Adding Security to BPEL Workflows of Web Services

Ola Ajaj
Department of Computer and Electrical Engineering and Computer Science
Florida Atlantic University
777 Glades Road, Boca Raton, Florida 33431-0991 USA
oajaj@fau.edu

Eduardo B. Fernandez
Department of Computer and Electrical Engineering and Computer Science
Florida Atlantic University
777 Glades Road, Boca Raton, Florida 33431-0991 USA
ed@cse.fau.edu

Abstract—BPEL (Business Process Enterprise Language) is a language for web services composition and several implementations of it exist. For BPEL to be effective, it is necessary that it provides more support for security. BPEL doesn’t present any means to specify security constraints for workflows. BPEL through its activities tries to provide specific functional aspects and any non-functional aspects are expected to be addressed by other (lower-level) specifications. We present here a way to specify security requirements in BPEL. Since BPEL describes workflows, we present its activities using UML activity diagrams, where we apply a threat enumeration approach to determine the required security mechanisms to stop these threats. Our approach goes beyond BPEL and can be applied to BPMN and other business flow languages.

I. INTRODUCTION

BPEL (Business Process Enterprise Language) is intended to coordinate interactions among the web services that participate in an application workflow. BPEL as standardized by OASIS [1], [2] is the most popular language for web service composition. This is supported by the fact that many software companies have integrated BPEL orchestration engines into their products. Its value for cloud applications is further increasing its use. We need to combine web services from different providers in order to create more advanced and collaborative web services. Two aspects are applied to represent the service workflow, one takes care of which web services participate in the interaction and in what order (control flow), and the other deals with the data been transferred between these interactions (data-flow). There are three main activities involved in an interaction: <invoke> for invoking an operation from one of the partner web services, <reply> to send a response for the requestor, and <receive> to receive a request from the requestor.

For most applications it is critical to consider non-functional aspects in these compositions, such as security, reliability and other aspects of Quality of Service (QoS). With the development of many web services standards such as WS-Security [3], WS-Policy [4], WS-Trust [5] and WS-Federation [6], more concerns are addressed to cover non-functional aspects for application specifications in order to encourage the companies to adopt web services in their business activities [7].

BPEL includes some low-level aspects and it is better to specify process structuring in a more abstract way. Two common notations used for process modeling are the BPMN and the UML. BPMN is a modeling notation aimed specifically at business process modeling [8]. UML is a general-purpose modeling notation that was originally developed for designing and specifying software-intensive systems, but which is being increasingly used in other areas, including business process modeling. We adopt UML due to its ability to allow all the application views to be captured and modeled using a single modeling language thus avoiding the need for different notations to be used within a system for process modeling and technical activities. We will use in particular Activity diagrams and Sequence diagrams to cover a part of the big picture implied in a BPEL process.

A few approaches exist to specify BPMN security but they lack some important aspects. We propose a model based on our previous work on threat enumeration. We developed an approach for security requirements elicitation based on misuse activities [9], and [10] improved it later to include threat analysis by adding two
aspects; the security attribute subverted (confidentiality, integrity, availability and accountability), and the source of threats. By applying these improvements, the approach became more effective, since several more threats can be found. In this paper we are applying this approach in order to be able to add security constrains to workflows. Our contribution stems from finding a better way to add security to workflows, which differently from other approaches, defines security specifications without the need for security specialists. We show our ideas by example, the formal basis for our approach can be found in [9] and [10].

The remainder of this paper is organized as follows. Section 2 explains and illustrates composition in BPEL. In Section 3 we introduce an example of a collaborative business process. In Section 4 we present our approach of capturing threats using activity diagrams and sequence diagrams. Section 5 explains some techniques to stop or mitigate the identified attacks. We discuss related work in Section 6. Section 7 summarizes the paper and considers future work.

II. BACKGROUND

BPEL specifies a service composition as a process, which declares the web services participating in the composition (partners), data containers (variables), and a set of activities with specific patterns of control and data flow. The building blocks of BPEL processes are activities. There are primitive activities such as <receive>, <invoke>, and <replay> and structured activities such as <sequence> and <flow>. Structured activities manage the order of execution of their enclosed activities. BPEL processes can run on any BPEL-compliant orchestration engine. The engine orchestrates the invocations of the partner web services according to the process specification.

