AC-Composer: An Access Control System for Web Services Orchestration

Harry Wang  
*University of Arizona*

Leon Zhao  
*University of Arizona*

Jay Nunamaker, Jr.  
*University of Arizona*

Mohan Tanniru  
*University of Arizona*

Follow this and additional works at: [http://aisel.aisnet.org/amcis2004](http://aisel.aisnet.org/amcis2004)

Recommended Citation
[http://aisel.aisnet.org/amcis2004/514](http://aisel.aisnet.org/amcis2004/514)
AC-Composer: An Access Control System for Web Services Orchestration

Harry J. Wang
Department of Management Information Systems
University of Arizona
jiannan@eller.arizona.edu

J. Leon Zhao
Department of Management Information Systems
University of Arizona
lzhao@eller.arizona.edu

Jay Nunamaker, Jr.
Department of Management Information Systems
University of Arizona
nunamaker@eller.arizona.edu

Mohan Tanniru
Department of Management Information Systems
University of Arizona
mtanniru@eller.arizona.edu

ABSTRACT

Web services orchestration is a new research area that focuses on the development of methodologies and techniques for composing web services to support business operations and processes. Despite the promises, the absence of adequate security is a major obstacle in the adoption of web services orchestration. Currently, existing web services orchestration standards do not offer direct support for security, resulting in a gap between the commercial needs and these standards. In this paper, we propose an access control system named Access Control Composer for web services orchestration by leveraging several existing authorization mechanisms. We show how the security gap can be mended by integrating access control techniques into web services orchestration standards. We also present a distributed, loosely-coupled system architecture that leverages several emerging standards, such as BPEL4WS and XACML.

Keywords:
Web Services Orchestration, Access Control, Workflow Security, BPEL4WS, XACML

INTRODUCTION

In order to offer easy integration with a wide range of business-partner applications, many companies are investing considerable resources to expose their business operations with web services technology (Geer, 2003). A higher business value can be achieved if the individual web services in a supply chain can be connected to form composite business processes. As an open, standards-based approach for web services connectivity, web services orchestration is receiving increasing attention. This standard describes how web services can interact with each other at the message level and defines the business logic and execution order of the web services interactions, which may span applications and/or organizations (Peltz, 2003). It also assumes that one of the business parties centrally controls the process, which makes the process management easier. With the support from major industrial vendors, such as IBM, Microsoft and BEA, web services orchestration is becoming the de facto web services composition standard.

Despite the promise, the absence of security is proving to be a major obstacle in convincing companies to participate in web services orchestration. Although there have been lots of research efforts in the development of web services security, and several security specifications have been proposed, such as WS-Security, Extensible Access Control Markup Language (XACML), and Security Assertion Markup Language (SAML) (Naedele, 2003), current orchestration standard languages, such as Business Process Execution Language for Web Services (BPEL4WS), do not offer direct support for those standards (Andrews et al., 2003), resulting in a gap between the commercial needs and these standards.

In this paper, we focus on the authorization aspect of this security concern. Access control policies are often specified for web services exposed to the public network to block illegal access requests (Kraft, 2002). In web services orchestration, web services often reside in different companies with disparate security systems. The complexity of the composite processes and the heterogeneity of security systems make access control in web services orchestration an intriguing and challenging problem.
Based on a review of several access control models and previous work in related domains, such as workflow authorization (Bertino et al., 1999, Kang et al., 2001), virtual enterprise access control (Coetzee and Eloff, 2003; Nayak et al., 2001), and web services security (Sirer and Wang, 2002, Nakamur et al., 2002), we propose an access control requirement framework for web services orchestration. The framework provides the basis for the development and evaluation of access control systems. Based on this framework, we design a web-services-enabled access control system named Access Control Composer (AC-Composer) to enable security management under the web services standard. To validate the system, we demonstrate via an example how this system provides authorization functionalities without affecting the underlying process definition and how easily it can be integrated with existing engines for web services orchestration. As a result, our work bridges the gap between two standards in access control and web services orchestration.

The remainder of the paper proceeds as follows. We first briefly review several access control models and determine the requirements of access control in web services orchestration. Then, two emerging standards, BPEL4WS and XACML, are introduced. Based on the discussion of service compositability, we present the system architecture of AC-Composer and show how it can address the problem by satisfying all the authorization requirements. Finally, we summarize our contributions and point out the future research directions.

