New Approach for the Dynamic Enforcement of Web Services Security

Azzam Mourad and Sara Ayoubi
Department of Computer Science and Mathematics,
Lebanese American University,
Beirut, Lebanon
Email:azzam.mourad@lau.edu.lb

Hamdi Yahyaoui
Department of Computer Science,
Kuwait University,
Kuwait

Hadi Otrok
Department of Computer Engineering,
Khalifa University of Science,
Technology & Research,
Abu Dhabi, UAE

***Abstract*—We propose in this paper a new approach for the dynamic enforcement of Web services security, which is based on a synergy between Aspect-Oriented Programming (AOP) and composition of Web services. Security policies are specified as aspects. The elaborated aspects are then weaved (integrated) in the Business Process Execution Language (BPEL) process at runtime. The main contributions of our approach are threefold: (1) separating the business and security concerns of composite Web services, and hence developing them separately (2) allowing the modification of the Web service composition at run time and (3) providing modularity for modeling cross-cutting concerns between Web services. We demonstrate the feasibility of our approach by developing a Flight System (FS) that is composed of several Web services. First, a RBAC (Role Based Access Control) model for the flight system, which we called RBAC-FS, is elaborated. Afterwards, the Web services that implement the security features are developed. Finally, the BPEL aspects that integrate the security functionalities dynamically into the BPEL process are created. The devised aspects realize the elaborated RBAC-FS model and provide authentication and access control features to the flight system. Case studies and experimental results are also presented to defend our propositions.**

I. Introduction

Security is becoming a hot topic in academia and industry. In this context, customers are expecting security to be delivered out of the box, even on programs that were not designed with security in mind. The challenge is even greater when legacy systems must be adapted to network/Web environments, which is the case for Web services.

Web services technology has been successful in making business applications available through the internet to a large number of users. It does not just widen the broad of applications accessibility but it is also a catalysis for collaboration between several distributed applications through the composition concept. Nevertheless, the successful deployment of this technology cannot hide the security breaches and threats [1] that a Web services can be exposed to. Enforcement of Web services security is one of the most important duties, which the research community has to perform. This means the design and development of concepts, processes and tools that help in making Web services protected from malicious users. Instances of these security features can be the enforcement of user authentication, access control and data confidentiality in Web services.

Several standard languages have been proposed to enforce Web services security. The Security Assertion Markup Language (SAML) [2], WS-Security [3] and WS-XACML [4] are the most successful ones. The main problem with language-based strategies is that the main focus is on embedding the security features in the design/code of the Web services, i.e, integrating them statically. However, many security features require run-time verification of the security policies, which may often be modified and updated. This means that when the security policies and/or the verification strategy change, the developer has to go back to the design/code of the Web services and update them accordingly. This problem is raised more when several Web services are composed together in a BPEL (Business Process Execution Language) process to form a more complex system. With the use of the current BPEL, there is a lack of modularity for modeling cross-cutting concerns and inadequate support for changing the composition at run time. Any changes in the environment and addition or removal of partner Web services requires static and dynamic adaptation. In other words, if a BPEL runtime process change is required, we have to stop the running process, change the needed Web service(s), modify the composition, and then restart. Such a mechanism is cumbersome, error-prone and tedious.

We propose, in this paper, an aspect-oriented approach for the dynamic enforcement of Web services security. Our proposition is based on a synergy between Aspect-Oriented Programming (AOP) and composition of Web services. AOP allows to specify the security concerns in separate components called aspects. These aspects are then weaved (integrated) in the BPEL process at runtime. The main contributions of our approach are threefold:

1) Separating the business and security concerns of composite Web services, and hence developing them separately.
2) Allowing the modification of the Web services compo-

This work is supported in full by the Lebanese American University under LAU grant # URC-F-2010-14

sition at run time to integrate, remove and/or update security mechanisms.

3) Providing modularity for modeling security cross-cutting concerns between Web services.

To demonstrate the feasibility of our propositions, we developed a Flight System (FS) that is composed of several Web services. A RBAC (Role Based Access Control) model for the flight system, which we called RBAC-FS, is elaborated. Afterwards, the Web services that implement the security features are developed. Then, the BPEL aspects that integrate the security functionalities dynamically into the BPEL process are created. The devised aspects realize the elaborated RBAC-FS model and provide authentication and access control features to the flight system. Case studies and experimental results are also presented to defend our propositions.

The rest of the paper is organized as follows. In Section II, we discuss the related work. Section III is devoted to the description of the FS. Section 3 is dedicated to the illustration of the proposed approach. In Section V, we present the RBAC-FS model of the flight system. In Section VI, we illustrate the implementation of our propositions in a case study. Finally, we give some concluding remarks in Section VII.

