Int. J. Web and Grid Services, Vol. X, No. Y, XXXX

New XACML-AspectBPEL approach for composite web services security

Sara Ayoubi and Azzam Mourad*

Department of Computer Science and Mathematics,
Lebanese American University, Lebanon
Email: sara.ayoubi@lau.edu.lb
Email: azzam.mourad@lau.edu.lb
*Corresponding author

Hadi Otrok

Department of Electrical and Computer Engineering,
Khalifa University of Science, Technology and Research, UAE
Email: hadi.otrok@kustar.ac.ae

Ahmad Shahin

CIS Department,
Lebanese University, Lebanon
Email: ashahin@ul.edu.lb

Abstract: Web services technology is the latest evolution in distributed computing. With all of the advantages of web services, one of the main hurdles remains security in composite web services. In this paper, we tackle this problem through a new approach towards the integration of security into the BPEL (Business Process Execution Language) process of composite web services. Our approach allows specifying the XACML (eXtensible Access Control Markup Language) policies that determine join points in a BPEL process where security is needed. Subsequently, BPEL flows with the needed security are generated into AspectBPEL security aspects to be weaved in the aforementioned process. The main contributions of our approach are: (a) describing dynamic security policies using a standard language XACML, (b) generating automatically the AspectBPEL aspects of the XACML policies and (c) separating the business and security concerns of composite web services, hence developing and updating them separately at the BPEL side.

Keywords: web services security; XACML; BPEL; security; AOP; RBAC.

Reference to this paper should be made as follows: Ayoubi, S., Mourad, A., Otrok, H. and Shahin, A. (xxxx) 'New XACML-AspectBPEL approach for composite web services security', *Int. J. Web and Grid Services*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Sara Ayoubi holds an MSc in Computer Science from the Lebanese American University (LAU). She also holds a bachelor degree in Computer Science from the Lebanese University. The topics of her research activities are security engineering, web services security, security policies and cloud computing.

Copyright © 200X Inderscience Enterprises Ltd.

Azzam Mourad is an Assistant Professor in the Department of Computer Science and Mathematics at the Lebanese American University (LAU). He holds PhD degree in Electrical and Computer Engineering from Concordia University, Canada and MSc degree in Computer Science from Laval University, Canada. The main topics of his current research activities are web services security, web services engineering, aspect-oriented programming, ad-hoc network security, information security, software security and security engineering. He is currently serving as program chair, TPC member and/or reviewers of several international conferences and journals. In the past, he served as Postdoctoral fellow at Concordia University.

Hadi Otrok is an Assistant Professor in the Department of Computer Engineering at Khalifa University. He received his PhD in ECE from Concordia University, Montreal, Canada. His research interests are mainly on network and computer security. He has interest in resources management in virtual private networks and wireless networks. His PhD thesis is 'Intrusion Detection System (IDS) using Game Theory and Mechanism Design'. Before joining Khalifa University, he was holding a postdoctoral position at the École de technologie supérieure. He is serving as a technical programme committee member for different international conferences and regular reviewer for several journals.

Ahmad Shahin has a PhD in Computer Science from La Rochelle University – France. He has worked on Doppler Color Aliasing Correction with the Cardiology Center of Poitiers Hospital from 1994 to 1998. Since 1999, he is lecturing at the Lebanese University (LU). Currently, he is part of the Laboratory of Mathematics and their Applications of LU. For several years, he was the Head of the CIS Department at LU. His research focus on image processing and data compression, identification using biometrics, and web services. He is the Chairman of the IT Association in Lebanon and IEEE member.

1 Introduction

Web Services are the latest evolution in distributed computing. This technology allows access to services over a network through a combination of Extensible Markup Language (XML), Web Service Description Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description, Discovery and Integration (UDDI). While it may seem very similar to client/server applications, web services do not offer a GUI to the client and its functionality focuses on sending and processing data in a peer-to-peer fashion. It has gained a lot of popularity for being both platform and language independent. In addition, it offers an ideal framework for service and data exchange between partners. With all of the advantages of web services, one of the main hurdles remains security.

In this regards, several approaches were proposed to ensure security for web services, such as XACML for access control, SAML for authentication and WS-Security that offers various encryption and dynamic authentication and access control features for web services (Lockhart, 2008; Moses, 2011; Atkinson, 2006). However, these approaches focus on embedding security (e.g. tokens, encryption, keys etc.) in the SOAP message header of each web service. This may cause conflicts among security policies, redundancy of security measures, and imposes an enormous execution overhead in a process that orchestrates between multiple web services. BPEL is one example of a language that provides orchestration between several web services that are referred to as "partners". In the current form of BPEL use, it is only given the responsibility of business level orchestration, while security is left to each individual web service to deal with when

invoked by a user. When a client invokes a BPEL process, this latter invokes on his behalf every web service in the process. Thus, the aforementioned problems related to conflicts and redundancy are still valid here because security measures (e.g. authentication, access control, credentials verification, etc.) of the same user will be executed at each web service.

In this paper, we address the aforementioned problems and introduce a new approach for composite web services security. It aims to eliminate conflicts among policies and reduce the overhead by verifying the credentials of users at the BPEL side. Our approach is based on a synergy between XACM, AOP and BPEL, and built on top of our AspectBPEL language (Mourad et al., 2012). XACML offers the capability of describing the security policies required for a system, while AOP allows specifying the security concerns in separate components called aspects. A BPEL trust enforcer ensures the application of the security policies at the BPEL side. The main contributions of our approach, in addition to the aforementioned advantages are:(1) describing the security policies using a standard policy language (XACML), (2) generating BPEL aspects conformed to XACML policies, (3) separating the business and security concerns of composite web services, and hence developing them separately, (4) allowing the modification of web services composition at run time to integrate, remove and/or update security mechanisms and (5) providing modularity for modeling security cross-cutting concerns between web services.

To demonstrate the feasibility of our proposition, we have developed a Flight Reservation System (FS) that is composed of several web services. A RBAC (Role Based Access Control) model for the flight reservation system, which we called RBAC-FS, is elaborated where its security requirements are specified using XACML. The trust enforcer parses these policies and generates BPEL aspects that integrate the security functionalities into the BPEL process. The XACML policies and their corresponding aspects provide authentication and access control features to the flight reservation system. Case studies and experimental results are presented to defend our propositions.

The rest of the paper is organised as follows. In Section 2, we discuss the related work. Section 3 is devoted to the description of the flight reservation architecture. In Section 4 we present our the XACML-AspectBPEL architecture. In Section 5, we illustrate our approach's design and implementation. In Section 6 we illustrate our proposed approach. In Section 7 we show our experimental results. Finally, we give some concluding remarks in Section 8. A demo video of the XACML-AspectBPEL framework is available on: http://youtu.be/khzp-a0ey3I.