ACCESS CONTROL REQUIREMENTS FOR WEB SERVICES ORCHESTRATION

Information systems security refers to the protection of information systems against unauthorized access to or modification of information (Joshi et al., 2001). Several models have been proposed to address the access control requirements. The traditional discretionary access control (DAC) and mandatory access control (MAC) models share a common view of security by using subject-object-permission mechanism. These models become cumbersome and hard to manage, when the number of subject and objects becomes very large (Joshi et al., 2001). Rooted in the use of grouping in UNIX and privilege groupings in database management systems, role-based access control (RBAC) model was proposed to reduce the security administrative overhead by grouping subjects into roles, which are representations of business functions in companies, such as CEO, supply manager and etc (Sandhu et al., 1996). Authorizations are then granted to roles, rather than to single users. The authorizations granted to a role are strictly related to the data objects and resources that are needed for executing the functions associated with the role. In the past decade, many extensions to the basic RBAC model have been proposed and the resulting RBAC models have become a generalized approach to access control (Ferraiolo et al., 2001).

The models discussed previously have a limited scope and are not flexible enough to specify access policies based on the content of information or the nature of tasks in business processes (Joshi et al., 2001). For example, certain authorizations have to be granted to subjects only during the execution of certain tasks and must be revoked immediately after the tasks are completed. Several authorization models have been developed to provide support for activity and task-intensive applications. Task-based Access Control (TBAC) is an active security model that is well suited for information processing activities, where users access to data and applications in order to perform certain tasks. TBAC approaches security management from the application perspective rather than from a system-centric subject-object view. In addition, the TBAC paradigm also considers the temporal constraints where access is permitted based on a just-in-time fashion for the activities or tasks in consideration.

In web services orchestrations, various web services resident in heterogeneous systems are centrally coordinated. RBAC and TBAC are initially designed for intra-organizational authorization management. In order to support these models, the authorization services must be extensible across and beyond enterprise boundaries. Based on the previous works (Kang et al., 2001, Nakamur et al., 2002, Coetzee and Eloff, 2003, Kraft, 2002), we propose the access control requirement framework for web services orchestration as described below:

- **Role mapping and resolution mechanism.** This is the fundamental prerequisite to incorporate RBAC in an inter-organizational environment. There must be a central abstract role hierarchy, where the internal roles from each individual security domain can find corresponding matches (Ferraiolo et al., 2003).

- **Security domain autonomy.** Providers and requestors of web services are independent business entities that should have complete control over their resources and have the freedom to define their specific access control policies and mechanism. (Coetzee and Eloff, 2003)

- **Fine grained and context-based access control.** According to different service properties, service provider may require different level of security. Therefore, we need to integrate several access control models aforementioned to provide fine-grained authorization that is based on user’s role and working context (Kang et al., 2001).
Separation between process definition and security infrastructure. This would allow the modification of security related information without affecting underlying process, which provides more flexibility and extensibility.

Standard-based implementation. Web services orchestration is a dynamic collaborative environment, where business relationships among service providers change constantly. In order to avoid interoperability problems among disparate information systems, the access control system must be implemented using widely accepted open standards (Coetzee and Eloff, 2003).

This framework can be further extended while security models and enabling technology are evolving. It serves as the basic guidelines for the development of access control systems. Next, we discuss the enabling technologies, on which our system is built.

WEB SERVICES ORCHESTRATION AND SECURITY STANDARDS

Web services orchestration is about providing an open, standard-based approach for connecting web services together to create higher-level business processes (Peltz, 2003). Standards have been developed to reduce the complexity required to orchestrate web services. Some early work in this standard development includes XLANG from Microsoft and Web Services Flow Language (WSFL) from IBM. Recently, Business Process Execution Language for Web Services (BPEL4WS) was jointly released by IBM, Microsoft, and BEA as a combination of the best standards for web services composition, such as XLANG and WSFL.

BPEL4WS is built on top of Web Service Definition Language (WSDL), which is used to describe the entry and exit points of BPEL processes, the external services and the data type of information exchanged between requests (Peltz, 2003). As shown in figure 1, we create a sample composite web services process, which is a simplified office supply e-procurement process. This process initializes by receiving a model number of a particular printer cartridge from a user portal. Then it distributes inventory checking requests to two suppliers: CartridgeMax and CartridgeDepot, and based on the returned inventory levels it sends out price quote requests to available suppliers. Finally, the process selects the most favorable offer and presents the best offer as the return value for the flow. This process consists of four web services resident at two different companies and a third-party service provider can centrally orchestrate the whole process.