II. Related Work

Web services security is one of the topics that attracted the attention of the research community. From the definition of standards to the publication of research papers, the goal is to provide policies and mechanisms for enforcing Web services security. In this context, several standards such as Security Assertion Markup language (SAML) [2], WS-Security [3] and WS-XACML [4] were proposed.

SAML [2] is a specification language that is proposed by OASIS. Based on XML, it defines how to specify security credentials, which are represented as assertions. SAML can be used to manage secure sessions between organizations and can leverage several mechanisms such as basic password authentication, SSL and X. 509 certificates, etc. A security token is delivered to the requester after successful authentication. This security token allows granting certain permissions to the requester.

WS-Security [3] is a standard that is proposed by IBM, Microsoft, and Verisign. WS-Security is a means for using XML to encrypt and digitally sign SOAP messages. Another feature of WS-Security is that it allows exchanging security tokens for authentication and authorization of SOAP messages.

The Web Services eXtensible Access Control Markup Language (WS-XACML) [4] is proposed by OASIS as XML-based language to specify and exchange access control policies. WS-XACML is designed to define authorization policies for principals that are specified using XML.

Bhatti et al. [5] proposed X-RBAC: 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.

Damiani et al. [6] proposed a design of a Web services architecture for enforcing access control policies. They also provide an example of implementation based on the WS-Policy [7], [8] 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 realizes 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.

Bertino et al. [9] proposed a RBAC-WS-BPEL framework for defining authorization policies and constraints for WS-BPEL business processes. WS-BPEL is a language for composing Web services. To specify authorization policies, the authors used XACML. They introduce the Business Process Constraint Language (BPCL), which can be used to specify authorization constraints. The users are associated with roles as done in Role Based Access Control (RBAC) models.

The main problem with the proposed solutions for the enforcement of Web services security is the static embedding of the security features in the design/code of the Web services. In fact, many security features require run-time verification of the security policies, which may often be modified and updated. This means that when the security policies and/or the verification strategy change, the developer has to go back to the design/code of the Web services and update them accordingly. Such a mechanism is cumbersome, error-prone and tedious. Our approach relies on the dynamic injection of AOP aspects into BEPL processes. This allows to easily update the security measures when needed, without affecting the business logic of the BPEL process.

Regarding the use of AOP for security, the following is a brief overview of the available contributions. Cigital labs proposed an AOP language called CSAW [10], which is a small superset of the C programming language dedicated to improve the security of C programs. De Win, in his Ph.D. thesis [11], discussed an aspect-oriented approach that allowed the integration of security aspects within applications. It is based on AOSD concepts to specify the behavior code to be merged in the application and the location where this code should be injected. In [12], Ron Bodkin surveyed the security requirements for enterprise applications and described examples of security crosscutting concerns, with a focus on authentication and authorization. Another contribution in AOP security is the Java Security Aspect Library (JSAL), in which

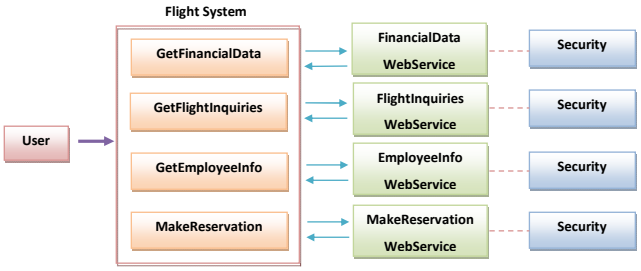

Figure 1. FS Architecture

Huang et al. [13] introduced and implemented, in AspectJ, a reusable and generic aspect library that provides security functions. In [14], Shlowikowski and Ziekinski discussed some security solutions based on J2EE and JBoss application server, Java Authentication and Authorization service API (JAAS) and Resource Access Decision Facility (RAD). These solutions are implemented in AspectJ. These approaches are useful to explore the feasibility of using AOP in software security. Hence, we can benefit from their achievements in building our security model.

III. BPEL Process Architecture of the Flight-System Web Services

In this section, we describe the architecture of the flight system Web services and their corresponding BPEL process.

A. Flight System Overview

Figure 1 explores the interactions between the user, the BPEL process and the Web services of the flight system. As depicted in the Figure, the security features are deployed on the Web services side (i.e, not in the BPEL process). This clearly shows that any changes in these security features need a modification in the corresponding Web service.

Our Flight System is mainly composed of four separate Web services, a BPEL process and a graphical user interface that allows the user to request information about the Flight agency staff, get available flights and its monthly revenues, and make a reservation. The system available services are shown in the system main page. 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 plus the name of the airline, and the available seats and tickets price. The Employee Information Service allows the user to view information about the flight system staff by entering their ID number. This information includes the employee's full name, phone number, email address, post and his office number. Finally, the Make reservation Service enables the user to reserve a seat on a certain flight. All users who can access the system have records in the database. In other words, each user has an ID and a Password stored in the database, in addition to his/her personal information. Each time a user wishes to access one of the flight system services, both the authentication and access control services are invoked to ensure that he/she is not only a valid user, but he/she also has the permission to view the requested information.