For illustration, we present a skeleton of the BPEL process that corresponds to a travel agency request to reserve a room in a hotel. For conciseness, we omit some parts of the code.

```xml
<process name="requesttoReserveRoomProcess"/>
<partnerlinks>
  <partnerLink name="supplier"/>
  <partnerLink name="bank"/>
  <partnerLink name="HotelCompany"/>
</partnerlinks>
```

III. AN EXAMPLE FOR A COLLABORATIVE BUSINESS PROCESS

An example from the area of travel will be used to illustrate the security issues arising when we define a BPEL process that would be shared across many partners.

For illustration, we present a skeleton of the BPEL process that corresponds to a travel agency request to reserve a room in a hotel. For conciseness, we omit some parts of the code.

```xml
<variables>
  <variable name="clientrqst" messageType="orderInMT"/>
  <variable name="clientrspse" messageType="orderOutMT"/>
  <variable name="payrequest" messageType="payInMT"/>
</variables>
<sequence name="Main">
  <receive partnerlink="HotelCompany" operation="order" variable="orderqst"/>
  <invoke partnerlink="supplier" operation="putOrder" inputvariable="supplyrequest"/>
  ...
  <invoke partnerlink="bank" operation="pay" inputvariable="payrequest"/>
  ...
  <reply partnerlink="HotelCompany" operation="order" variable="clientrspse"/>
</sequence>
</process>

Listing1. RequestToReserveRoom process

Figure 1: An example of a collaborative business process

The distributed business process shown in Figure 1 is for a travel agency with a chain of offices around the world. The travel agency has a partnership with a hotel
company and a bank and communicates with them using web services. The travel agency defines a BPEL process to serve its customers. This process will reserve rooms from the hotel company and invokes the bank’s payment web service to pay for the transaction. More specifically, a BPEL process Control is executed in a hotel company. Customers send their request of reservation to the travel agency. A ReserveRoom web service of the travel agency invokes the web service offered by the control process at the hotel, which provides a list of rooms to be booked. Before placing a reservation, the travel agency expects a price offer accompanied by a commitment with respect to the reservation date.

Figure 2 shows the UML activity diagram for booking a room for a customer. We indicate “swimlanes” for Customer, Travel Agency and Hotel Company. These actions result in new information, including objects for the customer’s itinerary, her confirmation, and her invoice. In Figure 3, all the messages between the involved parties indicate the behavior of the web services shown in Figure 1.

When the customer requests a Reservation, the Travel Agency forwards it to the Hotel Company, which invokes a checkAvailability service for checking the availability of the rooms in some locations passing it to the list of reservations. After checking availability in several locations, several lists of rooms that match are retrieved from partners. The Hotel Company now invokes the priceOffer web services of its sub-partners and provides the respective list of items to each of them.

The priceOffer web service checks the availability of rooms on the list in order to return a list of prices and availability on specific locations. The Hotel Company then invokes a calculateOffer web service of the hotel. For this purpose, the control process passes the lists returned from the partners to the calculateOffer web service for final prices.

The calculateOffer web service identifies the proper request of the travel agency’s reservation by processing the data passed by the checkAvailability web service. Finally, the offer is returned to the control process and will be passed to checkOffer web service of the travel agency, which in turn returns an ‘OK’ or ‘Reject’ response to the control process. Upon accepting the offer from the hotel company, the billOffer web service invokes the chargeBill web service for the bank to charge the customers the required fees and notify the travel agency to finalize the process.

![Figure 2: A BPEL Activity diagram for reserving a room](image)
IV. THREATS TO THE ACTIVITIES

Each activity is potentially susceptible to attack, although not necessarily through the computer system. Figure 4 shows the same activity diagram with some possible threats. The attacks are presented as threats (dotted lines). Undesired consequences in the form of additional or alternative objects (dotted lines) have also been added.

We only show some of the threats; in particular, for the last activities:

- **Threat T1 (Illegal dissemination).** The travel agency collects the customer’s itinerary and uses it illegally.
- **Threat T2 (Charge Spurious Fees).** The travel agency charges the customer spurious fees.
- **Threat T3 (Sends Spurious Ticket and Invoice).** The travel agency sends spurious ticket and/or invoice.

Note that:

- We can list systematically all (or most) possible application threats. While completeness cannot be assured, we can consider at least all important possible attacks. The threats that we postulate come from our experience, from the knowledge of the application, and from the study of similar business processes (many online shopping processes have similar threats).
- We can identify internal and external attackers. The actors in these attacks could be external attackers (hackers). It is also possible that a person in a legitimate role can be malicious (internal attacks). For example, Threat T1 and Threat T3 are performed by insiders, while Threat T2 is performed by either external or internal attackers.

- We are not restricted to analyze each activity in isolation. Some workflows require several activities, e.g. "Cancel Reservation" could be followed by "Refund to Customer". We can consider attacks that take advantages of this sequence, for example, by bypassing some steps that perform checks. These threats, in general, are harder to find.

