CONTEXT-DRIVEN POLICY ENFORCEMENT AND RECONCILIATION FOR WEB SERVICES

S. Sattanathan[1], N. C. Narendra[2], Z. Maamar[3] and G. Kouadri Mostéfaoui[4]
[1]*National Institute of Technology Karnataka, Surathkal, India*
[2]*IBM India Research Lab, Bangalore, India*
[3]*Zayed University, Dubai, UAE*
[4]*University of Montreal, Montreal, Canada*

Keywords: Policy, Context, Security, Web service.

Abstract: Security of Web services is a major factor to their successful integration into critical IT applications. An extensive research in this direction concentrates on low level aspects of security such as message secrecy, data integrity, and authentication. Thus, proposed solutions are mainly built upon the assumption that security mechanisms are static and predefined. However, the dynamic nature of the Internet and the continuously changing environments where Web services operate require innovative and adaptive security solutions. This paper presents our solution for securing Web services based on adaptive policies, where adaptability is satisfied using the contextual information of the Web services. The proposed solution includes a negotiation and reconciliation protocol for security policies.

1 INTRODUCTION

1.1 Motivation

Web services are emerging as a major technology for deploying automated interactions between distributed and heterogeneous systems. With services relying on the insecure Internet for mission-critical transactions, security turns out to be a major concern to the adoption of Web services by the IT community and the reliability of these transactions. In a previous work (Maamar et al., 2004), we argued that because services require computing resources on which they operate, it was deemed appropriate guaranteeing that neither the Web services misuse the resources (e.g., blocking a resource for longer time periods) nor the resources affect the integrity of Web services (e.g., altering sensitive data of a Web service). Currently, a battery of security techniques permits protecting Web services at the levels of authentication, message safety, and data integrity (Lilly, 2004). However, these techniques are statically determined at design-time and cannot be adjusted during the life-cycle of Web services without going through an error-prone programming exercise.

In order to develop adaptive security strategies for Web services that are inline with the dynamic nature of the Internet, we propose, in this paper, a dynamic approach that combines first, a context management driven by the requirements of security and second, policies dedicated to achieving this security. While the security context, as presented in this proposal, takes advantage of our prior work on context ontology for Web services (Maamar et al., 2005), the value-added of policies to Web services security is detailed throughout this paper. The expected use of policies is to take actions according to the occurring threats and detected attacks that put in risk the security of Web services. Policies are to be loaded and triggered with respect to the current *context*, which features the environment that surrounds a Web service (e.g., user location, day time, attack type). Some of the elements that are tracked using a security context are multiple including the identification of the security violations, the alteration situations that affected the integrity of Web services, and the corrective actions to address attempts of resource misuse. In (Kouadri Mostefaoui and Brezillon, 2004), the authors employed context-based security to adapt the security strategy depending on a set of relevant information collected from the dynamic environment. By promoting a security context we aim at tracking all the concerns and threats that affect contexts of Web services and at deploying appropriate measures based on previous security contexts. Figure 1 illustrates how the connection between security policies, service contexts, and security contexts is deployed. The

Sattanathan S., C. Narendra N., Maamar Z. and Kouadri Mostéfaoui G. (2006).
CONTEXT-DRIVEN POLICY ENFORCEMENT AND RECONCILIATION FOR WEB SERVICES.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - SAIC*, pages 93-99
DOI: 10.5220/0002441000930099

configuration of a security policy is progressively tuned using the information that security contexts cater. This information is obtained after assessing the integrity of the content of service contexts. The content of a service context has been detailed in (Maamar et al., 2005), and permits for instance to track the participation of a Web service in the execution of a composite service. In case the content of a service context was subject to attacks, it would be deemed appropriate reporting these attacks at the level of the security context, so the concerned security policies are reviewed. In this paper, we adopt Ponder as a language for specifying security policies (Damianou et al., 2001). We selected Ponder for policy specification in compliance with some requirements that need to be satisfied like expressiveness, simplicity, and scalability.