BPEL4WS includes support for both basic and structured activities. Basic activities are in charge of receiving or replying to message requests as well as invoking external services, whereas structured activities manage the overall process flow, specifying what activities should run and in what order. For example, the sample process calls the two asynchronous inventory checking services in parallel, each of which includes two basic activities to invoke a service and to receive the results. We express these parallel activities in BPEL4WS as shown in List1. Here, \texttt{<flow>} tag provides concurrency and synchronization, while \texttt{<sequence>} tag specifies the order of sequential activities.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{composite_web_services.png}
\caption{Composite Web Services E-procurement Process}
\end{figure}
List 1. Asynchronous Inventory Checking Process in BPEL4WS

BPEL4WS also support conditional looping and dynamic branching using <switch>, <while>, and <pick> tags. The rich set of elements makes this specification expressive and complex. Therefore, we could not elaborate all the features of this standard in this paper and we refer interested readers to (Andrews et al., 2003) for more details.

Although BPEL4WS has received remarkable attention and moved into a number of product implementations, such as IBM’s BPWS4J engine and Collaxa’s orchestration platform, many companies are concerned about how security can be addressed (Peltz, 2003). WS-Security can ensure the messages have not been modified or forged while in transit or while residing at destinations, but higher-level access control policies cannot be supported (Andrews et al., 2003). In the example above, without enforcing authorization constraints to the exposed web services, the inventory checking and price quote web methods can be called by anyone from anywhere at anytime, which is not desirable for most companies. Access control policies must be clearly specified and authorization mechanism must be in place to give companies enough confidence to expose their internal business operation as web services and link them through orchestration. Although there have been many existing proprietary and application-specific access control languages, there was a pressing need for a generic, open, and standard-based authorization specification language (OASIS, 2003).

eXtensible Access Control Markup Language (XACML) is an XML based access control policy specification language. It’s developed by OASIS under the sponsorship of several major vendors, such as IBM, SUN, BEA and Entrust (OASIS, 2003). This policy language is used to describe access control requirements in any environments, form a query to ask whether or not a given action should be allowed, and interpret the result. Some of the key elements are listed below with brief explanations (OASIS, 2003).

- **Policy, PolicySet**: A Policy represents a single access control policy, expressed through a set of Rules. A PolicySet is a container of other Policies or PolicySets.
• **Target**: Target is a set of simplified conditions for the Subject, Resource and Action that must be met for a PolicySet, Policy or Rule to apply to a given request.

• **Attribute**: Attributes are characteristics of the Subject, Resource, Action, or Environment in which the access request is made. For example, a user’s name, group membership, a file they want to access, and etc.

• **Function**: Functions are used to compare the attributes in the request and the attributes in the policies to make access decisions.

• **Request**: A Request contains Attributes in each of the four categories: Subject, Resources, Action and Environment.

• **Response**: A Response consists of one or more Results. Each Result contains a Decision (Permit, Deny, NotApplicable, or Indeterminate).

In our example, CartridgeMax may define an access control policy saying that only the employee of my current partner company: Printer4Less (identified by an email name in printer4less.com namespace) can check my cartridge inventory. This policy can be expressed in XACML as shown in List 2.

```
<Policy PolicyId="CartridgeMaxInventoryCheckPolicy" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Target>
    ...
  </Target>
  <Rule RuleId="CMInventoryCheckRule" Effect="Permit">
    <Target>
      <Subjects>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
          <AttributeValue DataType="urn:oasis:names:tc:xacml:1.0:datatype:rfc822Name">printer4less.com</AttributeValue>
        </SubjectMatch>
      </Subjects>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
            <AttributeValue DataTyp...>wsdl</AttributeValue>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataTyp...>read</AttributeValue>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
</Policy>
```

**List 2. Sample Access Control Policy in XACML**

When a user tries to call the inventory checking web method (try to read the WSDL file of this web service), the model number along with user’s email address as the subject id is passed to the authorization server of CartridgeMax. The rule in this policy checks user’s email to see whether it includes printer4less.com. If yes, the model number will be passed to
the method and inventory level will be returned. If no, the user request will be denied. In this way, we can add access control constraints to web services to provide additional security.