2 Related Work

In this section, we provide an overview on the related work in the area of Web services security.

Security Assertion Markup Language (SAML) (Lockhart, 2008) proposed by OASIS is an XML-based specification language used to specify security credentials, which are expressed as assertions. It allows to manage secure between organisations, from basic password authentication, to SSL and X. 509 certificates.A security token is delivered to the requester after successful authentication. This security token allows granting certain permissions to the requester. Oasis also proposed another standard language for access control: The eXtensible Access Control Markup Language (WS-XACML) (Moses, 2011). WS-XACML is designed to define authorisation policies for subjects that are specified using XML.

In the line of standards in the area of web services security, IBM in collaboration with Microsoft and Verisign proposed WS-Security (Atkinson, 2006). WS-Security is a mean for using XML to encrypt and digitally sign SOAP messages. Another feature of WS-Security is allowing exchanging security tokens for authentication and authorisation of SOAP messages.

Hummer et al. (2011) introduced an integrated approach for identity and access management (IAM) in a SOA Context. Their approach is based on the elaboration of a domain specific language (DSL), to define an IAM policy that enforces role-based access control security for SOAs. The main contribution of their work is specifying the RBAC permission with regards to a certain context, and matching each context element to a WS-BPEL scope element. This context element allows single-sign on by reusing the same SAML assertion for each activity within a single scope. With every scope change, a new SAML assertion is generated for the corresponding role and context. However, our approach adopts an aspect oriented mechanism to weave security aspects in a WS-BPEL at any WS-BPEL element, and is not restricted to a scope element. Also, our approach offers the ability to dynamically update the WS-BPEL composition at run-time to adapt to new security requirements and separation of concerns between security and business logic.

X-RBAC (Bhatti, 2003) is an XML-based RBAC policy specification framework for enforcing access control in dynamic XML-based Web services. The specification uses representations of users, roles and permissions. The two main components of the proposed framework are: the XML and the RBAC processors. The XML processor is implemented in Java using Java API for XML Processing (JAXP). Some modules have the duty to get the DOM instance of parsed XML documents and forward them to the RBAC Processor. The RBAC module is responsible for administration and enforcement of the policy according to the supplied policy information.

The Business Process Constraint Language (Paci et al., 2008) was introduced by Paci et al. to allow the specification of authorisation policies and constraints for WS-BPEL business processes. Ardagna et al. (2006) proposed a design of a Web service architecture for enforcing access control policies. They also provided an example of implementation based on the WS-Policy (Schlimmer, 2004; Nolan, 2004) as access control language. The main components of the proposed architecture are: Policy Administration Point (PAP), Policy Evaluation Point (PEP) and Policy Decision Point (PDP). The PAP module is a policy repository that provides an administrative interface for inserting, updating, and deleting policies. The PEP module realises the enforcement of the policies returned by the PAP module. The access request is granted if at least one policy is satisfied; the access is denied otherwise. A PDP module is the interface between the service and the enforcer module. It is responsible for taking final access control decisions based on the input from the PEP module.

All of the aforementioned approaches target the security policies implementation, deployment and/or verification at the Web services side. However, they do not address any of the aforementioned problems that occur at the composite web service level(e.g. dynamic adaptation, services interruption and performance). On the other hand, our approach relies on enforcing the security policies at the composite web service and reducing the amount of overhead imposed by standard approaches.

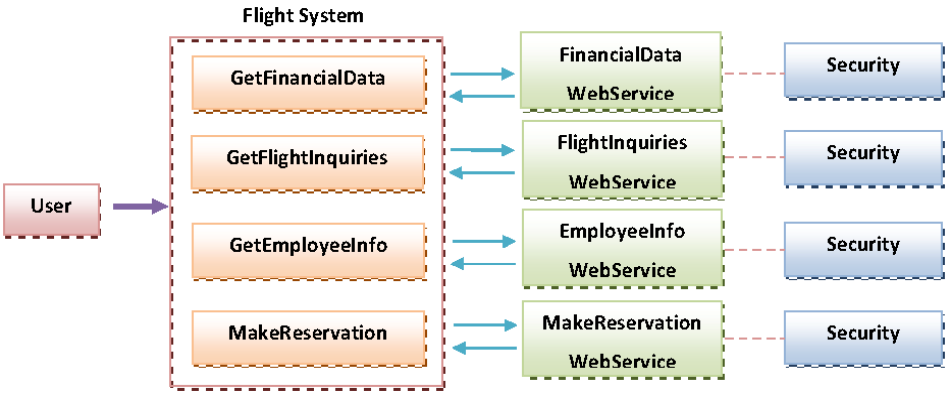
In the same line of research, Charfi and Mezimi (2004) introduced a tool called AO4BPEL, which an aspect oriented extension for BPEL that offers modularity and adaptability to workflow processes. The join points are represented by activities in the BPEL process. Pointcuts are represented in the XPATH language and advices are the BPEL activities to be added. This work has been extended in the Cooperative Aspect Oriented Programming for Executable Business Processes (Co-AOP) tool, which aims at

making the aspects reusable (Di Francescomarino and Tonella, 2009). An aspect code is developed for a specific BPEL process, which makes it difficult to reuse. Co-AOP alleviates that challenge by introducing what is called the Explicit Join Points (EJP). These EJPs allow the base code to be aware of the aspect interfaces, and hence improve aspect reusability by decoupling base code and aspects. The aspects are initiated in the base code and described in their advices code, which forces the communication to be parameterised between both the base and aspects codes. AO4BPEL offers the BPEL process the ability to adapt to future changes in the BPEL process. However, AO4BPEL has few limitations. First, it requires the use of a special orchestration engine to manage the BPEL process, which makes it incompatible with the major adopted BPEL development environments such as Eclipse, NetBeans, etc. Second, their approach induces some performance overhead since it performs a check on each activity in the process to determine whether or not their aspect code is associated with it. On the other hand, our approach proposes a framework that is fully operational on any BPEL process regardless of the adopted development environment. Moreover, our approach reduces enormously the overhead since it is based on intercepting only selective join points, i.e., only those, which are associated with the aspect code.

3 Access control policy specification for the fight system

In this section, we describe the architecture of the flight system BPEL process and its partner web services. Figure 1 explores the interactions between the users, the BPEL process of the flight system, and the web services. Our Flight System is mainly composed of three separate web services, a BPEL process and a graphical user interface.