B. BPEL Process Architecture

Figure 2 illustrates a part of the the architecture of the BPEL process of our flight system. For space restriction, the figure only explores one service invoked from the BPEL process. We called this Web service *AnyFSWebService*. This Web service may or may not run with security on the side. The process is invoked when the user requests one of the services offered by the Web service. Then, the process assigns the input to the FlightService Request message. Afterwards, the latter calls the appropriate operation of the requested service and returns the needed info. Finally, the Web Service response message is assigned to the BPEL process output variable and then forwarded to the client.

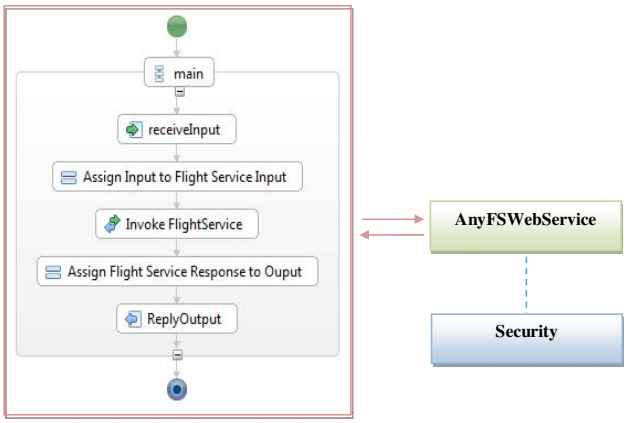

Figure 2. FS BPEL Process

IV. Approach Description

Aspect Oriented Programming (AOP) [15], [16], [17], [18], [19] is one of the most prominent paradigms that have been devised for integrating non-functional requirements (e.g. security) into software. 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 [19]).

Security is one of the software aspects that are important to deal with. Generally, developers do not separate between security and business logic codes. This means that any change in the security strategy has to be done on the application code, which can have impact on the business logic. AOP solves this issue by embedding security in aspects. Aspects allow to

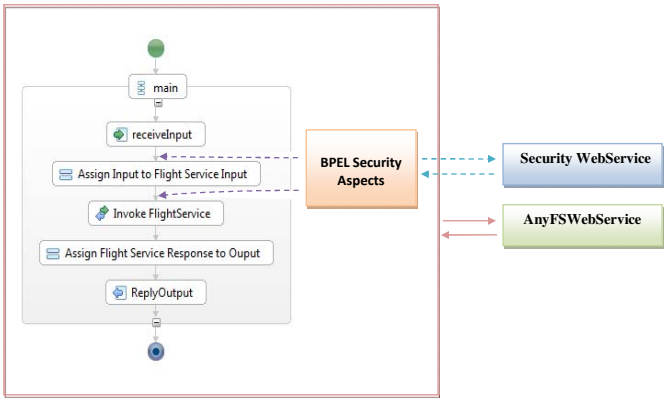

Figure 3. Approach Schema

precisely and selectively define and integrate security objects, methods and events within application, which make them interesting solutions for many security issues. Many contributions [10], [11], [12], [13], in addition to our experiments [20], have proven the usefulness of AOP for integrating security features into software.

In this context, we present in this section an aspect-oriented approach for the dynamic enforcement of Web services security. Our proposition is based on the use of AOP in the BPEL process of the composed Web services. It allows to specify the security concerns into separate components called aspects. These aspects are then weaved in the BPEL process at runtime.

The proposed approach is depicted in Figure 3. It illustrates the BPEL process where the invoke of the security features is embedded in BPEL aspects. By comparing Figure 2 with Figure 3, we can see that the security features are no longer part of the Web services, and hence any modification in the security policies can be reflected in the BPEL aspects that are weaved dynamically in the system BPEL process. The BPEL aspects may contain direct security verification to be integrated at some identified join points in the BPEL process, or it may contain an invoke to external Web service(s) that handle the security verification. Examples of BPEL security aspects are presented in Section VI.

Access control is an instance of security features that require checking at run-time if a user has the right to perform a specific operation. Embedding these features in BPEL aspects allows run-time checking and dynamic update in case the verification strategy changes. Moreover, Web services security is generally represented as a specification of policy rules that are written in a specific language like WS-XACML, WS-Policiy, etc. Linking these rules to security aspects will give more power to the dynamic verifications that can be done. In fact, through pointcuts and join points, security rules can be injected in certain flow points of the BPEL process of Web service code. The specification of these pointcuts can be updated at any time without the need to modify the business logic and the Web services code.

