Designing Secure Service Workflows in BPEL

Luca Pino, Khaled Mahbub, and George Spanoudakis

Department of Computer Science, City University London, London, United Kingdom
{Luca.Pino.2,K.Mahbub,G.E.Spanoudakis}@city.ac.uk

Abstract. This paper presents an approach that we have developed to support the design of secure service based applications in BPEL. The approach is based on the use of secure service composition patterns, which are proven to preserve composition level security properties if the services that are composed according to the pattern satisfy other properties individually. The secure service composition patterns are used for two purposes: (a) to analyse whether a given workflow fragment satisfies a given security property, and (b) to generate compositions of services that could substitute for individual services within the workflow that cause the violation of the security properties. Our approach has been implemented in a tool that is based on Eclipse BPEL Designer.

1 Introduction

An important concern in the development of a service-based application (SBA) is the ability to assure that the application will have certain security properties. Assuring security is important for any application but acutely so in the case of SBAs. Such applications, in fact, are based on services which might not be under the control of the SBA provider and can compromise critical security properties (e.g., the integrity and confidentiality of data passed to, stored or produced, or the availability).

An increasingly accepted view on how to best assure security is that security properties should be achieved by design rather than be dealt with as an aftermath concern. Despite being increasingly adopted in the design of normal software applications security-by-design is not so well supported in the case of SBAs. SBA design is typically iterative focusing on the development of an orchestration model to coordinate the services that will constitute the SBA [3]. During it, it is necessary to discover services that can fit with the orchestration model that is being designed or, where this is not possible, to change the orchestration model in a systematic manner in order to make it fit with the available services whilst preserving required properties.

Existing approaches are effective in discovering individual services (e.g., [1,2,3]) and service compositions that have functionality and quality properties that are compatible with SBA designs (e.g., [4,5,6]). However, they do not support effectively the discovery of individual services and service compositions with required security properties, and the validation of the overall security of a service orchestration process when the discovered individual services are composed into it. This paper presents an approach that we have developed to address this problem.

Our approach supports the design of secure SBAs. It is based on the use of *Secure Service Composition patterns* (SSC patterns), which are proven to preserve certain composition level security properties if the services that are composed according to the pattern satisfy other properties individually. SSC patterns are used for two purposes: (a) to analyse whether a given workflow fragment satisfies a given security property, and (b) to generate service compositions that could substitute for individual services within a workflow that cause the violation of the security properties required of it. Our approach supports also the replacement of individual services, which violate given security properties, by other individual services or compositions that are discovered based on properties identified by the patterns. The satisfaction of security properties at the service level is determined by digital service security certificates. We implemented our approach in a tool that extends Eclipse *BPEL Designer* [7].

The paper is structured as follows. Section 2 presents scenarios of secure SBA process design. Section 3 introduces the SSC patterns. Section 4 presents the validation and adaptation supported by the SSC patterns. Finally, Section 5 reviews related work and Section 6 summarizes our approach and outlines directions for future work.

2 Scenarios for Secure Workflow Design

To exemplify our approach, assume an SBA, called *StockBroker* allowing stock investors to buy and/or sell stocks in different stock exchanges. Upon receiving a request from an investor, *StockBroker* retrieves the investor's portfolio of stocks, and fetches the trading values of a selected stock and index of the relevant stock market (e.g. NASDAQ, Dow Jones). It then matches these values with the preferences of the investor and contacts different services to carry out the trade and to pay for it.