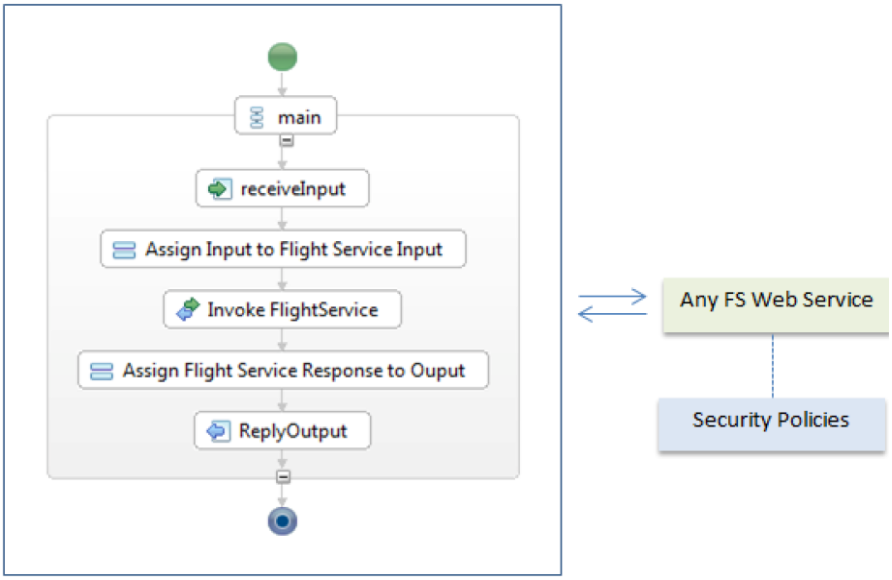
Figure 1 FS architecture (see online version for colours)



First, the financial data service allows the user to request the revenues and expenses of the flight agency for a given month. Second, the flight inquiry service returns a list of the available flights including the time and date of the departure and arrival, the airline, the number of available seats and ticket prices. The employee information service allows the user to view information about the flight system staff by ID. This information includes the employee's full name, phone number, email address, post and office number. Finally, the make reservation service enables the user to reserve a seat on a certain flight.

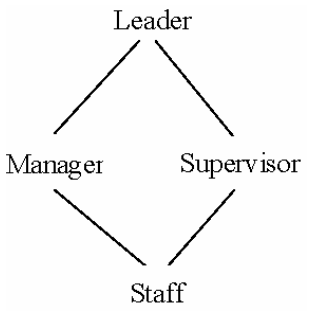
Figure 2 illustrates a part of the flight system process. For space restriction, the figure only explores one service invoke. We call this web service *AnyFSWebService*. The process is invoked when a user requests one of the services offered by the FS. The process begins by assigning the user's input to the FlightService request message, then invokes the requested service and returns the needed info. Finally, the service's response message is forwarded to the client. Each time a user wishes to access one of the flight system services, his credentials are passed to each web service catering his request, in order to authenticate and authorise the user.

Figure 2 FS BPEL process (see online version for colours)



The access control in our flight System is role-based and consists of four different roles. We will refer to it as RBAC-FS. The highest role is the *leader* which has access to all the available services. The *supervisor* and *manager* have less access rights than the leader but more access rights than the *staff* members. The staff role has the least access rights. Figure 3 illustrates the role hierarchy of the flight system access control. Listings 1 and 2 (included in Section 6 for paper readability) present the XACML specification of the corresponding RBAC-FS.

Figure 3 RBAC-FS role hierarchy



4 XACML-AspectBPEL architecture

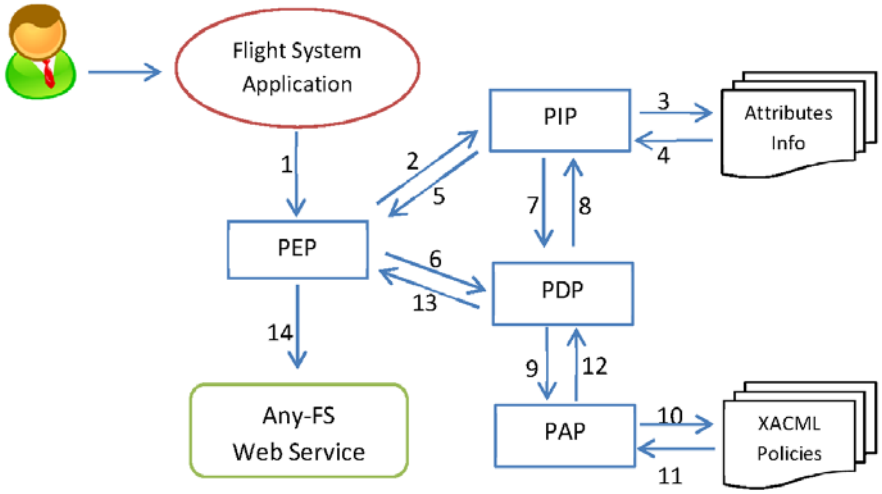
Security is one of the software aspects that are very important to deal with. Generally, developers describe in ad-hoc manner the required security rules and integrate them directly in their code. Moreover, they do not separate between security and business logic code. This means that any change in the security strategy has to be done on the application code, which can have an impact on the business logic of web services. XACML and AOP contribute to solve these issues by specifying the security policies in XML-based documents, then embedding them in aspects. Aspects allow defining and integrating security objects, methods and events within application, which make them interesting solutions for many security issues. Many contributions (Shah, 2003; DeWin, 2004; Bodkin, 2004; Huang et al., 2004; Pavlich-Mariscal, 2007; Fuentes and Sanchez, 2006; Evermann, 2007; Kiczales, 2001; Kiczales et al., 1997; Slowikowski and Zielinski, 2003; Sun et al., 2009; Wu-Lee and Hwant, 2010) have proven the usefulness of AOP for integrating security features into software. Moreover, previous approaches (Moses, 2011; Atkinson, 2006; Lockhart, 2008) explored the usefulness and efficiency of specifying security policies in standard languages like XACML. Using such languages also enforce the concept of separation of concerns, which is one of the main advantages of AOP. The main objective of AOP is to have a separation between cross-cutting concerns. This is achieved through the definition of aspects. Each aspect is a separate module in which pointcuts are defined. A pointcut identifies one or more join points. A join point identifies one or many flow points in a program (in our case a program is a BPEL process). At these points, some advices will be executed. An advice contains some code that can alter the process behavior at a certain flow point. The integration of aspects within the application code is called weaving and is performed through one of the weaving technologies (e.g., AspectJ; Kiczales, 2001).