To demonstrate the feasibility of our propositions, we first developed the Flight System (FS) described in Section III. The flight system is supported by a RBAC model, which we detail in the following section.

V. RBAC-FS: An Authorization Model for a Flight-System Web Services

In this section, we focus on describing and defining the RBAC-FS model of our flight system. This model inherits all the components of traditional RBAC models: users, roles, permissions, role hierarchies, user-role assignment, and role-permission assignment relations. Users are assigned to roles and roles are assigned to permissions. An RBAC permission represents the ability to access a certain system service. A user is permitted to execute a service activity if he/she is assigned to a role that has the permission to perform that activity. RBAC roles are structured in a hierarchy. More details about describing RBAC model for Web services is presented in [9].

A: is the identifier of an activity (e.g., view flight inquiries).
R: is the set of roles (e.g., Leader, Manager, Staff).
U: is the set of potential users.
P: is the set of permissions, (e.g., execution of an activity).

Definition 1*: RBAC-FS Permission:* Let FS be our System. An *RBAC-FS* permission is a tuple (A_i, *Action*), where A_i is the identifier of an activity in FS and *Action* identifies the type of the action that can be performed on activity A_i. For example, the tuple (*Make reservation*, *execute*) allows the authorized user to *execute* the "Make Reservation" service provided by the Flight System.

Definition 2*: RBAC-FS Role:* An RBAC-FS role r is a set of attribute conditions r= $\{ac_i \mid ac_i$=$AttrName_i$ op $AttrValue_i\}$, where $AttrName_i$ identifies a user attribute name, op is a comparison or a set of operators, and $AttrValue_i$ is a value, a set, or a range of attribute values. Note that the roles r and r' might be recognized by the same set of attribute names. However, it is a must that at least one of the values that the attributes of r and r' assume must be different. A user can be assigned to only one role while two users identified by the same attributes with the same values are assigned to the same role since we assume that a set of attribute conditions uniquely identifies a role.

Definition 3*: RBAC-FS Role Hierarchy:* Let R be a partially ordered set of roles. A role hierarchy defined over R is the graph of the partial-order relation between the roles in R. If r, $r_0 \in R$ and $r < r_0$, then we say r_0 dominates r. For instance, our flight System consists of four different roles. The highest role is the $leader$ which has access to all the available services. The $supervisor$ and $manager$ are two different roles with the same permissions. They have less access rights than the leader but more access rights than the staff members. The $staff$ role has the least access rights. Figure 4 illustrates the role hierarchy of the flight management system.

Definition 4*: RBAC-FS User-Role Assignment Relation:* Let U be the set of all potential users and R be a partial ordered set of roles. The *RBAC-FS* user assignment relation

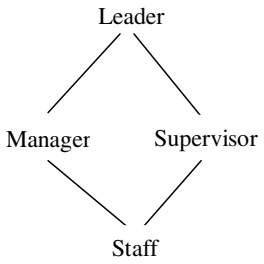

Figure 4. RBAC-FS Role Hierarchy

Table I
RBAC-FS ROLE HIERARCHY

R	
Leader characters,	{Employment= Leader, ID= a string of at most 9 Password= a string of at most 9 characters}
Manager	{Employment= Manager, ID= integer of 9 digits, Password= a string of at most 9 characters}
Supervisor	{Employment= Supervisor, ID= integer of 9 digits, Password= a string of at most 9 characters}
Staff	{Employment= Staff, ID= integer of 9 digits, Password= a string of at most 9 characters}

is the set of attributes UA= {(u,r) $\in$ U x R | $\forall$ ac_i = $AttrName_i$ op $AttrValue_i$ $\in$ r, $\exists$ $attr_j$ $\in$ CredSet(u)[1] | $attr_j$= $AttrName_i$ $\bigwedge$ ac_i is evaluated to "true" according to the value of $attr_j$}. As for the flight system, the set of roles are R={*Leader*, *Manager*, *Supervisor* and *Staff*}. Assigning users to roles results in a set of attributes that defines the RBAC-Flight user assignment relation. For example, in our Flight System the set of attribute conditions for the Leader role is r= {Type= "Leader", ID= a string of 9 characters, Password= a string of at most 9 characters}; thus a credential set of the user u={Type= "Leader", ID="Nadia", Password= "Moati"} will be evaluated as "true" and u is assigned to $Leader$.

Definition 5*: RBAC-FS User-Permission Assignment:* Let P be the set of permissions of the activity A_i supported by the system, and RP be the set of permission/role assignments. Thus, the RBAC-FS user-permission assignment relation is the set of attributes UP= {(u,p) $\in$ U x P | $\exists$ (u,r) $\in$ UA | (r,p) $\in$ RP}. For instance, a permission for viewing the financial data only is assigned to $Leader$ by the RP relation. Thus, a user u can view financial data only if he is assigned first to $Leader$.