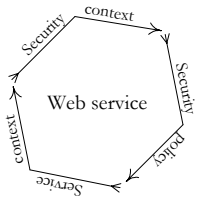


Figure 1: Collaboration of contexts and policies.

Each service, Web or composite, dynamically determines its security mechanisms based on the guidelines that it receives from its respective security context. Initially, a set of security policies are by default set according to an initial state of the content of a service context. This is done by domain applications' administrators after an initial threat assessment. When a change in a service context takes place, required changes in the security context are reported and policies are triggered if needed.

Because of the heterogeneity of the Internet, it is unlikely that a certain provider would deliver all types of context information and all types of security policies. Therefore, contexts and policies have a content of different granularities and structures as well. To manage this heterogeneity, there is a need for a common security policy on which providers agree. Policy reconciliation is achieved using a policy negotiation protocol to be shown in this paper. For context heterogeneity, readers are referred to our previous work (Maamar et al., 2005).

1.2 Contributions

Our contributions revolve around two major aspects: security context for tracking threats, breaches, and alterations (Maamar et al., 2005), and context-driven security policy for addressing the elements of the first aspect (this paper's focus). In the rest of this paper we present some background information in Section 2. Our approach towards securing Web services is discussed in Section 3. In Section 4, we illustrate this approach via an example. The paper concludes and discusses our future work in Section 5.

2 BACKGROUND

2.1 Rationale of Adopting Policies

Two reasons motivate our adoption of policies for securing Web services and their intrinsic context content. First, policies permit managing Web services at a higher level where guidelines for conducting composition of Web services are separated from guidelines for securing contexts of Web services. On the one hand, composition guidelines concern among others how to look for the relevant Web services, how to integrate user preferences into Web services, how to track the execution progress of Web services, and last but not least how to assess the context of Web services. On the other hand, guidelines for the security of Web services concern among others how to secure the incoming and outgoing messages received and sent out by Web services, how to authenticate requests of users, and how to suspend executions in case of risks of behavior alteration, information interception, or constraint violation.

The second reason for adopting policies is the opportunity of changing policies without affecting specifications of compositions. This separation of concerns is important; it permits working on security issues at composition and component levels. By changing policies, Web services can be continuously adjusted to accommodate variations in environments.

2.2 Related Work

Context-based security as a general paradigm is discussed in (Kouadri Mostefaoui, 2004) where the author gives a formal definition of both context-based security and security context. The definition is "*a state of the working environment that requires taking one or more security actions. A security context is formed by a set of information collected from the user's environment and the application environment and that is relevant to the security infrastructure of both the user and the application.*" The mentioned contribution, also, presents a worth discussion of the need for adaptive -context-based-

policies for emerging applications such as Web services.

In (Bhatti et al., 2004), Bhatti et al. argue that traditional identity and capability based schemes for access control do not scale well with Web services architectures and therefore, propose the incorporation of trust and context in access control to Web services. As a first step, Bhatti et al. offer an XML-based Generalized Temporal Role-Based Access Control (X-GTRBAC), which allows policy enforcement in heterogeneous, distributed environments. As a second step, they make use of trust management credentials in order to allow distributed authentication.

In (Hu and Weaver, 2004), Hu and Weaver describe a dynamic authorization enforcement scheme for Web services-based healthcare systems. This scheme makes authorization decisions based on runtime parameters rather than simply the user role. In (Agarwal and Sprick, 2004), Agarwal and Sprick present the requirements for an access control system for semantic Web services, and present an algebra for composing access control policies for composite Web services from those of their components. This idea is further extended in (Agarwal et al., 2004) with the specification of a distributed credential-based access control system, which unifies concepts from SPKI/SDSI and DAML-S (forerunner of OWL-S).