Our approach is based on a synergy between XACM, AOP and BPEL, and built on top of our AspectBPEL language (Mourad et al., 2012). XACML (Moses, 2011), in addition to other standard languages (Lockhart, 2008; Atkinson, 2006), are very useful for the organised description of complex and composed security policies. They allow avoiding the ad-hoc description of security rules and specifying them in XML-based document. The XACML infrastructure as depicted by the Oasis standard consists of 4 components that can be distributed over the network as web services themselves:

- Policy Administration Point (PAP): Stores XACML Access Control Policies
- Policy Information Point (PIP): Hosts attributes about users and services
- Policy Decision Point (PDP): Decides about granting and denying access to a resource
- Policy Enforcement Point (PEP): Enforces a PDP's access decision and grants or deny physical access.

To better illustrate the XACML infrastructure and the way these 4 components work together to provide authentication and access control security to web services, consider Figure 4.

Figure 4 XACML infrastructure (see online version for colours)



In Figure 4, a user wants to access one of the services offered by the FS. As we have previously mentioned, this system is composed of 4 separate web services. When the user wants to access one of these services, the access request is directed to this web service's PEP. The PEP accesses the PIP that stores information about the user in order to authenticate him. Subsequently, the PIP returns a SAML token with the corresponding authentication response. Next, the PEP accesses the PDP in order to retrieve the user's access rights. The PDP invokes the PAP that stores the XACML policies to return the policy of the given resource and the PIP to retrieve the SAML token. With the SAML token and the XACML policy, the PDP returns to the PEP whether this user is granted or denied access to the requested web service. Finally, the PEP either directs the access request to the web service or returns to the application with a response that the user has been denied access to the requested resource.

The motivation behind the need for a composite web services security is to replace the monopolisation of web services security by placing it at the BPEL side. This shift will reduce the overhead imposed by restricting security checks at individual invoke activities. In its current form, every web service invoke in the BPEL process, goes through the PEP, PIP, PDP and PAP of this corresponding service. This causes a lot of overhead and dramatically reduces the performance of the BPEL process, as demonstrated by our performance analysis in Section 7. For this purpose, we have accommodated our approach with a trust enforcer, where each partner web service can securely place their security policies and allow the process to manage security at its side. In addition, we have consolidated our approach with a systematic mechanism for identifying selective join points where security checks need to be integrated, rather than a dogmatic call for security at each invoke activity.

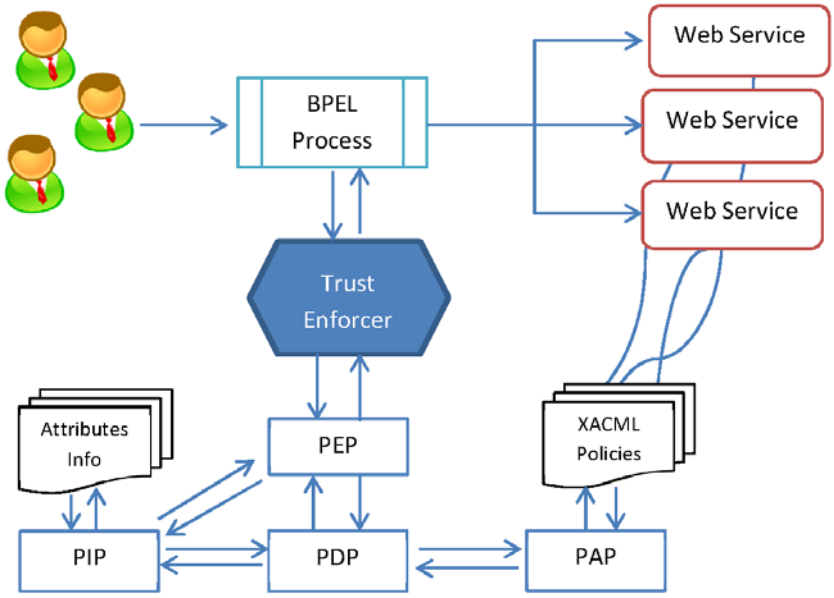
To better illustrate the importance of composite web services security let's consider our Flight System illustrated in Figure 1 and let's consider a process where a leader would want to do a check on the monthly sales then check the flight inquiries to make sure that the sales match with the amount of tickets sold for a given month. In this case, the flight reservation process will consist of a sequence of two web services invoke: First, the process will invoke the financial data web service to retrieve the sales figures of the current month and then it will invoke the flight inquires web service to retrieve the

amount of tickets sold. The retrieved information will either be used by the leader to manually ensure that the sales figures match the amount of tickets sold, or it will be directed to an automated function to do the required check.

As the process invokes the financial data web service, it will pass through the service's PEP to check whether this user has the right to access the requested data. Once the request for access is identified, the process flow will resume and another access to the PEP of the flight inquires web service is needed to check whether that same user has access to the requested resource. A closer look to this flow shows the overhead due to the need of security checks at each web service invoke.

Our suggested approach is depicted in Figure 5 that consists of moving security from the web service side to the BPEL side in case of composite web services. Each web service will deploy its XACML policy at the trust enforcer's PAP.

Figure 5 Approach architecture (see online version for colours)



The trust enforcer adopts an XACML infrastructure to enforce security at the BPEL level. The trust enforcer through its access to the XACML policies will generate aspects for selective join points in the BPEL process where security checks are needed. These aspects are herein after referred to as AspectBPEL aspects. When the BPEL process receives a request, it directs it to the trust enforcer that will trigger the chain of calls to the 4 components of the XACML structure in Figure 5. As a result, it either grants the user access to the requested service or returns an "access denied" message to the user.

Selective join points are the locations identified in the BPEL process where security checks are required. These locations are found by parsing the XACML policies and matching resources and actions in these policies to "invoke" activities in the BPEL Process. The advantage of this approach is to avoid security checks on "invoke" activities where security isn't needed, and instead immediately direct the request to the corresponding service.

To better illustrate the selection of join points consider the XACML policy presented in Listing 1 and 2. The permission to get current month sales rule indicates that a leader is allowed access to the FinancialDataWS to get the Current month sales. Since there is specific authentication requirements imposed, this rule will be translated into an AspectBPEL aspects that will be weaved accordingly before the "invoke" activity of the financial data web service that calls the "GetMonthlySales" operation. On the other hand, the Permission to get the login window does not require any authentication or access control and thus this activity is not considered among the selective joint point and will be processed directly.

