2016

Engineering Secure Adaptable Web Services Compositions

Peter T. Nkomo
University of Johannesburg, pnkomo@gmail.com

Marijke Coetzee
University of Johannesburg, marijkec@uj.ac.za

26. Engineering Secure Adaptable Web Services Compositions

Peter T Nkomo
University of Johannesburg
pnkomo@gmail.com

Marijke Coetzee
University of Johannesburg
marijkec@uj.ac.za

Abstract

Service-oriented architecture defines a paradigm for building applications by assembling autonomous components such as web services to create web service compositions. Web services are executed in complex contexts where unforeseen events may compromise the security of the web services composition. If such compositions perform critical functions, prompt action may be required as new security threats may arise at runtime. Manual interventions may not be ideal or feasible. To automatically decide on valid security changes to make at runtime, the composition needs to make use of current security context information. Such security changes are referred to as dynamic adaptation. This research proposes a framework to develop web services compositions that can dynamically adapt to maintain the same level of security when unforeseen security events occur at runtime. The framework is supported by mechanisms that map revised security requirements arising at runtime to a new security configuration plan that is used to adapt the web services composition.

Keywords

Adaptation, Security, Web services composition, Monitoring, Policies, Runtime, Strategy, Aspect, Variability

1. Introduction

Service-oriented architecture (SOA) (Erl 2008) has gained attention as a unifying architecture that addresses the challenges of deploying applications where rapidly changing business requirements are present. SOA applications are built by assembling or composing autonomous components called web services (Erl 2008) to create a web services composition. Such web services may be owned and maintained by different service providers spanning many administrative boundaries. It is therefore possible for a web services composition to invoke third party web services without any guarantee that the web service being invoked meets the expected security requirements. Enforcement of the security properties of a web services composition, using mechanisms such as authentication, authorization, confidentiality and integrity, can be affected and may need to be changed at runtime (Hoque et al. 2013). When such changes occur, it becomes necessary for a web services composition to ensure that the expected security is maintained. When a specific web service poses a security risk to the entire web services composition, the ability of the web services composition to dynamically replace such a web service with a more suitable and secure web service becomes a desirable feature. Even if such a web service supports the required level of security, and does not pose a security risk, the web services composition may still need to be reconfigured to implement new security mechanisms to support possible changes in the security properties.

Despite the recognized need to handle unexpected security events that can occur at runtime, the ability to dynamical reconfigure the web services composition due to unforeseen security events is still an open and challenging research topic (Alferez & Vicente 2013). Most existing approaches to secure web services compositions focus on capturing and enforcing security requirements at design time without paying attention to how a web services composition could dynamically adapt to cope with unanticipated security requirements that may arise at runtime. From a security perspective, the web services composition may often be left incomplete and insecure.

This paper proposes a high level framework to allow a services composition to dynamically adapt its security by reacting to changing security requirements that arise at runtime. The contribution of this paper is a set of steps to create models that sufficiently describe a web services composition's security context to enable runtime security requirements to be mapped to flexible runtime security configurations. The framework further provides concrete mechanisms to automatically generate and enforce a security adaptation strategy to ensure the security of the web services composition.

This paper is organized as follows: Section 2 introduces the motivating scenario. Section 3 identifies the security challenges of web services compositions and section 4 lists requirements for a secure and adaptable composition. Section 5 gives a review of literature that attempts to satisfy the security requirements discussed in section 4. In section 6 a framework to create a web services composition that can adapt when changes on the security context occur is presented. Section 7 concludes the paper.

2. Illustrative scenario

In order to understand the challenges encountered when developing secure and adaptable web services compositions that are able to react to unexpected runtime events, a motivating example of a virtual wholesaler services composition, consisting of Retailers, Suppliers and a Bank is presented in Figure 1.