In (Damiani et al., 2004), Damiani et al. show how access control policies for semantic Web services can be expressed using XACML (eXtensible Access Control Markup Language) thereby, creating a semantics-aware access control language. More fine-grained access control is investigated in (Damiani et al., 2001), where the authors present a technique to enforce fine-grained access control restrictions on individual XML elements that make up a SOAP call made by one Web service to another.

Our proposal for developing context-driven security policies differs from the research projects mentioned above. First, we handle the aspects related to security in a dynamic way by updating the security policies on a regular basis and based on the information that security contexts cater over the content of service contexts. Second, we provide a negotiation protocol to achieve the reconciliation between conflicting security policies. Interesting point to note, that this protocol can be used for dynamically modifying the security mechanisms without extensive reprogramming. This modification is achieved using service contexts and security contexts, which are separately specified via context ontologies. Separate specification of these contexts allows for their modification without affecting the base functionality of Web services execution.

To illustrate via a simple example, if a travel agent service has to make payments to a payment service, both services need to authenticate themselves to each other. Plus, the messages passed by each service need to be encrypted to the satisfaction of both parties. Similarly, mutually acceptable hashing algorithms need to be implemented in order to ensure the integrity of the messages sent. These security techniques are dynamically set/updated based on the service and security contexts, without disturbing the core functionality of each service.

3 SECURITY OF WEB SERVICES

3.1 Support Architecture

In (Maamar et al., 2005), the context model associated with Web services composition spreads over three levels (composite service, Web service, and Web service instance, details on Web services instantiation are given in (Maamar et al., 2004). Since composition involves many Web services (in fact instances), interactions between the three levels happens according to the following patterns: between service instances of the same composite service; from service instances to Web services and *vice-versa*; from service instances to composite services and *vice-versa*; and from Web services to composite services and *vice-versa*. To secure these interactions, we decompose the threats into three types (Maamar et al., 2005): *identity* threats where an attacker impersonates a legitimate Web service or user; *content-borne* threats where an attacker attacks Web services directly; and *operational* threats that render Web services unusable.

Figure 2 illustrates our model for securing the interactions between contexts of Web services. The model presents three security contexts: *ISec-context* for Web service instance, *WSec-context* for Web service, and *CSec-context* for composite service. These security contexts are defined along with the regular service contexts as reported in (Sattanathan et al., 2005) (i.e., I/W/C-context). A security context highlighting the security strategy that a service adopts. Any change in this strategy is automatically reflected on the security context so other peers can be aware of the change in case of compliance reasons.

To keep the paper self-contained, we only list the arguments of ISec-context, WSec-context, and CSec-context. Readers are referred to (Maamar et

al., 2005) for more details. The arguments of ISec-context are Label, Signature, Security mechanism, security status, security violation, corrective actions, and date. WSec-context is designed to take care of the security of a Web Service. Some arguments featuring WSec-context are as follows: label, signature, security mechanism, security status, security violation, corrective actions, security status/service instances, and date. Finally, CSec-context is designed to take care of the security of a composite service. Some arguments featuring CSec-context are as follows: label, signature, security mechanism, corrective actions, security per previous Web service instances, security of current Web service instances, security per next Web service instances, and date.

3.2 Context-driven Policies

Figure 2 shows the way service context and security context are gathered. In (Sattanathan et al., 2005), we reported that an unawareness or poor consideration of the security challenges during Web services composition and execution result in a lack of the quality and relevance of information that permits tracking the composition, monitoring the execution, and handling exceptions. The side-effects of this unawareness or poor consideration are multiple like adopting a wrong strategy for selecting a component Web service (e.g., favoring execution-cost over reliability criterion instead of the opposite), delaying the triggering of some urgent component Web services, or wrongly assessing the exact execution status of a Web service (e.g., being suspended instead of being thought as under execution). Therefore, the security of Web services and their associated contexts need to be dealt with according to the environment wherein these Web services operate.