Although we have BPEL4WS as a web services orchestration standard and XACML as an access control specification standard, to the best of our knowledge, there is no existing solution on how these two standards can complement each other to add access control features to web services orchestration. One way to address this problem is to extend the BPEL4WS specification with access control elements from XACML. This will result a bulky specification and reduce the flexibility by binding security with process design. Alternatively, we can take the concept of composability to provide the same capabilities while retaining the simplicity and independence of each specification.

**ADDING ACCESS CONTROL TO WEB SERVICES ORCHESTRATION**

The term composability is used to describe independent specifications that can be combined to provide more powerful capabilities (Ferguson et al., 2003). One of the fundamental characteristics of web services is a regular, multipart message structure, which allows new elements supporting new services to be added easily. As showed in List 3, a simple web services message can contain elements from other specifications, such as WS-Addressing and WS-Security (Ferguson et al., 2003). This allows the developer to use elements on as-needed basis. Composability has and continues to be one of the key design principles for web services.

```
<SOAP-ENV:Envelope …>
  <SOAP-ENV:Header>
    <wsa:To:http://fabrikam123.com/Traffic</wsa:To>
  </SOAP-ENV:Header>
</SOAP-ENV:Envelope>
```

**List 3. Composability of Web Services Specifications**

We can use the same approach to relate each web services in orchestration with its access control policy. For instance, List 4 shows that the policy in List 2 can be inserted into the process definition in List 1.

```
<flow>
  <!-- invoke web service1 in CartridgeMax (CM) -->
  <sequence>
    <!-- initiate the remote service -->
    <Policy PolicyId="CartridgeMaxInventoryCheckPolicy" …>
      …
    </Policy>
    <invoke name="invokeCMInventoryCheck"… />
    …
  </sequence>
</flow>
```

**List 4. BPEL4WS with XACML Policy**

Although with this approach we can know what authorization constraints have to be enforced for each service invocation in the orchestrated flow, the problem is that this combined process definition can not be understood and interpreted by existing orchestration engines. When we deployed a BPEL process with XACML policy in Collaxa Orchestration Server, the process could not be compiled because of the invalid elements. In order to solve this problem, we propose a web services access control system named Access Control Composer (AC-Composer). In general,
AC-Composer can provide authorization control for any web services. It is a comprehensive infrastructure for modeling, managing and deploying access control policies. In particular, AC-Composer can integrate with any web services orchestration server to enhance its security features. Figure 2 shows its system architecture. Next, we will walk through the deployment process of AC-Composer and elaborate every component in details. We demonstrate how Access Control Composer provides agile access control to web services orchestration by satisfying its access control requirements we summarized before.

The life cycle of AC-Composer deployment includes the following events:

1. Receive BPEL definition from the orchestration server.
   After a web services composite process is created and validated by the orchestration server, the definition file in BPEL4WS and a configuration file that includes the WSDL information of all the participating web services are sent to AC-Composer. These files do not include any security related information.

2. Visualize BPEL with access control policy GUI and receive policies specification from orchestration participants.
   Once the BPEL file is received, the Policy GUI component generates a graph representation of the composite flow and adds access control policy manipulation functions for each web services. As shown in Figure 3, the graph is presented as a webpage. Each web services owner must login to retrieve this graph. Web services and corresponding access control menus are only visible to their owner. On one hand, this approach protects company privacy. On the other hand, distributed security management is supported because the management of access control is delegated to each web services owner rather than being centrally manipulated by the orchestration server. Secure Process Producer (SPP) is in charge of the modeling and managing policies. It has a large role library and allows the web services owners to map their internal organizational role into this unified role view to support role-based access control. XACML specification needs to be extended with new elements to support task based access control and we defer this work to our future research. If the services owners already have access control policy written in XACML, they can directly import the file. Otherwise, they can use the policy builder of SPP to construct the policies from the scratch. The final product of this step is a BPEL file with XACML policies shown in List 4.