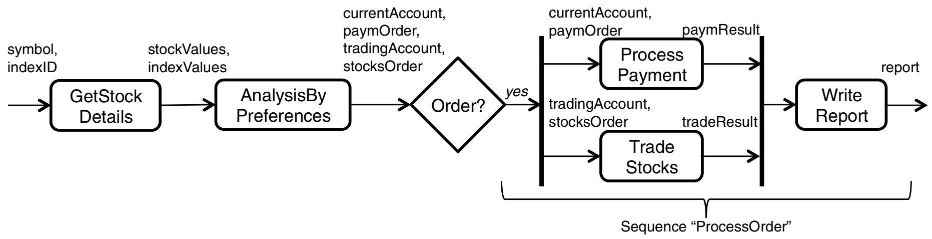

Fig. 1. The *StockBroker* BPEL workflow

Fig. 1 shows the workflow that realises *StockBroker*. This workflow receives a stock symbol and a stock market index ID; invokes a stock information service (cf. activity *GetStockDetails*) to get the details for the given stock in the particular market; matches these details with preferences (cf. activity *AnalysisByPreferences*); and, if a trade order is to be placed, it invokes in parallel the payment service (cf. activity *ProcessPayment*) and the trading service (cf. activity *TradeStocks*)[1]. Finally, a report of all results is produced by the reporting service (cf. activity *WriteReport).*

[1] Carrying trading in parallel with payment is possible as clearing of payment transactions can be completed after the trade transaction has taken place.

In designing secure service workflows, we have identified two scenarios. In the first scenario (Scenario 1), an SBA designer wants to verify if an external service operation, used in the workflow through an *invoke* activity, satisfies a required security property. In this scenario, if the service that is currently bound to the activity does not satisfy the property, support is offered to discover alternative services that would satisfy the required property and, if no such individual services can be found, to explore if it is possible to build a composition of other services that satisfies the security property and could, therefore, be used as a substitute for the original service. An example of a composition is shown in Fig. 2. The composition *ParallelStockDetails* shown calls two service operations in parallel, namely *GetStockValues* and *GetStockMarketIndex.GetStockValues* returns the trading value for a stock, identified by its symbol, and *GetStockMarketIndex* returns the value of a stock market index.

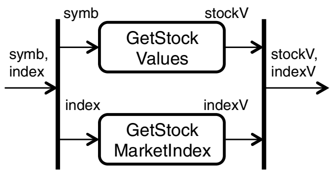

Fig. 2. Service composition *ParallelStockDetails* to be substituted for *GetStockDetails*

The second scenario arises in cases where the SBA designer wishes to verify that a part of a workflow (as opposed to an individual activity of it) satisfies a given security property. Workflow fragments are identified (delimited) by a control flow activity. In the *Stock Broker* workflow, for instance, a designer might wish to verify whether the sub sequence of activities designated as *ProcessOrder* in Fig. 1 preserves the confidentiality of the personal current account information of *Stock Investor.*

3 Secure Service Composition Patterns

SSC patterns are used to specify how security properties of whole abstract workflows (i.e., composition level security properties) can be guaranteed via security properties of the individual services used in the workflow. The causal relation between workflow and activity level properties specified in such patterns is formally proven.

An SSC pattern is composed of: (a) an abstract workflow structure (*Pattern.WF*), called *workflow specification*, that indicates how services are to be composed and the data flows between them; (b) the composition level *security property* that the pattern guarantees (*Pattern.CSP*); and (c) the *security properties* required of the *partner services* that may be bound to the workflow specification (i.e., to the abstract *invoke* activities of the workflow) to guarantee the security property specified in (b) (*Pattern.ASP*). SSC patterns are expressed as rules of the production system Drools [9], to enable their application for workflow security validation and adaptation.

In the following, we present an example of an SSC pattern that we have encoded specifying the effect of composition on the security property of *separability*. Separability is a security property introduced in [20] and has been defined as complete independence between high (confidential) and low level (public) sequences of actions. For this property to hold there should be no interaction between confidential and public sequences of actions (e.g., running these actions as two separate processes without any communication between them). The composition of separability, proven in [20,21], is used for specification of the SSC pattern in Drools as given in Sect. 4.1.

4 Application of SSC Patterns

SSC patterns are used to infer the security properties that the individual services should have for the workflow to have another security property as a whole. This allows to: (a) analyse whether a given workflow (or a fragment of it) satisfies a given security property (security validation); and (b) generate compositions of services that could substitute for individual services, which prevent the satisfaction of the security properties required (security driven workflow adaptation). In the following, we present the approaches that enable these forms of applications.