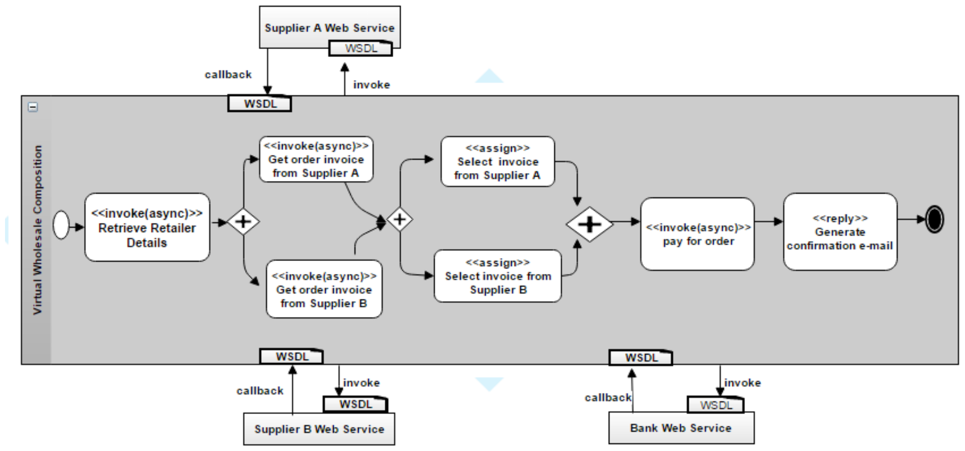

Supplier A Web Service
WSDL
callback
invoke
WSDL
Virtual Wholesale Composition
<<invoke(async)>> Retrieve Retailer Details
<<invoke(async)>> Get order invoice from Supplier A
<<invoke(async)>> Get order invoice from Supplier B
<<assign>> Select invoice from Supplier A
<<assign>> Select invoice from Supplier B
<<invoke(async)>> pay for order
<<reply>> Generate confirmation e-mail
WSDL
callback
invoke
WSDL
Supplier B Web Service
WSDL
callback
invoke
WSDL
Bank Web Service

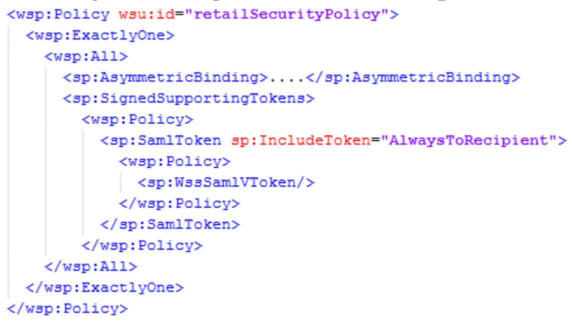
Figure 1. Virtual wholesaler services composition model

The web services composition enables different retailers to place orders and make payments for inventory sourced from different product suppliers. The virtual wholesaler web services composition is a typical SOA based application that uses a workflow defined using the Business Process Execution Language (BPEL) (Song et al. 2011). The composing is designed with a set of security requirements which include (i) encrypt sensitive information in all communications, (ii) only invoke partner web services with valid digital signatures, (iii) authenticate all parties in all interactions among web services and (iv) retailers with a valid digital certificate can access the web services composition.

Each web service participant in the virtual wholesaler composition is described in an interface document using the Web Services Description Language (WSDL) (Curbera et al. 2002). The security requirements of each web service is defined in XML-based WS-Policy (Rosenberg & Remy 2004) and WS-SecurityPolicy (Rosenberg & Remy 2004) policy documents and referenced from the WSDL of the web service at the operation or endpoint level. Figure 2 gives a snippet of the retail web service security policy that requires that all interactions to and from it use Security Assertion Markup Language (SAML) (Rosenberg & Remy 2004) authentication mechanisms.

When attempting to find a third party web service to use in a web services composition, the Universal Description, Discovery, and Integration (UDDI) (Curbera et al. 2002) registry can be searched. A web service is chosen based on a match of the functionality it provides, and whether it supports the security requirements of the web services composition. For example, consider the security configuration of a retailer web service that satisfies all the web services composition SAML authentication security requirements listed above. When a user accesses functionality provided by the web services composition via the retailer web portal, he/she logs in using a username token. Via a single sign-on mechanism, the user credential is submitted to the web services composition as a SAML assertion, which is one or more statements about the user. The private key linked to the portal signs both the assertion and the SOAP message body. The portal thus vouches for contents of the user message and the SAML assertion.