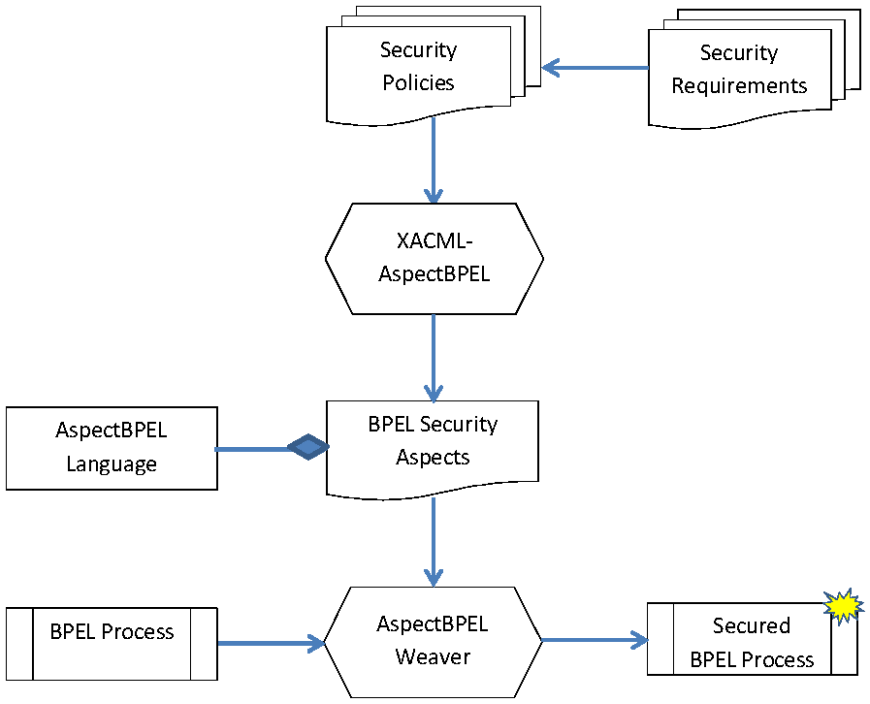
It is important to note that our approach doesn't omit security from the web service side. When a web service becomes part of a composition (such as a BPEL process), our framework would ensure proper security checks at the process side. However, each web service will keep its individual XACML structure to serve requests from other clients that are not directed through the BPEL process.

5 Approach design and implementation

In this section, we will describe the structure of our XACML-AspectBPEL platform that is integrated at the trust enforcer side for identifying selective join points and generating the corresponding security AspectBPEL aspects to be weaved at the BPEL side. A demo video of the XACML-AspectBPEL framework is available on: http://youtu.be/khzp-a0ey3I.

The structure of our XACML-AspectBPEL framework is illustrated in Figure 6

Figure 6 Approach schema (see online version for colours)



5.1 XACML-AspectBPEL generator

Our XACML parser is developed in Java based on the DOM parser for xml. It begins by parsing all the resources identified and actions presented in the policy. Each action on a resource is matched to an "invoke" activity in the BPEL process.

For each join point, an AspectBPEL aspect is generated. These aspects are referred to as security checks and consist of passing information to the trust enforcer about the action and resource the user wishes to access. If the user is granted access, the process flow resumes normally, else it returns an error message to the user indicating that he was denied access to the requested resource.

5.2 AspectBPEL weaver

In order to weave the generated AspectBPEL aspects into the specified BPEL process, our AspectBPEL weaver begins by reading the join points which represents the name and type of the activity where security checks will be inserted. Next, the weaver parses the BPEL process to find the corresponding join points in the process. Once found, the behavioral code will be inserted before, after or replace the join point respectively.

6 Illustration of the proposed approach

In this section, we present the implementation of the RBAC-FS model that illustrates all the procedures and mechanisms described in our proposed approach for the systematic enforcement of security at the process-level. In what follows, we will show the generated XACML-AspectBPEL aspects (Listing 3) realising the aforementioned XACML policy of the RBAC-FS model (Listings 1 and 2). The syntax and constructs of the AspectBPEL language will be cited or included in the final version of the paper.

Listing 1 and Listing 2 outlines a summary of the XACML-based access control policy for the flight reservation system. Normally, each web service has its own policy file, but due to space limitation, we included in the listing all the roles and permissions of the FS, but we only elaborate on the GetFlightInquiries (line 71 to line 98) and the GetMonthlySales permissions (line 33 to line 60). The others are set in similar way. First, the roles are defined. A general role (root of the hierarchy) is denoted by *leader* and has 2 sub-roles: *manager* and *supervisor*. The leader is given permission to perform any action to any resource. The manager and supervisor roles have *staff* as a common sub-role and are assigned respectively to PPS:manager:role and PPS:supervisor:role policies. The staff role has PPS:staff:role as policy. Each of the permission policies defines the set of permissions related to each role. For instance, the PPS:supervisor:role includes viewing each staff's tasks list.

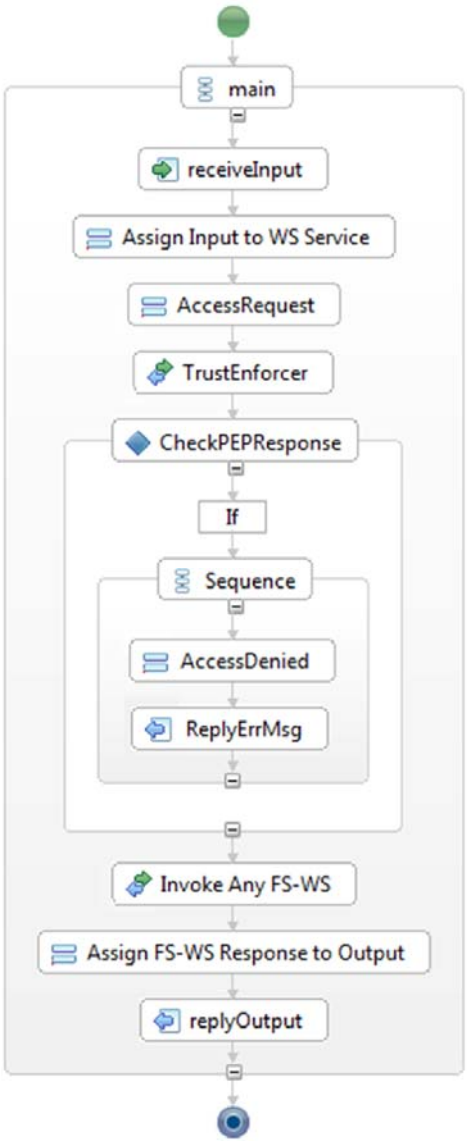
Listing 3 illustrates an excerpt of the generated AspectBPEL aspects that realise the XACML policy of Listing 1 and Listing 2. Due to space limitation, we will only show the XACML-AspectBPEL aspects generated for one sequence flow of the entire BPEL Process that we have already described in Section 4. The generated AspectBPEL aspects shows two security checks that will be weaved before the "CurrentMonthSales" invoke activity that calls the "GetCurrentMonthSales" operation of the financial data web services, and before the "CurrMonthFlightInquiries" invoke activity that calls the "GetCurrMonthFlightInquiries" of the flight inquiries web service. Each of these security

AspectBPEL aspects consists of calling the trust enforcer, to check whether the current user is allowed to access the requested operations. The trust enforcer returns a permit or deny response, and the process resumes its work accordingly.