VI. CASE STUDY: DYNAMIC ENFORCEMENT OF THE RBAC-FS MODEL IN THE FS

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 dynamic enforcement of authentication and access control features in the flight system.

A. BPEL Aspects Realizing the RBAC-FS Model

In what follows, we describe the authentication and access control aspects realizing the aforementioned RBAC-FS model. Please note that the syntax and constructs of the BPEL aspects are similar to AspectJ and SHL [21] constructs. The development of these constructs in the the BPEL compiler is still in progress.

User Authentication BPEL Aspect: Listing 1 illustrates the aspect to authenticate the user and assign him role and permission(s). Role and permission descriptions are presented in the aforementioned RBAC-FS model (see description in section V). We define the pointcut P_1 as the <bpel:receive...> construct that receives the request from the user. We insert the BPEL authentication verification after the execution of the input request. The Authentication verification code consists first of assigning the client's login info to the Web service request message. Then, the Web service is invoked. This latter calls the UserAuthentication operation that loops through the database and returns one of four possible indexes: 1 if the client's username is invalid, 2 if the client has entered correct username and password, 3 if he entered valid username but incorrect password or 4 if he has no more trials and is not allowed to login anymore. Our system allows the user three trials to enter the correct credentials. After the invoke, an If condition is integrated to check the result returned by the Web service. If the user is not authenticated, the Web service is invoked again to get the appropriate error string for the returned index. The returned error string is then assigned to the BPEL process output variable and forwarded to the client. On the other hand, if the user is authenticated, the BPEL process continues its execution.

Listing 1. Aspect for Authentication

```
Aspect Authentication
{
Pointcut P1: <bpel:receive name="receiveInput" partnerLink
     ="client" .../>;

After P1
{
<!-Initialize Authentication Request Message-->
<bpel:assign validate="no" name="Assign Input to
     Authentication message">
   <bpel:copy>
   <bpel:from>
<bpel:literal xml:space="preserve"><impl:userAuthentication
      xmlns:impl="http://t320.open.ac.uk" xmlns:xsi="http://
     www.w3.org/2001/XMLSchema-instance">
<impl:Username></impl:Username>
    <impl:Password></impl:Password>
    <impl:index></impl:index>
</impl:userAuthentication>
</bpel:literal>
   </bpel:from>
   <bpel:to variable="AuthenticationRequest" part="
        parameters">
   </bpel:to>
  </bpel:copy>
   <-- Copying the input to Authentication Request Message
       -->
 <bpel:copy>
   <bpel:from part="payload" variable="input"></bpel:from>
   <bpel:to part="parameters" variable="
        AuthenticationRequest">
   </bpel:to>
 </bpel:copy>
</bpel:assign>

<!-Invoking the Authentication WS-->
<bpel:invoke name="Invoke Authentication" partnerLink="
     Authentication" operation="userAuthentication" portType
     ="ns:Authentication" inputVariable="
     AuthenticationRequest" outputVariable="
     AuthenticationResponse">
```

```
</bpel:invoke>

<!-- if User is InValid, Reply with error message and Exit
     Process-->
<bpel:if name="If Invalid User">      <bpel:condition><![
     CDATA[($AuthenticationResponse.parameters/ns:
     userAuthenticationReturn!=2)]]>
</bpel:condition>
<bpel:sequence>

<!-Initialize the ErrorString message-->
<bpel:assign validate="no" name="Assign
     AuthenticationResponse to GetErrorStr">
<bpel:copy>
   <bpel:from>
<bpel:literal xml:space="preserve"><impl:getMessageStr
     xmlns:impl="http://t320.open.ac.uk"    xmlns:xsi="http
     ://www.w3.org/2001/XMLSchema-instance">
<impl:messageIndex></impl:messageIndex>
</impl:getMessageStr>
</bpel:literal>
</bpel:from>
 <bpel:to variable="GetErrorStr" part="parameters">
</bpel:to>
</bpel:copy>

<!-Copy the AuthenticationResponse to the ErrorStr -->
<bpel:copy>
  <bpel:from part="parameters" variable="
       AuthenticationResponse">
  <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:
       sublang:xpath1.0">
  <![CDATA[ns:userAuthenticationReturn]]></bpel:query>
  </bpel:from>
  <bpel:to part="parameters" variable=" GetErrorStr">
  <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:
       sublang:xpath1.0">
  <![CDATA[ns:me  ssageIndex]]></bpel:query>
  </bpel:to>
 </bpel:copy>
</bpel:assign>

<!-Invoke GetErrorStr WS-->
 <bpel:invoke name="Invoke GetErrorStr" partnerLink="
      Authentication" operation="getMessageStr" portType="ns
      :Authentication" inputVariable="GetErrorStr"
      outputVariable="ReturnErrorStr">
</bpel:invoke>

<!-Initialize Bpel Output Message-->
<bpel:assign validate="no" name="Assign GetErrorStr To
     Output">
<bpel:copy>
   <bpel:from>
<bpel:literal xml:space="preserve"><tns:
     AnyWSProcessResponse xmlns:tns="http://test" xmlns:xsi
     ="http://www.w3.org/2001/XMLSchema-instance">
    <tns:result></tns:result>
</tns:AnyWSProcessResponse>
</bpel:literal>
   </bpel:from>
  <bpel:to variable="output" part="payload"></bpel:to>
</bpel:copy>

<!-Copy Authentication ErrorStr to Output variable-->
<bpel:copy>
  <bpel:from part="parameters" variable="ReturnErrorStr">
  <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:
       sublang:xpath1.0">
  <![CDATA[ns:getMessageStrReturn]]></bpel:query>
  </bpel:from>
  <bpel:to part="payload" variable="output"><bpel:query
            queryLanguage="urn:oasis:names:tc:wsbpel:2.0:
       sublang:xpath1.0">
  <![CDATA[tns:result]]></bpel:query>
  </bpel:to>
 </bpel:copy>
</bpel:assign>

<!-Return "User Not Authorized" to the Client"-->
<bpel:reply name=" Return ErrorString" partnerLink="client"
      operation="process" portType="tns:AnyWSProcess"
     variable="output">
</bpel:reply>
</bpel:sequence>
</bpel:if>
}
```