Figure 3 is about the interactions between service contexts, security contexts, and security policies repository. Interaction (1) is about sensing the environment so that context data are collected for the purpose of populating the content of service contexts (I/W/C-Contexts). The sensing and collection mechanisms are described in (Maamar et al., 2005). Once completed, there is a flow of contextual information from service context to security context (Figure 3, (2)). Security contexts initially report on the security mechanisms that are by default set according to an initial state of the content of I/W/C-contexts. This default set-up is reported at the security-context level. The security mechanisms and thus, the security contexts will be changed on the fly according to the changes of the service contexts. Changes in service contexts (Figure 3, (3)) are detected as per the context sensing and collection mechanism described in (Maamar et al., 2005). The security policy will be determined/updated according to the security contexts (the security information flow is shown in Figure 3, (4-5)). Finally this feedback will be sent back to service contexts (Figure 3, (6)).

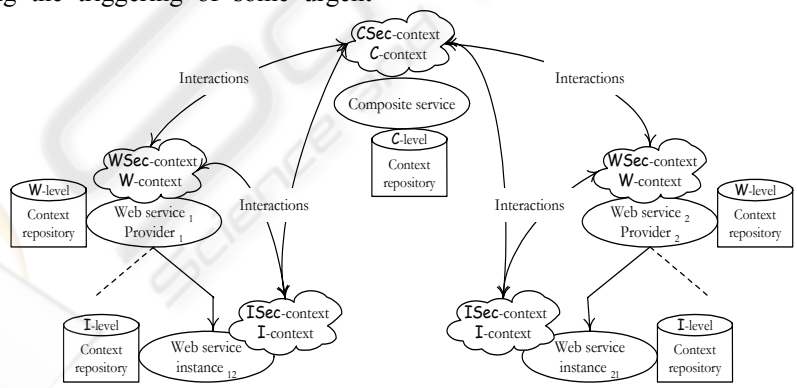


Figure 2: Context interactions during Web services composition.

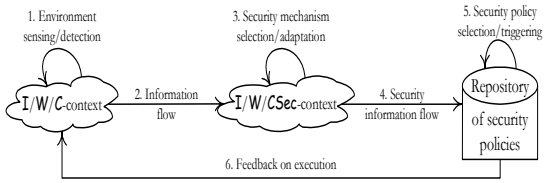


Figure 3: Interactions between contexts and security policy.

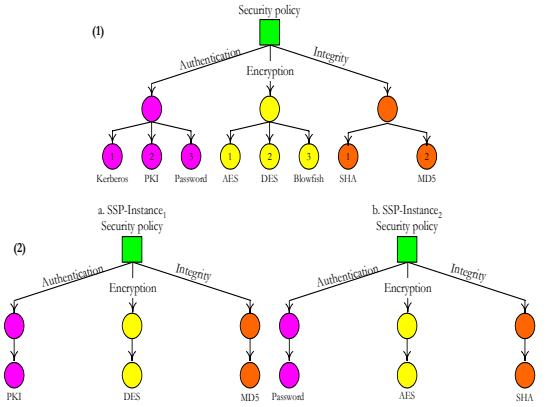


Figure 4: Sample structure of a security policy proposal and an SPP-Instance.

Setting up default policies for security requirements is motivated by the partial observability that features a Web services composition scenario. It is in general not obvious to exactly know all types of threats that a Web service will be subject to. Limited means of sensing and lack of information inhibit removing this uncertainty. As an alternative, a Web service has to consider the potential states in which it will be after initiating interactions or performing actions. Subject to these states, a reevaluation of the security policies is undertaken.

In general, every provider has an independent security policy for protecting its Web services and their contexts. Because composite services have component Web services from independent origins, conflicts could emerge including those related to policies. This calls for a reconciliation strategy. During service composition, interactions happen between Web services and thus, need to be secured. For this purpose establishing a common security-policy at design time is not feasible; service requirements are dynamic and change over time. Our solution is based on a *Security Policy Proposal* (SPP). It is assumed that each composite service will be associated with a SPP to be represented as a directed acyclic graph. Before triggering any composite service, policies of component services are reconciled through negotiation (Section 4). Each negotiation session generates an SPP of type instance. An SPP instance depicts the agreed upon security policies that will be followed by the participating Web services.