3. Convert each policy into a web service and modify the original BPEL definition by adding authorization web services invocations.
   As we discussed, the product from step two cannot be interpreted by the orchestration server. Our solution is to convert each policy into a web service and extend the original BPEL file with more authorization web services invocation, resulting in a new process, which we refer to as “secure BPEL process”. This is done by the Authorization WS Creator. Now this secure BPEL file can be compiled and interpreted by the orchestration server. For example, the policy shown in List 2 will be converted into a web service, which accepts the use’s email address as input and returns “permit” or “deny” based on the evaluation of the authorization rules. This web service will be hosted by Access Control Composer in the ACWS repository and its WSDL information will be added to the
configuration file that came from the orchestration server with the original BPEL file in step 1. Then, List 4 is revised by adding one authorization service invocation as shown in List 5.

![Web-based Policy GUI](image)

**Figure 3. Web-based Policy GUI**

```xml
<!-- invoke web service1 in CartridgeMax (CM) -->
<sequence>
  <!-- invoke the access control service -->
  <invoke name="invokeAccessControl4CMInventoryCheck"
    partnerLink="AccessControl4CartridgeMax"
    portType="services:AccessControl4InventoryCheckingService"
    operation="evaluate"
    inputVariable="SubjectID"/>
  <!-- receive the access control decision -->
  <receive name="receiveDecision4CMInventoryCheck"
    partnerLink="AccessControl4CartridgeMax"
    portType="services:AccessControl4InventoryCheckingServiceCallback"
    operation="onResult"
    variable="Decision4CMInventoryCheck"/>
  <!-- if "permit" is returned, continue to invoke CMInventoryCheck -->
  <case ...>
    ...
  </case>
  <!-- otherwise terminate the process -->
  <otherwise>
    ...
  </otherwise>
</switch>
<!-- initiate the remote service -->
<invoke name="invokeCMInventoryCheck"
  partnerLink="CartridgeMax"
  portType="services:InventoryCheckingService"
  operation="initiate"
  inputVariable="ModelID"/>
  ...
</sequence>
```

**List 5. Secured BPEL Process**

4. Send Secured BPEL definition to orchestration server.
The secure BPEL process definition file and the new configuration file that includes all the WSDL information of all the access control services are sent back the orchestration server. The server compiles and deploys the process as usual. The secure process may ask the process initiator for more security-related information. For instance, the original sample process shown in List 1 only requires a model number to trigger the process, while the secure process shown in List 5 requires one more input: the user’s email address. These additional input requirements are retrieved from the WSDL files of all the access control web services and presented to the process initiator in the WSDL file of the secure process.

---

*Proceedings of the Tenth Americas Conference on Information Systems, New York, New York, August 2004*
5. Monitor and maintain authorization web services.
   After the secure process is deployed and running, AC-Composer monitors and logs each access control request. Some log analysis tools can help to detect malicious authorization request and security holes in access control policies.

Access Control Composer represents a new set of standard-based access control mechanisms for web services orchestration. It separates the access control from the underlying process, which provides flexibility and extensibility. AC-Composer also fulfills the architectural requirements recommended by W3C, because its functionalities align with the corresponding components of the web services policy model as shown in Figure 4 (Booth et al., 2004). For instance, the POLICY, PERMISSION, ACTION components can be modeled and handled by Secure Process Producer and the AUDIT GUARD component can be mapped to the monitoring and logging features of AC-Composer. To the best of our knowledge, AC-Composer is the first standard-based access control system for web services orchestration.

CONCLUSION
Web services are rapidly emerging as the most practical approach for business application integration. While many companies are exposing business operations as web services, the real value will come when companies connect services together to streamline the whole supply chain. Web services orchestration is positioned to be the standard of web services composition and is receiving remarkable attention. But, the current web services orchestration standard does not provide direct support for security, which make companies reluctant to participate in orchestration.

In this paper, we proposed an access control system named AC-Composer to add authorization capability to web services orchestration. Our contribution is threefold: First, we proposed the access control requirement framework for web services orchestration, which forms the basic guidelines for web-services-based access control system developments. Second, we presented an approach to leverage existing standards, such as BPEL4WS and XACML, to offer authorization support in composite web services processes, while retaining the simplicity and independence of each standard. Third, we designed a light-weight system architecture for modeling and managing web services access control policies, which can be easily integrated with orchestration servers. Our future research plan includes:

- The implementation of Access Control Composer and its integration with existing web services orchestration servers, such as IBM BPWS4J and Collaxa Orchestration Platform,
- The extension of XACML to incorporate more access control models, such as role-based access control and task-based access control,
- User studies to provide quantitative evaluation of the system’s usability, efficiency and flexibility.
REFERENCES