Access Control BPEL Aspect for AnyFSWebService: Listing 2 illustrates the aspect to authorize the user access to the flight system services. As described in the aforementioned RBAC-FS model (please see description in section V), each user is assigned a role together with its corresponding permission(s). We define the pointcut P_2 as the <bpel:assign ...> BPEL construct responsible of assigning the input to the requested Web service and consequently invoking it. We insert the BPEL access control verification before the execution of the $assign$ construct. The access control verification BPEL code consists of assigning the client's login info to the Web service request message, then invoking the access control Web Service. The Web service will first call the "getPermission" operation that will check the user's permission based on its username and password. Once the user's permission is identified, the "getPermission" function will call the "checkAccess" operation that will check if the user's permission allows him to execute the requested service (1: GetEmployeeInfo, 2: GetFlighInquiries, 3: GetFinancialData or 4: MakeReservation). The Web service returns one of two possible responses: "Granted Access" or "Denied Access". The Process enters an IF condition to check the Web service response. If the user was not granted access, the Web service response message will be assigned to the BPEL process output variable and forwarded to the client. Otherwise, the BPEL process proceeds with the execution of the requested Web service.

Listing 2. Aspect for Service Access Control

```
Aspect Service-Access-Control
{
Pointcut P2: <bpel:assign validate="no" name="Assign Input
     to WS message">;

Before P2
{
<!-Initialize AccessControl Request Message-->
<bpel:assign validate="no" name="Assign Input To
     AccessControl message">
 <bpel:copy>
  <bpel:from>
<bpel:literal xml:space="preserve"><impl:getPermission
     xmlns:impl="http://t320.open.ac.uk" xmlns:xsi="http://
     www.w3.org/2001/XMLSchema-instance">
   <impl:Username></impl:Username>
 <impl:Password></impl:Password>
    <impl:index></impl:index>
</impl:getPermission>
</bpel:literal>
 </bpel:from>
<bpel:to variable="AccessControlRequest" part="parameters
     "></bpel:to>
</bpel:copy>

<!-Copy input to AccessControl Request Message-->
<bpel:copy>
  <bpel:from part="payload" variable="input"></bpel:from>
  <bpel:to part="parameters" variable="AccessControlRequest
       "></bpel:to>
</bpel:copy>
</bpel:assign>

<!-Invoke AccessControl WS-->
<bpel:invoke name="Invoke AccessControl" partnerLink="
     AccessControl" operation="getPermission" portType="ns:
```

```
    AccessControl" inputVariable="AccessControlRequest"
    outputVariable="AccessControlResponse">
</bpel:invoke>

<!-Check if Access is denied-->
<bpel:if name="If Denied Access">
<bpel:condition><![CDATA[(starts-with("Denied",
    $AccessControlResponse.parameters/ns:
    getPermissionReturn))]]>
</bpel:condition>
 <bpel:sequence>

<!-Initialize Bpel Output Message-->
 <bpel:assign validate="no" name="Assign AccessControl
      Response to Output">
  <bpel:copy>
    <bpel:from>
<bpel:literal xml:space="preserve"><tns:
    AnyWSProcessResponse xmlns:tns="http://test" xmlns:xsi
    ="http://www.w3.org/2001/XMLSchema-instance">
<tns:result></tns:result>
</tns:AnyWSProcessResponse>
</bpel:literal>
    </bpel:from>
<bpel:to variable="output" part="payload"></bpel:to>
</bpel:copy>

<!-Copy AccessControl response message to Output Variable
    -->
<bpel:copy>
 <bpel:from part="parameters" variable="
      AccessControlResponse">
 <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:
      sublang:xpath1.0">
<![CDATA[ns:getPermissionReturn]]></bpel:query>
 </bpel:from>
<bpel:to part="payload" variable="output">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:
    sublang:xpath1.0">
<![CDATA[tns:result]]></bpel:query>
</bpel:to>
</bpel:copy>
</bpel:assign>

<!-Return "Access Denied" to the Client"-->
<bpel:reply name="ReplyAcessDenied" partnerLink="client"
    operation="process" portType="tns:AnyWSProcess"
    variable="output">
</bpel:reply>
</bpel:sequence>
</bpel:if>
}
```