4.1 Inferring Security Properties of Workflow Activities

SSC patterns are used to infer the security properties, which have to be satisfied by the individual activities (services) of a composition, for the whole composition to satisfy a given security property. In general, there can be zero, one or several alternative combinations of activity level properties, called *security solutions*, that can guarantee the security property required of the composition. The algorithm that applies SSC patterns for this purpose is given in Table 1.

Table 1. Algorithm to infer security properties for activities within a composition

```
Algorithm: INFERSECPROPERTY(WF, RSP, InSolutions): OutSolutions
Inputs:   WF – /* workflow specification of a service composition process */
          RSP – /* security property requested for WF */
          InSolutions– /* list of security solutions used for recursion. Base case: {RSP} */
Output:   OutSolutions – /* list of security solutions for the activities in WF */
-------------------------------------------------------------------------------
For each pattern Patt such that Patt.CSP matches RSP do
    If Patt.WF /* i.e. the workflow specification of Patt */ matches WF then
        For each element E /* i.e. individual activity or a sub-workflow */ of WF do
            Properties[E] := security properties for WF.E identified by Patt.ASP
        For each security solution S in InSolutions do
            S' := replace RSP by Properties in S
            $SolutionList_{Patt}$ := ADD($SolutionList_{Patt}$, S')
        For each element E in WF that is a sub-workflow specification do
            $SolutionList_{Patt}$ := INFERSECPROPERTY(E, Properties[E], $SolutionList_{Patt}$)
        OutSolutions := ADDALL(OutSolutions, $SolutionList_{Patt}$)
    Endif
Return OutSolutions
```

As shown in the table, given an input service workflow WF and a required security property RSP, the algorithm (INFERSECPROPERTIES) tries to apply all the SSC patterns that would be able to guarantee the requested security property RSP. A pattern is applied if the workflow specification of the pattern (*Pattern.WF*) matches with WF.

If a pattern matches the workflow, then the security solutions computed up to that point are updated to replace the requested security property RSP with the security properties for the matched elements in WF (these can be individual activities or sub-workflows). If a matched element E of WF is an atomic activity, the process ends w.r.t it. If E is a sub-workflow, the algorithm is applied recursively for it.

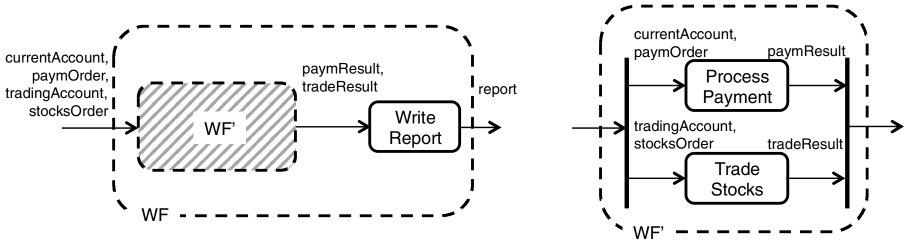
currentAccount, paymOrder, tradingAccount, stocksOrder
WF'
paymResult, tradeResult
Write Report
report
WF
currentAccount, paymOrder
paymResult
Process Payment
tradingAccount, stocksOrder
tradeResult
Trade Stocks
WF'

Fig. 3. The workflow patterns of sequence *ProcessOrder*

As an example of applying INFERSECPROPERTIES consider the case where an SBA designer wishes to verify that the subprocess *ProcessOrder (PO)* within the *Stock-Broker* process of Fig. 1 preserves the confidentiality of the Stock Investor current account. This security property can be expressed as *separability*, with *currentAccount* being confidential. *ProcessOrder* can be seen as a sequential workflow consisting of a sub-workflow WF' and the atomic activity *WriteReport* that follows it (see Fig. 3). WF' itself is a parallel workflow involving two atomic activities: *ProcessPayment* and *TradeStocks*.

