presented in research work concerning the resolution of policy constraint conflicts [8]. Finally, we have to prove that the petri net models used to calculate the semantic precondition of workflows can be extended with our model to determine the aggregation of security requirements. These topics will be addressed by future work.

References

1. B. Carminati, E. Ferrari, and P.C. K. Hung. Web Service Composition: A Security Perspective. WIRI 2005: 248–253, 2005.
2. Barbara Carminati, Elena Ferrari, and Patrick C. K. Hung. Security Conscious Web Service Composition. ICWS 2006: 489–496, 2006.
3. David Chappel. Understanding Windows CardSpace, April 2006.
4. Grit Denker, Lalana Kagal, Timothy W. Finin, Massimo Paolucci, and Katia P. Sycara. Security for DAML Web Services: Annotation and Matchmaking. ICWS 2003: 335–350, 2003.
5. Giovanni Della-Libera, Martin Gudgin, and et all. Web Services Security Policy Language (WS-SecurityPolicy). Public Draft Specification, Juli 2005.
6. Jochen Haller and Christian Wolter. Trust Indicator Integration into SLAs for Virtual Organisations. eChallenges 2007 Conference, 2007.

7. Jan Jürjens. UMLsec: Extending UML for Secure Systems Development. UML 2002: 412–425, 2002.
8. Trent Jaeger, Reiner Sailer, and Xiaolan Zhang. Resolving constraint conflicts. SACMAT 2004: 105–114, New York, NY, USA, 2004. ACM.
9. Harald Meyer. On the Semantics of Service Compositions. Proceedings of The First International Conference on Web Reasoning and Rule Systems (RR 2007), 2007.
10. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter Brown, and Rebekah Metz. Reference Model for Service Oriented Architecture 1.0. OASIS Committee Specification, February 2006.
11. Charles P. Pfleeger and Shari Lawrence Pfleeger. Security in Computing. Prentice Hall Professional Technical Reference, 2002.
12. Nick Ragouzis, John Hughes, Rob Philpott, and Eve Maler. Security Assertion Markup Language (SAML) V2.0 Technical Overview, 2006.
13. Eric Wohlstadter, Stefan Tai, Thomas Mikalsen, Isabelle Rouvellou, and Premkumar Devanbu. GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions. ICSE 2004:189199, 2004.
14. Dong Huang, Semantic Descriptions of Web Services Security Constraints, SOSE 2006: 81–84, IEEE Computer Society, Los Alamitos, CA, USA, 2006