B. Discussion and Experimental Results

Weaving the security aspects in Listing 1 and Listing 2 with the BPEL process of the flight system presented in Figure 2 produces the secure BPEL process illustrated in Figure 5. The resulted BPEL process provides dynamic authentication and access control features for the flight system. The BPEL process begins by receiving the client's login info and the index of the service he requested. After receiving the input, the authentication Web service gets invoked to ensure that the user has valid username and password. If he is not authenticated, the process enters the conditional branch and returns an error string to inform the client of the authentication failure. On the other hand, if the user is authenticated, the process will continue to assign the client's input to the access control request message. Then, the access control Web service gets invoked to get the client's permission level and check whether the client has the right to see the requested service. If the access is denied, the process assigns the access control response message to the BPEL output variable and forward it to the client. Otherwise, the process will proceed to invoke the flight system Web service AnyFSWebservice and return the client's requested service.

Verifying the successful integration of the RBAC-FS security features in the original BPEL code of the flight 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.

VII. Conclusion

We presented in this paper a new approach for the dynamic enforcement of Web services security. Our proposition is based on a synergy between AOP and composition of Web services. It allows the separation between business and security concerns of composite Web services, and hence developing them separately. It also allows the modification of the Web services composition at run time and provides modularity for modeling cross-cutting concerns between Web services. The experiments resulting from developing the RBAC-FS model and their BPEL aspects, and then deploying them dynamically in the BPEL process of the flight system, demonstrate the feasibility and appropriateness of our propositions. They also illustrate the successful dynamic integration and modification of authentication and access control features in the flight system.

References

[1] K. N. Computing, "XML and Web Services: Message Processing Vulnerabilities," http://www.webservicessummit.com/Articles/MessagingThreats.htm.

[2] B. Lockhart and al., "OASIS Security Services TC (SAML)," http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

[3] B. Atkinson and al., "Web services security (WS-Security)," http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

[4] T. Moses, "OASIS eXtensible Access Control Markup Language (XACML), OASIS Standard 2.0," http://www.oasis-open.org/committees/xacml/.

[5] R. Bhatti, J. Joshi, E. Bertino, and A. Ghafoor, "Access Control in Dynamic XML-Based Web-Services with X-RBAC," in *Proceedings of the International Conference on Web Services (ICWS'03)*, 2003, pp. 243–249.

[6] C. A. Ardagna, E. Damiani, S. D. C. di Vimercati, and P. Samarati, "A Web Service Architecture for Enforcing Access Control Policies," *Electronic Notes Theoretical Computer Science*, vol. 142, pp. 47–62, 2006.

[7] J. Schlimmer, "Web Services Policy Framework (WS-Policy)," 2004, http://www-128.ibm.com/developerworks/webservices/library/specification/ws-polfram/.

[8] P. Nolan, "Understand WS-Policy processing," IBM Corporation, Tech. Rep., 2004.

[9] F. Paci, E. Bertino, and J. Crampton, "An Access-Control Framework for WS-BPEL," *International Journal of Web Services Research*, vol. 5, no. 3, pp. 20–43, 2008.

[10] V. Shah, "An aspect-oriented security assurance solution," Cigital Labs, Tech. Rep. AFRL-IF-RS-TR-2003-254, 2003.

[11] B. DeWin, "Engineering application level security through aspect oriented software development," Ph.D. dissertation, Katholieke Universiteit Leuven, 2004.