Table 2. Specification of a pattern in Drools

```
rule "Separability on Parallel Workflow"
 when
   $wf : Parallel( $A1 : act1, $A2 : act2 )
   $csp : Property( propertyName == "Separability", subject == $wf, $cspAttr : attributes )
   $solution : Solution( properties contains $csp )
 then
   Solution newSolution = (new Solution($solution).removeProperty($csp);

   Property asp1 = new Property ($csp, "Separability", $A1);
   asp1.getAttributes().put("public", new Subset($cspAttr.get("public")));
   asp1.getAttributes ().put("confidential", new Subset(new Complement($cspAttr.get("public"))));
   newSolution.getProperties().add(asp1);
   insert(asp1);
   Property asp2 = new Property ($csp, "Separability", $A2);
   asp2.getAttributes().put("public", new Subset($cspAttr.get("public")));
   asp2.getAttributes ().put("confidential", new Subset(new Complement($cspAttr.get("public"))));
   newSolution.getProperties().add(asp2);
   insert(asp2);

   insert(newSolution);
end
```

Hence, when InferSecProperties is applied on to it, in the first iteration an SSC pattern for the sequential flow can be applied on WF, returning two security properties: one for WF' requiring confidentiality for *currentAccount*, *paymResult* and *tradeResult*, and another for *WriteReport*, requiring confidentiality for *paymResult* and *tradeResult*. The second iteration of the algorithm applies another SSC pattern, but for the parallel flow, to WF'. In particular InferSecProperties applies SSC patterns specified as rules of the Drools production system [9]. Table 2 shows the specification of the SSC pattern about separability on parallel flow (see [21]) as a Drools rule.

More specifically, the rule defines that if the workflow (*$wf*) is a parallel composition of activities and the composition level security property is separability (*$csp*) then the security property of separability is required of the individual activities *$A1* and *$A2* of the composition (this is expressed by the property *asp1* and *asp2*). Hence, by applying the rule of the SP pattern to WF', the algorithm creates and adds two security properties to the final solution, i.e., *asp1* (separability) for *currentAccount* and *paymResult* of *ProcessPayment* and *asp2* (separability) for *tradeResult* of *TradeStocks*.

4.2 Validation of Security of Individual Services and Workflow Fragments

In order to validate whether a security property is satisfied by a fragment of a workflow, we assume that a fragment consists of a BPEL *scope* or a control flow (i.e., *sequence*, *flow*, *if-then-else* or *pick*) activity that can contain multiple service invocations (in the form of *invoke* activities) and further control flow activities.

Given a request to verify whether a workflow fragment (WF) satisfies a required security property (RSP), the algorithm InferSecProperty is applied to identify the list of alternative security solutions (i.e., combinations of security properties of the individual services in the fragment) that would guarantee RSP. As explained earlier InferSecProperty tries to apply different SSC patterns in order to identify these alternative solutions. If such solutions exist, each of them is analysed further to check if the security properties required by it are provided by the services in the fragment.

To validate whether an individual service satisfies the security property required of it by a security solution, we express the property as a service discovery query and then use the discovery algorithm described in [8] to match the specification of the individual service with the query and establish if it satisfies the query or not. In applying the service discovery process, we assume the existence of machine-readable security certificates that indicate the security properties that a service S has [8]. If the individual service validation succeeds for all the services of the fragment by even one of the identified security solution, then the fragment is validated. Otherwise, if no security solution can be found, or if none of the found security solution can be satisfied by the services in the fragment, the fragment is reported as not validated.

4.3 Workflow Adaptation

In certain cases, it might be possible to adapt a workflow in order to make it satisfy a required security property. In our approach, this adaptation can take two forms, by: (a) replacing individual services in it by other individual services, or (b) replacing individual services in it by service compositions that are constructed in a way that guarantees the security property required of the service to be replaced. When a workflow fragment is not validated, the SBA designer can compare and select the security solutions for the workflow fragment found by the validation algorithm. This allows to replace the security property over the fragment with security properties over the *invoke* activities within it. Once a specific security solution is selected, the service adaptation mechanism is triggered to adapt the workflow.