V. STOPPING OR MITIGATING THE THREATS

After we enumerate systematically the threats, we can now find out what policies are needed to stop these attacks. For this purpose, we can select from the typical policies used in secure systems [11], [12]. This selection should result in a minimum set of mechanisms instead of mechanisms piled up because they might be useful. For example, to avoid impostors we can have a policy of I&A (Identification and Authentication) for every actor participating in a business process. Table 1 shows the specific threats and policies for the last activities of Figure 4.

**TABLE1.** Mitigating or stopping threats to a business process using security policies

<table>
<thead>
<tr>
<th>Threats</th>
<th>Description</th>
<th>Security Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat 1 Illegal</td>
<td>The travel agency collects customer’s itinerary and uses it illegally.</td>
<td>Logging</td>
</tr>
<tr>
<td>dissemination</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Threat 2 Charge</td>
<td>The travel agency charges the customer spurious fees.</td>
<td>Protection against denial of service.</td>
</tr>
<tr>
<td>Spurious Fees</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Threat 3 Sends</td>
<td>The travel agency sends spurious ticket and invoice.</td>
<td>Separation of administration from use of data</td>
</tr>
<tr>
<td>Spurious Ticket and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invoice</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The security policies are chosen to mitigate or stop the threats defined in the activity diagrams. By analyzing more activities in the activity diagram, we are able to capture more threats, for which other security policies are needed. These policies are realized using security mechanisms, using appropriate metamodels as shown in [13]. In that approach, security mechanisms are realized as security patterns, where a security pattern is an encapsulated solution to a recurrent security problem. A policy may be realized by one or more patterns.
We can complement this analysis by considering any operation from the partners (Hotel Company, Bank) should require authentication, since it’s not acceptable that anyone who knows the web service link could have the ability to reserve a room or perform bank transactions. For that reason, the web service composer has to conform to the partner’s security policy before writing down the BPEL process that will invoke those services. To participate with the BPEL process, the security policy for hotel and bank web services must define which security model and mechanism (certificate, encryption algorithm, digital signature, etc.) they support.

Now that the partner web service can verify the identity of the requestor, the next step is to decide what the requestor is allowed to do. For non repudiation issues, it’s important that the office which did the reservation cannot deny doing so and that nobody can misuse its identity for malicious activities. To fulfill this requirement, digital signatures can be used. Another issue is data integrity; we need to make sure that when the offices perform the reservation process, it’s mandatory that nobody can modify the reservation and change its data. Appropriate mechanisms are also needed to avoid replay attacks, where attackers try to copy the reservation order and resend it again. For that kind of problems, a timestamp mechanism is applicable.

For the bank’s payment web service, we give more importance to confidentiality since sensitive data transferred between the travel agency and the bank should not be read by unauthorized people. The agreement to choose an encryption and decryption mechanism is a good choice.

Figure 5 summarizes this discussion and addresses the security considerations for the travel agency as a whole. Since some of the parties involved in this BPEL are external parties, the indicated mechanisms should be considered as expected requirements. On the travel agency case, the selection of those considerations depends on the degree of external and internal communication, and on the sensitivity of the data being transmitted. Note that this approach indicates only the security mechanisms needed in each unit; we need to continue refining the model to indicate which specific web services require which type of security mechanisms following a general approach we proposed in [9].
VI. CONCLUSIONS AND FUTURE WORK

We have presented an approach that enumerates the threats to a given BPEL process. We considered UML activity diagrams for collaborative business processes and showed how to list the possible threats and attacks that could happen in order to define the appropriate and suitable countermeasures to stop or mitigate them. The use of UML activity diagrams produces a clear and more intuitive way to analyze these attacks than working directly in BPEL. The first one to suggest making security annotations to activity diagrams was S. Johnston [18] and this idea has been applied in all the surveyed approaches, including ours. Our claim for improvement lies in that we can enumerate threats without the need of a security expert.

Future work includes a design model for our example of travel agency, which will explain how to deploy web services standards such as WS-Security, WS-Policy and WS-Trust in a systematic way. Such a design model will use our threat enumeration approach to specify exactly which security mechanisms we should deploy for which classes. We intend to incorporate also this approach as part of our secure application design methodology [9], which now starts from use cases instead of workflows. This would allow building systems combing web services and standard components, which are necessary in real architectures. Our approach starts from use cases and class/sequence diagrams’ conversion to BPEL models can be done using an approach such as the one in [19].

REFERENCES