7 Discussion and experimental results

The proposed framework generates the AspectBPEL aspects in Listing 3 from the XACML policy in Listing 1 and 2, then weaves them in the BPEL process of the Flight Reservation System presented in Figure 2 to produce the secure BPEL process illustrated in Figure 7.

Figure 7 FS Secure BPEL process (see online version for colours)



Listing 1 Excerpt of XACML policy for FS-Part I

```
[1]. <PolicySet>
[2]. <!--Permission To Get-Login-Window-->
[3]. <Policy>
[4]. <Rule RuleId="Permission:to:Login" Effect="Permit">
[5]. <Target>
[6].     <Subjects>
[7].         <AnySubject/>
[8].     </Subjects>
[9]. <Resources>
[10].     <Resource>
[11].      <ResourceMatch MatchId="function:string-equal">
[12].          <AttributeValue DataType="xml:string">LoginWindow</AttributeValue
     >
[13].          <ResourceAttributeDesignator AttributeId="resource:resource-id"
     DataType="xml:string"/>
[14].      </ResourceMatch>
[15].   </Resource>
[16]. </Resources>
[17]. <Actions>
[18].     <Action>
[19].      <ActionMatch MatchId="function:string-equal">
[20].          <AttributeValue DataType="xml:string">Login</AttributeValue>
[21].        <ActionAttributeDesignator AttributeId="action:action-id" DataType
     ="xml:string"/>
[22].     </ActionMatch>
[23].   </Action>
[24]. </Actions>
[25]. </Target></Rule></Policy>
[26]. <!--Defining the role policy set for the leader-->
[27]. <!--Role policy set for the Leader-->
[28]. ...
[29]. <!--Permissions policy set for the Leader-->
[30]. ...
[31]. <!--Permission to Get-Monthly-Sales-->
[32]. <Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" PolicyId="
     RPS:Leader:role" PolicyCombiningAlgId="policy-combine:permit-overrides">
[33]. <Rule RuleId="Permission:To:Get:Monthly:Sales" Effect="Permit">
[34]. <Target>
[35]. <Subjects>
[36].    <Subject>
[37].      <SubjectMatch MatchId="function:string-equal">
[38].     <AttributeValue DataType="xml:string">Leader</AttributeValue>
[39].     <ResourceAttributeDesignator AttributeId="subject:role-id" DataType="
     xml:string"/>
[40].    </SubjectMatch>
[41].    </Subject>
[42]. </Subjects>
[43]. <Resources>
[44].     <Resource>
[45].      <ResourceMatch MatchId="function:string-equal">
[46].          <AttributeValue DataType="xml:string">FinancialData</
     AttributeValue>
[47].          <ResourceAttributeDesignator AttributeId="resource:resource-id"
     DataType="xml:string"/>
[48].      </ResourceMatch>
[49].     </Resource>
[50]. </Resources>
[51]. <Actions>
[52].     <Action>
[53].         <ActionMatch MatchId="function:string-equal">
[54].          <AttributeValue DataType="xml:string">GetMonthlySales</
     AttributeValue>
[55].          <ActionAttributeDesignator AttributeId="action:action-id"
     DataType="xml:string"/>
[56].      </ActionMatch>
[57]. </Action>
[58]. </Actions>
```

Listing 2 Excerpt of XACML policy for FS-Part II

```
[59]. </Target>
[60]. </Rule>
[61]. <!--Include permissions of Manager, Supervisor, Staff roles-->
[62]. </Policy>
[63].
[64]. <!--Defining the role policy set for the Manager-->
[65]. <!--Role policy set for the Manager-->
[66]. ...
[67]. <!--Permissions policy set for the Manager-->
[68]. ...
[69]. <!--Permission to Get-Flight-Inquiries-->
[70]. <Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" PolicyId="
      RPS:Manager:role" PolicyCombiningAlgId="policy-combine:permit-overrides">
[71]. <Rule RuleId="Permission:To:Get:Flight:Inquiries" Effect="Permit">
[72]. <Target>
[73]. <Subjects>
[74].     <Subject>
[75].         <SubjectMatch MatchId="function:string-equal">
[76].        <AttributeValue DataType="xml:string">Manager</AttributeValue>
[77].      <ResourceAttributeDesignator AttributeId="subject:role-id" DataType
      ="xml:string"/>
[78].    </SubjectMatch>
[79].     </Subject>
[80]. </Subjects>
[81]. <Resources>
[82].     <Resource>
[83].         <ResourceMatch MatchId="function:string-equal">
[84].      <AttributeValue DataType="xml:string">FlightInquiries</
      AttributeValue>
[85].       <ResourceAttributeDesignator AttributeId="resource:resource-id"
      DataType="xml:string"/>
[86].     </ResourceMatch>
[87].  </Resource>
[88]. </Resources>
[89]. <Actions>
[90].     <Action>
[91].         <ActionMatch MatchId="function:string-equal">
[92].         <AttributeValue DataType="xml:string">GetMonthlyFlightInquiries</
      AttributeValue>
[93].         <ActionAttributeDesignator AttributeId="action:action-id"
      DataType="xml:string"/>
[94].     </ActionMatch>
[95].     </Action>
[96]. </Actions>
[97]. </Target>
[98]. </Rule>
[99]. <!--Include permissions of Supervisor, Staff roles-->
[100]. </Policy>
[101].
[102]. <!--Defining the role policy set for the Supervisor-->
[103]. <!--Role policy set for the Supervisor-->
[104]. ...
[105]. <!--Permissions policy set for the Supervisor-->
[106]. ...
[107].  <!--Permission to Get-Employee-Tasks-Lists-->
[108].     ...
[109].     <!--Include permissions of Staff roles-->
[110].
[111]. <!--Defining the role policy set for the Staff-->
[112]. <!--Role policy set for the Staff-->
[113]. ...
[114]. <!--Permissions policy set for the Staff-->
[115]. ...
[116].  <!--Permission to Make-Reservation-->
[117]. </PolicySet>
```