Securing a web services composition such as the virtual wholesaler is not trivial for a number of reasons. The next section discusses the security challenges of such a composition.



```xml
<wsp:Policy wsu:id="retailSecurityPolicy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:AsymmetricBinding>....</sp:AsymmetricBinding>
      <sp:SignedSupportingTokens>
        <wsp:Policy>
          <sp:SamlToken sp:IncludeToken="AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssSamlVToken/>
            </wsp:Policy>
          </sp:SamlToken>
        </wsp:Policy>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Figure 2. Retail security policy

3. Runtime security challenges

Web services compositions such as the virtual wholesaler are deployed in complex environments using technologies that may be prone to security attacks (Tiwari & Singh 2011). For example, WSDL documents, used to describe web services, can be generated automatically by current web services frameworks to reveal the entire application programming interface. This may provide enough information to mount a successful web service attack at runtime. Many UDDI registries do not provide robust mechanisms to verify the author of a web service, potentially allowing the addition of malicious web services to the registry. Such malicious web services could later be bound to the runtime of unsuspecting web services compositions. The SOAP messaging framework supports intermediary nodes between the sender and receiver making Secure Socket Layer or Transport Layer Security (SSL/TLS) unsuitable to use as it only functions using a point-to-point messaging model.

The challenge with a web services composition is that security of the entire composition depends on the security capabilities of each of the composed web services. This dependency occur within the context of the complex and dynamic relationship between the composition and the composed web services. In the case of the virtual wholesaler composition, it is shared between the various participating retailers, product suppliers and banks. Ensuring the security of such a web services composition is non-trivial. Various events and changes can occur within the web service composition runtime context that can affect its security. It is therefore desirable to rely on mechanisms to dynamically adapt the security of a web services composition during runtime, according to problematic events that occur in those specific contexts (Hoque et al. 2013). As there is a strong possibility that such problematic events can occur, it is important for the web services composition to understand and keep track of security properties of each of the participant web services at runtime. Understanding the lifecycle of a web service's security properties allows a web services composition to make an informed judgment about the likelihood that such a web service will be able to meet expected security requirements.

In order to understand the development of a secure adaptable web services composition, adaptation requirements of a web services composition are discussed in the next section.

4. Adaptation requirements

Traditionally, the development cycle of a web services composition involves gathering of functional and non-functional requirements, modelling of the web services composition, searching and selection of web services at design time, and finally binding and invocation of selected web services at runtime (Immonen & Pakkala 2014). Research by IBM has defined an architectural blueprint for self-adaptive systems (Kephart et al. 2007) that gives an understanding of the characteristics of a system that can dynamically react to problematic events in an execution context. From the architectural blueprint, this research identifies three properties that are applicable to secure dynamic web services compositions namely *self-healing*, *self-protecting* and *self-configuring*.

Self-healing is the ability to discover, diagnose and take proper actions to prevent a failure (Kephart et al. 2007). Self-protecting is the capability to anticipate, detect, identify and protect against threats. In the virtual wholesale web services composition, self-healing and self-protecting ensures that the web services composition proactively identifies security threats and reacts appropriately without manual intervention. Self-configuring is the ability to dynamically adapt to changing environments by composing or decomposing web services. Figure 3, adapted from the work by Immonen & Pakkala (2014) shows the design and runtime phases of a web services composition that reacts to runtime security changes. At the left, composition requirements are gathered, to finally be used in composition runtime adaptation and monitoring processes.

Using self-healing, self-protecting and self-configuring properties three *adaptation requirements* are now identified as follows:

- Events that affect security need to be identified or detected at runtime using *monitoring mechanism*s.