In Figure 4-(1), we show a sample structure of a security-policy proposal so that an initial agreement between providers of Web services can be reached. The values in Figure 4-(1) are the default policies originally set up as described in Figure 3. In Figure 4-(1), the graph structure shows a security policy that is dedicated to authentication (Kerberos, PKI, password), encryption (AES, DES, or Blowfish), and integrity (SHA or MD5) of a Web service. Authentication, encryption, and integrity algorithm priorities are mentioned in the form of numerical values (e.g., 1 has more priority than 2). In Figure 4-(2), we depict the structure of an SPP-Instance, which complies with its respective SPP. For example, SPP-Instance$_1$ adopts PKI for authentication, DES for encryption, and MD5 for Integrity. With regard to SPP-Instance$_2$, authentication type is password, encryption type is AES, and integrity type is SHA.

3.3 Security Policy Definition

We discuss the policies that are defined along with the security of contexts of services. To this end, a policy definition language is deemed appropriate. The selection of this language is guided by some requirements that need to be satisfied as Tonti et al. report in (Tonti et al., 2003): expressiveness, simplicity, enforceability, and scalability. In this paper, we employ Ponder (Damianou et al., 2001).

In Ponder, *authorization policies* define what activities a member of a subject domain can perform on objects in the target domain. In this paper, subject maps onto composite service (e.g., travel) and target maps onto Web service (e.g., payment). For the sake of simplicity, we depict one security policy of type authorization, each associated with a function for authentication, integrity and encryption.

Security policy of Travel composite service:

```
inst auth+ airlinesecuritypolicy{
    subject s = /travelservice;
    target t= /paymentservice;
    action authorize_payment(),check_balance(),
    airticket_details();
    when  s.authentication_algorithm("Kerberos",1)
    and
    s.integrity_algorithm("MD5",1) and
    s.encryption_algorithm("AES-128",1) and
    t.authentication_algorithm("PKI",2) and
    t.integrity_algorithm("SHA1",2) and
    t.encryption_algorithm("3DES",2);}
```

In Ponder a policy comprises several arguments. We focus, hereafter, on subject domain and target domain arguments. The subject domain categorizes the members that perform actions over the members of the target domain. We identify two types of member in the subject domain: composite service and Web service. Similarly, we identify a single type of member in the target domain: Web service.

4 POLICY NEGOTIATION AND RECONCILIATION

Appendix 1 illustrates how negotiation on the type of authentication mechanism, encryption technique, and integrity happens between travel composite-service and airline and payment component Web services. Initially, travel composite-service gets a booking request from a user (through his user-agent) with the necessary (complete) details.

To satisfy the user's request, airline and payment services are responsible for booking ticket and payment related issues, respectively. Before both services engage in interactions, the composite service establishes an SPP request between the component services (Appendix 1, (2) and (3)). If the SPP request is acceptable to both, airline and payment services, each service proceeds as reported in (4) through (7) of Appendix 1. Otherwise, travel composite-service needs to generate a new SPP request. After establishing an SPP, travel composite-service requests for an airline ticket with appropriate details. Then, airline service creates a service instance to satisfy travel composite-service request with the established security policy. After that travel service requests payment service for performing credit-card payment. For this, payment service creates a service instance with its established security policy.

Similarly, payment service instance needs the output of airline service instance. For this, there should be a common policy between both these services with regard to encryption/decryption technique, integrity algorithm, and authorization mechanism. Interactions (13) through (20) of Appendix 1 present the negotiation process between airline service instance and payment service instance. Initially airline service instance creates an SPP-Instance-1 as an instance of SPP and makes a request to payment service instance. If payment service instance does not accept, then further SPP-Instances (SPP-Instance-2, etc.) will have to be generated by either service instance, as further instances of SPP. This process keeps on running until the instances reach a mutual agreement. At the end of the policy reconciliation process, airline service instance sends the necessary details (e.g., flight name, amount, etc.,) for air-ticket payment to payment service instance. And the payment service instance also provides the confirmation of payment. Finally air-ticket is delivered to user.