Listing 3 Excerpt of generated AspectBPEL aspect for access control

```
[1]. Aspect AccessControl
[2].
[3]. BeginAspect
[4].
[5]. Before
[6]. Invoke <CurrMonthSales>
[7]. BeginBehavior
[8]. <bpel:assign validate="no" name="AccessRequest">
[9]. <bpel:copy>
[10]. <bpel:from><![CDATA["GetCurrentMonthSales"]]></bpel:from>
[11]. <bpel:to variable="PEPRequest" part="parameters"></bpel:to>
[12]. </bpel:copy>
[13]. <bpel:copy>
[14]. <bpel:from><![CDATA["FinancialDataWS"]]></bpel:from>
[15]. <bpel:to part="parameters" variable="PEPRequest"></bpel:to>
[16]. </bpel:copy>
[17]. </bpel:assign>
[18].
[19]. <bpel:invoke name="TrustEnforcer" partnerLink="TrustEnforcer" operation="
     PEP" portType="ns:engine" inputVariable="PEPRequest"  outputVariable="
     PEPResponse">
[20]. </bpel:invoke>
[21]. <bpel:if name="CheckPEPResponse">
[22]. <bpel:condition><![CDATA[$PEPResponse = "Deny"]]>
[23]. </bpel:condition>
[24]. <bpel:sequence>
[25]. <bpel:assign validate="no" name="AccessDenied">
[26]. <bpel:from><![CDATA["Access Denied"]]></bpel:from>
[27]. <bpel:to part="payload" variable="output"></bpel:to>
[28]. </bpel:copy>
[29]. </bpel:assign>
[30]. <bpel:reply name="ReplyErrMsg" partnerLink="client" operation="process"
     portType="tns:process" variable="output">
[31]. </bpel:reply>
[32]. </bpel:sequence>
[33]. </bpel:if>
[34]. EndBehavior
[35].
[36].
[37]. Before
[38]. Invoke <CurrMonthFlightInquiries>
[39]. BeginBehavior
[40]. <bpel:assign validate="no" name="AccessRequest">
[41]. <bpel:copy>
[42]. <bpel:from><![CDATA["GetCurrMonthFlightInquiries"]]></bpel:from>
[43]. <bpel:to variable="PEPRequest" part="parameters"></bpel:to>
[44]. </bpel:copy>
[45]. <bpel:copy>
[46]. <bpel:from><![CDATA["FlightInquiriesWS"]]></bpel:from>
[47]. <bpel:to part="parameters" variable="PEPRequest"></bpel:to>
[48]. </bpel:copy>
[49]. </bpel:assign>
[50].
[51]. <bpel:invoke name="TrustEnforcer" partnerLink="TrustEnforcer" operation="
     PEP" portType="ns:engine" inputVariable="PEPRequest" outputVariable="
     PEPResponse">
[52]. </bpel:invoke>
[53]. <bpel:if name="CheckPEPResponse">
[54]. <bpel:condition><![CDATA[$PEPResponse = "Deny"]]>
[55]. </bpel:condition>
[56]. <bpel:sequence>
[57]. <bpel:assign validate="no" name="AccessDenied">
[58]. <bpel:from><![CDATA["Access Denied"]]></bpel:from>
[59]. <bpel:to part="payload" variable="output"></bpel:to>
[60]. </bpel:copy>
[61]. </bpel:assign>
[62]. <bpel:reply name="ReplyErrMsg" partnerLink="client" operation="process"
     portType="tns:process" variable="output">
[63]. </bpel:reply>
[64]. </bpel:sequence>
[65]. </bpel:if>
[66]. EndBehavior
[67]. EndAspect
```

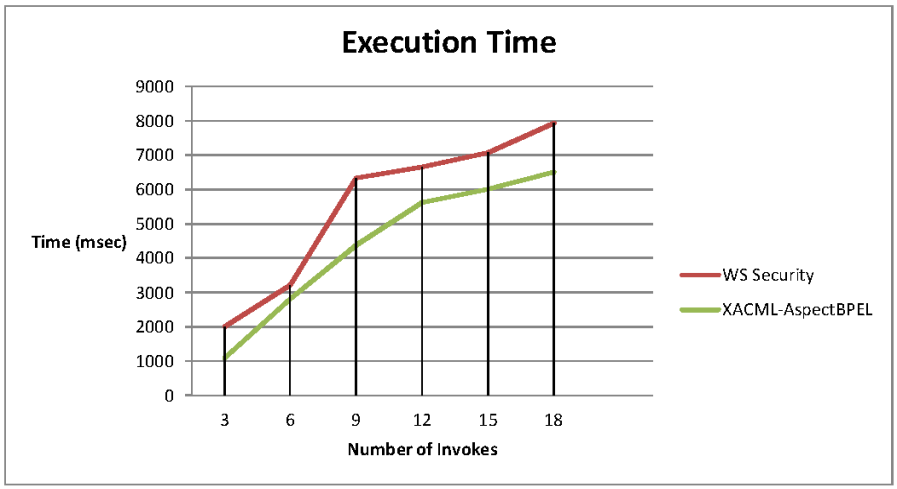
The resulted BPEL process provides dynamic authentication and access control features for the Flight Reservation system. The BPEL process begins by receiving the client's input and follows the process sequence flow. Once the process reaches an invoke activity, it invokes the trust enforcer in order to determine the user's access rights. If the access is denied, the process replies with an "access denied" message to the client. Otherwise, the process will proceed to invoke the flight reservation system web service AnyFSWebservice and return to the client the requested resource.

Verifying the successful integration of the RBAC-FS security features in the original BPEL code of the flight reservation system has been performed through extensive testing. Additional efforts have been spent on verifying that the original functionalities of the system have not been altered. Also several modifications have been applied to the security policy and reflected dynamically in the corresponding BPEL aspects. Consequently, the modification has been applied dynamically onto the BPEL process, which demonstrates the feasibility and appropriateness of our propositions.

To better demonstrate the effectiveness of our approach we have also conducted a performance analysis. This analysis allowed us to measure the variation in the execution time between a BPEL process with security on the web service side and a BPEL process with security on the process level.

Figure 8 shows the variation in the process's runtime. The process's execution runtime has been measured upon the number of invokes that the BPEL process includes, which reflects the number of participating Web services. The higher the number of invokes are, the bigger and more complex the BPEL process becomes. This helps demonstrate the scalability of our approach. The execution time has been measured using the Visual Studio Profiling Tool TPTPEclipse. This tool allows us to read the runtime of the process call.

Figure 8 Performance analysis (see online version for colours)



The process execution time chart shows two different lines: a process with security at the web service side and a process with XACML-AspectBPEL security.

New XACML-AspectBPEL approach

For the sake of the performance analysis, we built a BPEL process and augmented the number of web services invoke at each run. In the case of security at the web service side, we have developed for each web service an XACML policy and incorporated an XACML infrastructure to ensure proper security as depicted in the policy. However, in the case of security at the BPEL level, we removed the XACML components from the web service side and placed it once at the process side, with all of the web service policies at the PAP component.