- Security requirements that arise at runtime need to be captured and enforced by having *support for runtime security requirements.*
- The security of the web services composition needs to be reconfigured to maintain its level of assurance by using *adaptation mechanisms*.

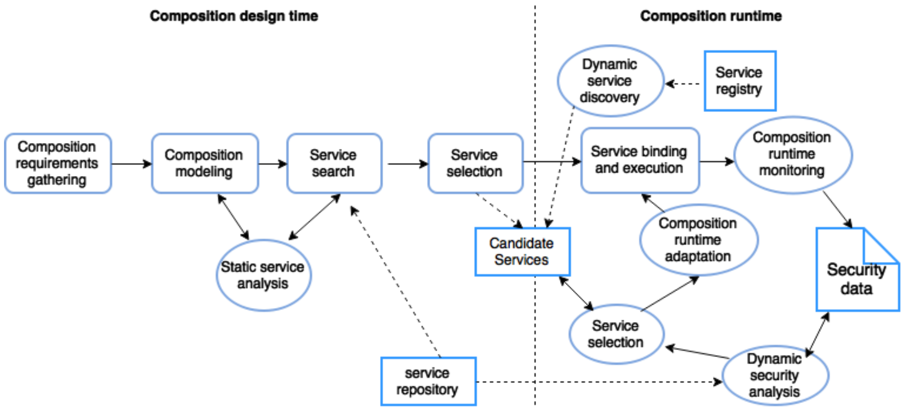


Figure 3: Web services composition design time and runtime phases

The next three sections discuss these adaptation requirements in more detail.

4.1 Monitoring mechanisms

To enable a web services composition to adapt to its security context it needs to capably detect changes in its operating context (Zhou et al. 2012). For example, when the retail web service in the virtual wholesale composition changes its security requirements and updates its security policies, the virtual wholesale web services composition need to dynamically detect such changes and react accordingly. The web services composition is therefore required to *monitor* itself and its context, *detect* problematic changes, *decide* how to react and *act* to satisfy its security requirements. This research refers to methods and mechanisms to *monitor and detect* changes as *monitoring mechanisms. Monitoring mechanisms are* deployed to monitor security property instances of web services at runtime.

4.2 Support for runtime security requirements

New security requirements may arise at runtime as security properties of participant web services change and new web services are added to the web services composition. When a participant web service has new security requirements at runtime, the web service is expected to update its security policy. If the security policy is updated, a compatibility check again the security policy of the web services composition is necessary to ensure that the web service still satisfies the security requirements of the web services composition. When selecting a new web service, the security policies of a set of candidate web services are matched against the security policy of the virtual wholesale web services composition, to identify a web service that satisfies the security requirements. Security policy matching and security policy compatibility checks require a vocabulary and a logical construction of security knowledge using relationships between classes and their subclasses in a way that supports formal reasoning (Cao et al. 2014). An *ontology* can provide such security vocabulary and knowledge and is therefore necessary to be able to support runtime security requirements.

4.3 Adaptation Mechanisms

When problematic security events are detected at runtime, it is important how well the web services composition *decides* to react and *acts* to satisfy security requirements. The ability to *decide* and to *act* constitute the web services compositions' *adaptation mechanisms*. Adaptation mechanisms include the definition and creation of a runtime *strategy* or *plan* of action that outlines a set of changes that need to be applied to the existing security configuration of the web services composition.

5. Related work on adaptation requirements

Research on adaptation requirements is an active field of research. The next sections briefly reviews current literature on monitoring mechanisms, support for runtime requirements and adaptation mechanisms in order to motivate the framework defined by this research. Finally, a gap analysis is given.

5.1 Monitoring mechanisms

The monitoring web services compositions to detect security violations is an active research area. Khaxar and Jalili (2012) propose that web services composition monitoring specifications can automatically be generated from descriptions of the web services composition. The challenge is that such specifications cannot be used to monitor choreographed services. Asim et al (2014) defines a monitoring framework that allows different user-specified policies to be monitored simultaneously. The challenge of this approach is that policies are defined in ConSpec language, which does not have clean semantics and is not a standardized security policy language.

5.2 Support for runtime security requirements

Methods to annotate security requirements on models has been explored by Menzel et al (2010) but has not yet been extended to capture runtime security requirements for a services composition. Aspect oriented technics to enforce security requirements have been explored by Yahyaoui et al (2012) and Almorsy et al (2014), however WS-SecurityPolicy based models are not considered. Various policy language have been proposed to define security policies by research such as Shah et al (2014). The main challenge identified is how to translate security models from one language to another.

5.3 Adaptation mechanisms

Research to address the challenge of adaptation mechanisms have focused on service discoverability, matching and selection of services. Chhun et al (2014) have attempted to solve the challenge of discovering web services, but they do not address searching by incorporating web services security policies. Cao et al (2014) proposes to match security capabilities as part of the process of selecting services but the automatic transformation of WS-Policy to an ontology model is not addressed. The concept of Software Product Lines (SPL) to define dynamic adaptation of services compositions have been adopted by Alférez et al (2014), however this approach does not consider the security of a web services composition.