It is noted that the work in (Wang et al., 2004) is about an algorithm and tool for policy reconciliation in a distributed computing environment. Our algorithm is similar to this one, but is tailored to context-driven Web service environments.

5 CONCLUSION

In this paper, we described our policy-based mechanism for securing contexts of services by leveraging our earlier work on context ontologies for Web services (Maamar et al., 2005). Our mechanism includes a policy negotiation protocol among the participating Web services, via the instantiation of a security policy proposal by the participants in the composition. Our approach takes benefit of other works by providing a rallying framework that enables securing Web services using context-driven policies. At present our proposal relies first, on Ponder to specify security policies and second, on context to trigger the appropriate policy. In addition, in this framework security policies are not specific to access control (like (Leune et al., 2004), (Damianou et al., 2001), and (Agarwal and Sprick, 2004)) but target the security mechanisms to enforce between Web services, users' client applications, and resources. These mechanisms are authentication, cryptographic, and integrity. We have implemented all the above mentioned ideas in the Context based semantic Web Services prototype for modeling context-based semantic Web services (Sattanathan et al., 2005).

ACKNOWLEDGEMENTS

The first author was supported by the Center for Advanced Studies program of IBM Software Labs India. He also thanks Prof. K. C. Shet of NITK, for supporting his doctoral work.

REFERENCES

Agarwal, S., and Sprick, B. (2004). Access Control for Semantic Web Services. In *Proc. of The 2nd IEEE Int. Conf. on Web Services,* San Diego, CA, USA.

Agarwal, S., Sprick, B., and Wortmann, S. (2004). Credential Based Access Control for Semantic Web Services. In *Proc. of The 2004 American Association for Artificial Intelligence Spring Symposium Series,* Stanford, CA, USA.

Berardi, D., Calvanese, D., De Giacomo, G., Lenzerini, M., and Mecella, M. (2003). A Foundational Vision for E-Services. In *Proc. of the Work. on Web Service, E-Business, and the Semantic Web held in conjunction with the 15th Conf. on Advanced Information Systems Engineering*, Klagenfurt/Velden, Austria.

Bhatti, R., Bertino, E., and Ghafoor, A. (2004). A Trust-

based Context-Aware Access Control Model for Web Services. In *Proc. of The 2nd IEEE Int. Conf. on Web Services,* San Diego, CA, USA.

Casati F., and Shan, M.C. (2001). Dynamic and Adaptive Composition of E-Services. *Information Systems*, 26(3).

Damianou, N., Dulay, N., Lupu, E., and Sloman, M. (2001). The Ponder Specification Language. In *Proc. of the Work. on Policies for Distributed Systems and Networks,* Bristol, UK.

Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., and Samarati, P. (2001). Fine Grained Access Control for SOAP E-Services. In *Proc. of the 10th Int. World Wide Web Conf.,* Hong Kong, China.

Damiani, E., De Capitani di Vimercati, S., Fugazza, C., and Samarati, P. (2004). Extending Policy Languages to the Semantic Web. In *Proc. of the Int. Conf. on Web Engineering,* Munich, Germany.

Hu, J., and Weaver, A.C. (2004). A Dynamic, Context-Aware Security Infrastructure for Distributed Healthcare Applications. In *Proc. of The 1st Work. on Pervasive Security, Privacy, and Trust held in conjunction with in Conjunction with The 1st Annual Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services*, Boston, MA, USA.

Kouadri Mostefaoui, G. (2004). Towards a Conceptual and Software Framework for Integrating Context-Based Security in Pervasive Environments. *Ph.D. Thesis No. 1463, University of Fribourg, Switzerland*, October.