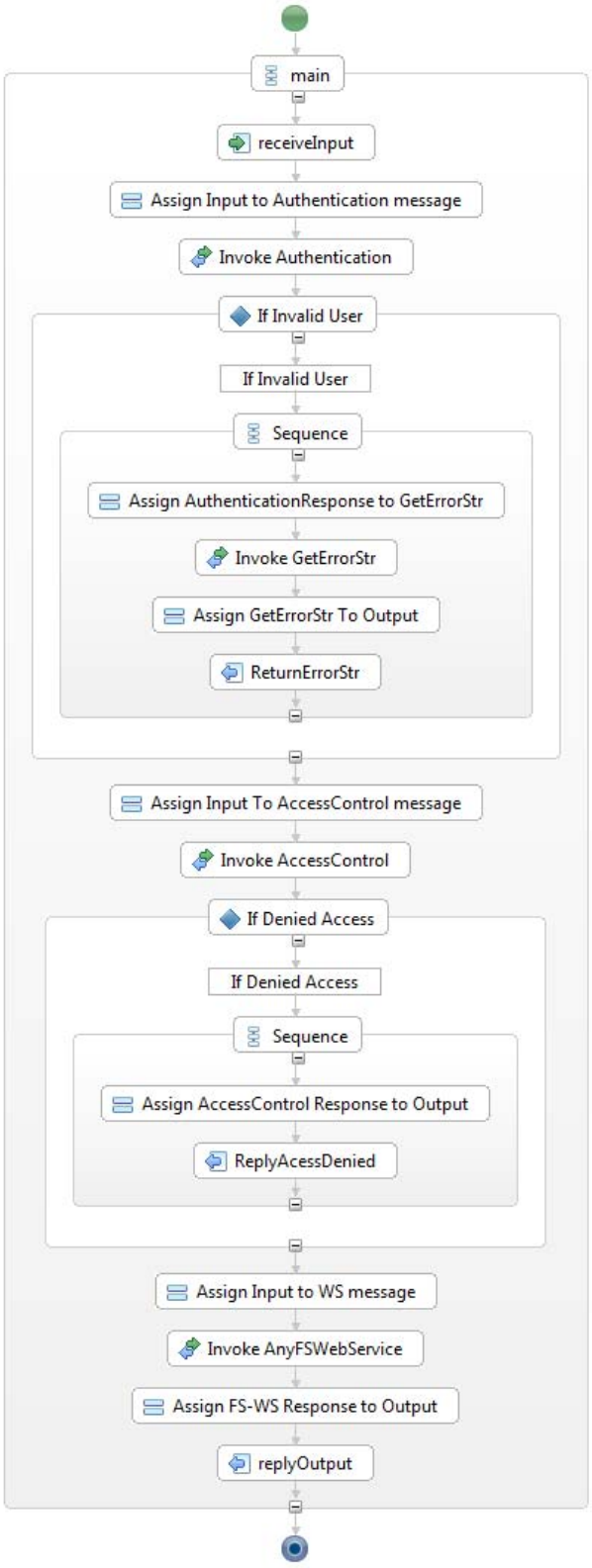

Figure 5. FS Secure BPEL Process

[12] R. Bodkin, "Enterprise security aspects," in *Proceedings of the AOSD 04 Workshop on AOSD Technology for Application-level Security (AOSD'04:AOSDSEC)*, 2004.

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

[14] P. Slowikowski and K. Zielinski, "Comparison study of aspect-oriented and container managed security," in *Proceedings of the ECCOP workshop on Analysis of Aspect-Oriented Software*, 2003.

[15] J. Pavlich-Mariscal, L. Michel, and S. Demurjian, "Enhancing UML to Model Custom Security Aspects," in *Proceedings of the 11th International Workshop on Aspect-Oriented Modeling (AOM@AOSD'07)*, 2007.

[16] L. Fuentes and P. Sanchez, "Elaborating UML 2.0 Profiles for AO Design," in *Proceedings of the International Workshop on Aspect-Oriented Modeling*, 2006.

[17] J. Evermann, "A Meta-Level Specification and Profile for AspectJ in UML," *Journal of Object Technology*, vol. 6, no. 7, pp. 27–49, 2007.

[18] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin, "Aspect-oriented programming," in *Proceedings European Conference on Object-Oriented Programming*, M. Akşit and S. Matsuoka, Eds. Berlin, Heidelberg, and New York: Springer-Verlag, 1997, vol. 1241, pp. 220–242. [Online]. Available: citeseer.ist.psu.edu/kiczales97aspectoriented.html

[19] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold, "An Overview of AspectJ," in *Proceedings of the 15th European Conference on Object-Oriented Programming (ECOOP'01)*. London, UK: Springer-Verlag, 2001, pp. 327–353.

[20] A. Mourad, M.-A. Laverdière, and M. Debbabi, "Towards an aspect oriented approach for the security hardening of code," *Computers & Security*, vol. 27, no. 3-4, pp. 101–114, 2008.

[21] ——, "A high-level aspect-oriented based framework for software security hardening," *Information Security Journal: A Global Perspective*, vol. 17, no. 2, pp. 56–74, 2008.