5.4 Gap Analysis

A gap analysis on current literature presented here reveals the following:

- Proposed models used to describe the security context of a web services composition do not sufficiently support dynamic security contexts. This limits the ability to map runtime security requirements to flexible runtime security patterns using security policies.
- There is lack of concrete mechanisms to automatically generate and enforce a security adaptation plan which is necessary for secure dynamic adaptable compositions.

In order to create secure adaptable composition, the above two gaps should be resolved. Currently there is no complete solution that holistically address the challenge of engineering secure adaptable web services composition. In the next section, a high level framework for dynamic adaptation of secure web services composition is presented.

6. Framework for dynamic adaptation of secure compositions

The premise of the framework proposed here is based on the observation that the security of a web services composition can be considered as a set of security properties and their enforcement mechanisms that offer a certain level of security that should be preserved at runtime. Adapting a web services composition to preserve a certain level of security can be viewed as:

- Dynamically replacing a web service, where its security mechanisms that enforce a security property have changed in such a way, that the level of protection provided by the web service no longer satisfies the security requirements of the web services composition.
- Dynamically configure new security mechanisms for the web service, to enforce a security property, in order to continue using the web service in the web services composition.

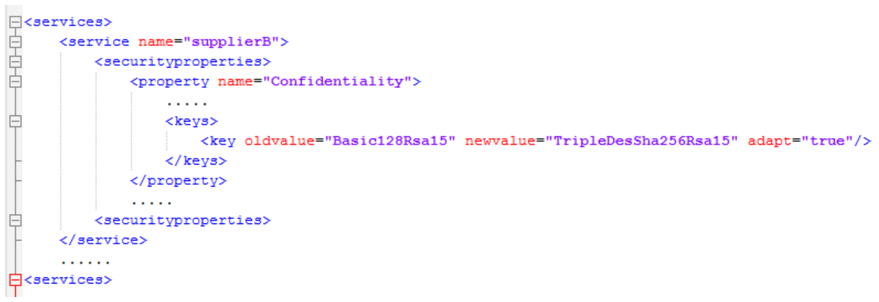
To be able to address these features, various models are created by the proposed framework to provide abstractions of the security of the web services composition and the composition runtime behavior. These models are defined in six steps, discussed next.

Step 1: Definition of the web services composition model

To be able to analyze and understand the security of a web services composition, a WS-BPEL model is first defined to abstract the web services composition as logical units of behavior, as specified by functional requirements. WS-BPEL is chosen because it is an XML-based industry standard language that provides a notation and semantics to specify business process behavior based on web services**.** WS-BPEL orchestrates web services by assigning responsibilities and sequencing calls to web services to create a conversation flow chart based on message exchanges. Exchanged messages are described in the WSDL**.** The web services composition model links all web services that the composition interacts with *partner links*. The wholesaler web services composition can then be viewed as a WS-BPEL model.

Step 2: Definition of a security context model

Next, a WS-SecurityPolicy based policy is defined to capture security requirements. WS-SecurityPolicy is a widely accepted security standard that describes a set of rules used to define security objectives to be satisfied when web services interact. A challenge encountered when comparing WS-SecurityPolicy based policies is that WS-SecurityPolicy only supports the syntactic matching of policies, which depend on the policy intersection mechanisms provided by WS-Policy standard. Syntactically different security assertions with the same meaning are found incompatible, which hinders compatibility checks between security assertions. This limitation can be addressed by extending WS-Policy with ontology classes defined in Web Ontology Language (OWL) (www.w3.org/2004/OWL). OWL is chosen because it is W3C recommendation, is expressive and support formal reasoning. Changes that occur on security policies of participating web services are detected by converting security policies of web services into an OWL ontology at runtime and then formal reasoning is performed on the semantically enriched policies. When the changes requires the web services composition to be reconfigured or adapted, an XML-based file which contain information about security property changes is generated. Figure 4 below shows a snippet of a file generated when SupplierB updates the *AlgorithmSuite* elements on its security policy from *Basic128Rsa15* to *TripleDesSha256Rsa15*. In this instance, SupplierB still satisfies the web services composition security requirements as the supported level of security is similar. The web services composition needs to ensure that a new key is used when communicating with SupplierB. We will refer to this file as the *security properties changes* file.