4.4 Implementation of the Approach

Our approach has been implemented in a tool called *A-BPEL Designer*. This tool is an extension of *BPEL Designer*, i.e., an Eclipse plugin [7] that offers comprehensive support for the editing and deployment of WS-BPEL processes through Eclipse IDE. In *A-BPEL Designer*, we have extended BPEL by allowing the specification of security properties for *invoke* or *control flow* BPEL. *A-BPEL Designer* offers also support for validating security properties of individual partner services or workflow fragments and adapting BPEL processes to ensure security as described. To offer these functionalities, A-BPEL designer has been integrated with the service discovery engine described in [3] and the service composition tool described in [8].

5 Related Work

Research related to the security of service based applications focuses on making secure an SBA, or verifying its security.

A common approach underpinning research in the former area is to secure SBAs by using additional security services that can enforce the required security properties [12,13,14]. More specifically, an aspect-oriented version of BPEL, called AO4BPEL [12], allows the integration of security specifications in a BPEL process. These specifications are then used to indicate security functionalities that are offered by a special Security Service, and integrate them in the AO4BPEL process.

Sectet [13] is a framework for the implementation of security patterns from design to the implementation of an orchestration. Sectet enables the design of orchestrations as UML message flow diagrams, which are converted into workflows and used to generate stubs for actual orchestrations. In orchestrations, services are wrapped by Policy Enforcement Points, whose purpose is to provide the required security properties.

PWSSec [14] describes a set of complementary stages to be added to the SBAs development phases in order to support security. In particular the WSSecArch is a design phase that takes care of the indications about which security requirements are achieved and where they are in the architecture. The approach makes usage of security architectural patterns to convert the security requirements into architecture specifications, with external security services providing the security functionalities.

Unlike the above approaches, our approach does not use special types of security components or services but supports the discovery of normal services and service compositions that themselves have the security properties required of an SBA.

Attention has been given also to the model based verification of security properties during the design of orchestrations [15,16,17]. These works usually require a UML specification of the system, the security threats associated with it and the description of required properties in order to verify the satisfiability of the latter. Our approach does not require the specification of threats. Furthermore, it does not perform exhaustive verification since its analysis is driven by specific SSC patterns. This is important as it makes security analysis more scalable at the expense of loss of completeness.

Some model based approaches [18,19] support also the transformation of from security requirements into security policies and architectures. This usually happens in an early design phase that must be followed by a subsequent phase where details about the implementation have to be worked out. Our approach offers the possibility to add and address security properties during the workflow design phase, without requiring designer to have a security background.

The METEOR-S project [10] allows annotation of abstract BPEL process to specify semantic-aware QoS properties, including security. The annotations are then used to discover appropriate services for the BPEL process, using an annotated registry. The Sec-MoSC (Security for Model-oriented Service Composition) tool [11] is an extension of the Eclipse BPMN Modeller that allows to design BPMN business processes and to add security properties to them. These two approaches focus only on the validation single service of security properties, while our approach allows the validation of workflow fragments and the substitution of services with service compositions.

6 Conclusion

In this paper we have presented an approach supporting the validation of security properties of BPEL workflows and the security based adaptation of such workflows during their design. A-BPEL Designer implements this approach in the Eclipse platform through the usage of a service discovery engine.

Our approach is based on Secure Service Composition (SSC) patterns, which encode formally proven causal relations between individual service level security properties and composition level security properties. The validation of workflow security is based on identifying (through the SSC patterns) the security properties that the individual partner services need to have for the workflow to have composition level properties. The identified service level properties are used to check if existing partner services satisfy them, discover alternative services for them in case they do not, and discover service compositions satisfying the services if necessary. Our approach supports also the automatic replacement of security non-compliant services.

Our current implementation supports workflows with sequential, parallel and choice control activities (i.e., BPEL *sequence*, *flow*, *if-then-else* and *pick* activities), and the replacement of individual service invocations. Hence, in its current form, its application is restricted to non-transactional and stateless services.