Kouadri Mostefaoui, G., and Brézillon, P. (2004). Modeling Context-Based Security Policies with Contextual Graphs. In *Proc. of The Work. on Context Modeling and Reasoning held in conjunction with The 2nd IEEE Int. Conf. on Pervasive Computing and Communication,* Orlando, Florida, USA.

Leune, K., van den Heuvel, W.J., and Papazoglou, M. (2004). Exploring a Multi-Faceted Framework for SOC: How to Develop Secure Web Service Interactions? In *Proc. of The 14th Int. Work. on Research Issues on Data Engineering*, Boston, USA.

Lilly, J. (2004). Tips and Tricks: Web Services Attacks and Defenses (White Paper). January 2004 (osdn.bitpipe.com/detail/RES/1080320572_938.html), visited June 2004.

Lupu, E., and Sloman, M. (1999). Conflicts in Policy-Based Distributed Systems Management. *IEEE Transactions on Software Engineering*, 25(6), November/December.

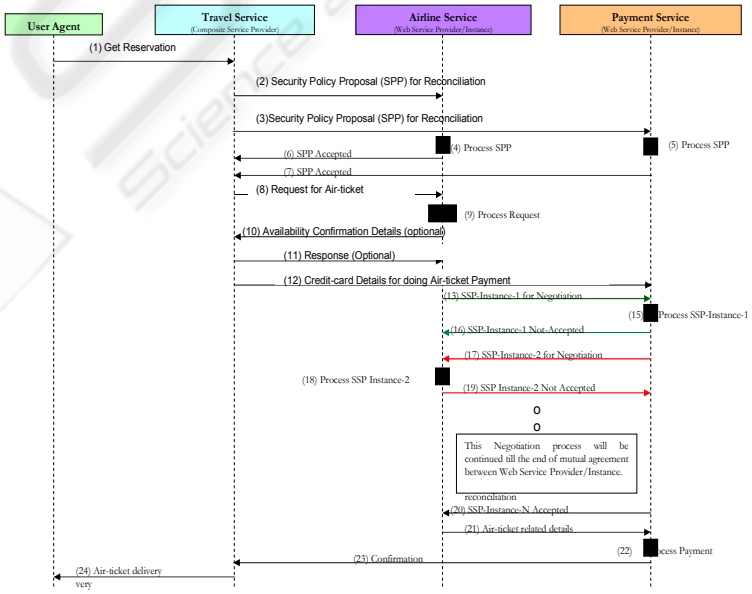
Maamar, Z., Kouadri Mostéfaoui, S., and Yahyaoui, H. (2004). A Web Services Composition Approach based on Software Agents and Context. In *Proc. of 19th Annual ACM Symposium on Applied Computing,* Nicosia, Cyprus.

Maamar, Z., Narendra, N.C., and Sattanathan, S. (2005). Towards an Ontology-based Approach for Specifying and Securing Web Services. In *Information and Software Technology* (forthcoming).

Sattanathan, S., Narendra, N.C., and Maamar, Z. (2005). ConWeSc - Context-based Semantic Web Services Composition Towards an Ontology-based Approach for Specifying and Securing Web Services. In *Proc. of The 3rd Int. Conf. on Service Oriented Computing*, Amsterdam, The Netherlands, December.

Tonti, G., Bradshaw, J., Jeffers, R., Montanari, R., Suri, N., and Uszok, A. (2003). Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder. In *Proc. of The 2nd Int. Semantic Web Conf.*, Sanibel Island, Florida, USA.

Wang, H., Jha, S., Livny, M., and McDaniel, P. D. (2004). Security Policy Reconciliation in Distributed Computing Environments, 2004. In *Proc. of the 5th Int. Work. on Policies for Distributed Systems and Networks,* New York, USA.



Appendix 1: Policy Negotiations among Travel Airline and Payment Services.