Figure8 illustrates the following results:

- A BPEL process with security implemented at the Web Service level runs at 3225.33 msec for 6 web services invoke, and rises to around 8000 msec for 18 web services invoke.
- A BPEL process with security implemented at the process level runs at 2817.42 msec for 6 web services invoke, and reaches 6521.56 msec when running a process with 18 web services invoke.

These results clearly show that security at the process level considerably enhances the runtime of the BPEL process. This performance analysis shows that the XACML-AspectBPEL framework enhances the overall runtime of the BPEL process, since it alleviate the overhead of sending the security verification at web service side and also reduces unnecessary security checks.

8 Conclusion

A novel approach was proposed for the systematic enforcement of security at the process level using a separate trust enforcer with an XACML infrastructure. Our proposition is based on a synergy between XACML, AOP and a composition of web services. It eliminates conflicts among policies and reduce the overhead by verifying the credentials of users at the BPEL side.It also allows the separation between business and security concerns of composite web services, and hence developing them separately. Moreover, it permits dynamic activation of security and modification of the web services composition at runtime. The experimental results demonstrate the feasibility and appropriateness of our proposition. We also illustrated the dynamic integration, activation and modification of access control features in the flight reservation system. Finally, we conducted a thorough performance analysis to demonstrate the efficiency of our approach and its scalability in long and complex processes.

Acknowledgements

This work is supported by the Lebanese American University(LAU) and CNRS, Lebanon.

References

Ardagna, C.A., Damiani, E., De Capitani di Vimercati, S. and Samarati, P. (2006) 'A web service architecture for enforcing access control policies', *Electronic Notes Theoretical Computer Science*, Vol. 142, 2006.

Atkinson, B. (2006) *Web services security (WS-Security)*. Available online at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

Bhatti, R., Joshi, J., Bertino, E. and Ghafoor, A. (2003) 'Access control in dynamic XML-based web-services with X-RBAC', *Proceedings of the International Conference on Web Services (ICWS 03)*, 2003.

Bodkin, R. (2004) 'Enterprise security aspects', *Proceedings of the AOSD 04 Workshop on AOSD Technology for Application-level Security (AOSD 04*), 2004.

Charfi, A. and Mezini, M. (2004) 'Aspect-oriented web service composition with AO4BPEL', *ECOWS 04*, 2004.

DeWin, B. (2004) *Engineering Application Level Security through Aspect Oriented Software Development*, PhD Thesis, Katholieke Universiteit Leuven.

Di Francescomarino, C. and Tonella, P. (2009) 'Cooperative aspect oriented programming for executable business processes', *Proceedings of the the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems*, Vancouver, Canada.

Evermann, J. (2007) 'A Meta-Level Specification and Profile for AspectJ in UML', *Journal of Object Technology*, Vol. 6, No. 7, pp.27–49.

Fuentes, L. and Sanchez, P. (2006) 'Elaborating UML 2.0 Profiles for AO Design', *Proceedings of the International Workshop on Aspect-Oriented Modeling*, 2006.

Huang, M., Wang, C. and Zhang, L. (2004) 'Toward a reusable and generic security aspect library', *Proceedings of the AOSD 04 Workshop on AOSD Technology for Application level Security (AOSD 04)*, 2004.

Hummer, W., Gaubatz, P., Strembeck, M., Zdun, U. and Dustdar, S. (2011) 'An integrated approach for identity and access management in a SOA context', *Proceedings of the 16th ACM Symposium on Access control Models and Technologies (SACMAT 11)*, New York, USA.

Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J. and Griswold, W.G. (2001) 'An overview of AspectJ', *Proceedings of the 15th European Conference on Object-Oriented Programming (ECOOP 01)*, London, UK.

Kiczales, G., Lamping, J., Menhdhekar, A., Maeda, C., Lopes, C., Loingtier, J-M. and Irwin, J. (1997) 'Aspect-oriented programming', in Aksit, M. and Matsuoka, S. (Eds): *Proceedings of the European Conference on Object-Oriented Programming*, Springer-Verlag, Berlin, Heidelberg, and New York.

Lockhart, B. (2008) *OASIS Security Services TC (SAML)*. Available online at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security)

Moses, T. (2011) *OASIS eXtensible Access Control Markup Language (XACML)*, OASIS Standard 2.0. Available online at: http://www.oasis-open.org/committees/xacml/

Mourad, A., Ayoubi, S., Yahyaoui, H. and Otrok, H. (2012) 'A novel aspect-oriented BPEL framework for the dynamic enforcement of web services security', *International Journal of Web and Grid Services*, Vol. 8, No. 4, pp.361–385.

Nolan, P. (2004) *Understand WS-Policy processing*, Technical report, IBM Corporation.

Paci, F., Bertino, E. and Crampton, J. (2008) 'An Access-Control Framework for WS-BPEL', *International Journal of Web Services Research*, Vol. 5, No. 3, pp.20–43.

Pavlich-Mariscal, J., Michel, L. and Demurjian, S. (2007) 'Enhancing UML to model custom security aspects', *Proceedings of the 11th International Workshop on Aspect-Oriented Modeling*, 2007.

Schlimmer, J. (2004) *Web Services Policy Framework (WS-Policy)*. Available online at: http://www-128.ibm.com/developerworks/Webservices/library/specification/ws-polfram/

Shah, V. (2003) *An Aspect-Oriented Security Assurance Solution*, Technical Report AFRL-IF-RS-TR-2003-254, Cigital Labs.

Slowikowski, P. and Zielinski, K. (2003) 'Comparison study of aspect-oriented and container managed security', *Proceedings of the ECCOP workshop on Analysis of Aspect-Oriented Software*, 2003.

Sun, M., Li, B. and Zhang, P. (2009) 'Monitoring BPEL-Based Web Service Composition Using AOP', *Proceedings of the 8th IEEE/ACIS International Conference on Computer and Information Science*, Washington, DC, USA.

Wu-Lee, C. and Hwang, G. (2010) 'Dynamic policies for supporting quality of service in service-oriented architecture', *Proceedings of the International Conference on Electronics and Information Engineering*, Washington, DC, USA.

Websites

Ken North Computing, XML and Web Services: Message Processing Vulnerabilities. http://www.Webservicessummit.com/Articles/MessagingThreats.htm.

Terence Parr, ANTLR, http://www.antlr.org/.

The Eclipse Test and Performance Tools Platform, www.eclipse.org/tptp/.

The Visual Studio Profiling Tool, http://msdn.microsoft.com/en-us/library/bb385770.aspx.