Our on-going work focuses on supporting transactional services. We are also conducting performance and scalability tests, in order to compare our results with competing approaches (especially approaches based on full verification of security).

Acknowledgment . The work presented in this paper has been partially funded by the EU F7 project ASSERT4SOA (grant no.257351).

References

1. Pawar, P., Tokmakoff, A.: Ontology-Based Context-Aware Service Discovery for Pervasive Environments. In: 1st IEEE International Workshop on Services Integration in Pervasive Environments (SIPE 2006), in conjunction with IEEE ICPS 2006 (2006)
2. Mikhaiel, R., Stroulia, E.: Examining usage protocols for service discovery. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 496–502. Springer, Heidelberg (2006)
3. Spanoudakis, G., Zisman, A.: Discovering Services During Service Based Systems Design Using UML. IEEE Trans. on Software Eng. 36(3), 371–389 (2010)
4. Fujii, K., Suda, T.: Semantics-Based Dynamic Web Service Composition. IEEE Journal on Selected Areas in Communications 23(12), 2361–2372 (2005)
5. Silva, E., Pires, L.F., van Sinderen, M.: On the Support of Dynamic Service Composition at Runtime. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 530–539. Springer, Heidelberg (2010)
6. Pino, L., Spanoudakis, G.: Constructing Secure Service Compositions with Patterns. In: IEEE SERVICES 2012, pp. 184–191. IEEE Press (2012)
7. BPEL Designer Project, `http://www.eclipse.org/bpel/`
8. ASSERT4SOA Consortium: ASSERTs Aware Service Based Systems Adaptation. ASSERT4SOA Project, Deliverable D2.3 (2012)
9. Drools – Jboss Community, `http://drools.jboss.org`
10. Aggarwal, R., Verma, K., et al.: Constraint Driven Web Service Composition in METEOR-S. In: IEEE SCC 2004, pp. 23–30. IEEE Press (2004)
11. Souza, A.R.R., et al.: Incorporating Security Requirements into Service Composition: From Modelling to Execution. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 373–388. Springer, Heidelberg (2009)
12. Charfi, A., Mezini, M.: Using aspects for security engineering of web service compositions. In: IEEE ICWS 2005, pp. 59–66. IEEE Press (2005)
13. Hafner, M., Breu, R., et al.: Sectet: An extensible framework for the realization of secure inter-organizational workflows. Internet Research 16(5), 491–506 (2006)
14. Gutiérrez, C., Fernández-Medina, E., Piattini, M.: Towards a process for web services security. J. of Research and Practice in Information Technology 38(1), 57–68 (2006)
15. Bartoletti, M., Degano, P., et al.: Semantics-based design for secure web services. IEEE Trans. on Software Eng. 34(1), 33–49 (2008)
16. Deubler, M., Grünbauer, J., Jürjens, J., Wimmel, G.: Sound development of secure service-based systems. In: ICSOC 2004, pp. 115–124. ACM, New York (2004)
17. Georg, G., Anastasakis, K., et al.: Verification and trade-off analysis of security properties in UML system models. IEEE Trans. on Software Eng. 36(3), 338–356 (2010)
18. Menzel, M., Warschofsky, R., Meinel, C.: A pattern-driven generation of security policies for service-oriented architectures. In: IEEE ICWS 2010, pp. 243–250. IEEE Press (2010)
19. Séguran, M., Hébert, C., Frankova, G.: Secure workflow development from early requirements analysis. In: IEEE ECOWS 2008, pp. 125–134. IEEE Press (2008)
20. McLean, J.: A general theory of composition for trace sets closed under selective interleaving functions. In: 1994 IEEE Symp. on Sec. and Privacy, pp. 79–93. IEEE CS Press (1994)
21. Mantel, H.: On the composition of secure systems. In: 2002 IEEE Symp. on Sec. and Privacy, pp. 88–101. IEEE CS Press (2002)