```
<services>
    <service name="supplierB">
        <securityproperties>
            <property name="Confidentiality">
                .....
                <keys>
                    <key oldvalue="Basic128Rsa15" newvalue="TripleDesSha256Rsa15" adapt="true"/>
                </keys>
            </property>
            .....
        <securityproperties>
    </service>
    ......
<services>
```

Figure 4: Security properties changes

Step 3: Creation of a web service security variability model

In order to capture the lifecycle of security properties defined for a participating web services, a model needs to be defined. For example, the general concept of access control, defined for the retailer web service in the virtual wholesaler composition, can be implemented by authentication and authorization mechanisms using username tokens or digital certificates. These two options can be considered as *variants* or *variation points* of the access control property and may require a different application programming interface to enforce. When enforcement mechanisms of a security property changes at runtime, the dynamic adaptation of the virtual wholesaler web services composition can be described in terms of the activation or deactivation of security property enforcement mechanisms.

The existence of *variations* to enforce security properties of a web service makes it possible to use software engineering approaches that have been successfully used to design *variants* at design time and then assigning concrete implementation to *variants* at runtime. Dynamic Software Product Lines (DSPLs) manages both *variants* and dynamic adaptation (Park et al. 2014) and is therefore chosen in the framework as a technology to model security properties of composed services. Using DSPL, security properties such as authentication, authorization, authenticity, confidentiality and integrity are expressed using the Common Variability Language (CVL). A tool for CVL is available as an Eclipse IDE plugin (Haugen et al. 2012). Using CVL, a variation specification (VSpec) tree structure of the security of a participating web service is decomposed into access control, data security and message security as shown in Figure 5, adapted from Horcas et al (2014). Security properties are thus decomposed into an available set of enforcement mechanisms. The security configuration of a security property can therefore be defined as a selection of a set of choices on the VSpec tree structure model. The tree structure model is referred to as the *security variability model* in this framework.

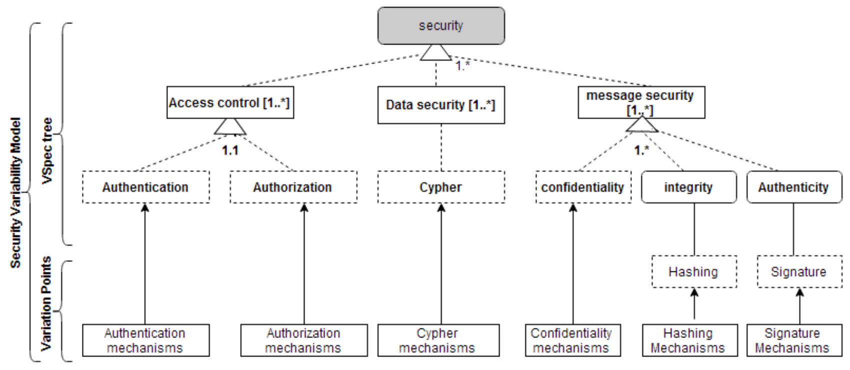


Figure 5. Security Variability Model

The low-level enforcement mechanisms are the *variations points* on the *security variability model*. *Variation points* express decisions leading to different methods and algorithms chosen at runtime to enforce security properties. When changes for example on the confidentiality security property depicted in Figure 4 are detected at runtime, the composition is reconfigured by re-binding the confidentiality *variation points* on the *security variability model* with a new enforcement mechanisms of the new confidentiality requirements.

Generally, functionality to enforce security properties at runtime on a WS-BPEL engine is defined in an external security library and is made accessible using an applicable programming interface (API), a reusable component. For example, on Apache Orchestration Director Engine (ODE), Rampart, an Axis 2 security module is used (http://axis.apache.org/axis2/java/rampart). The information about such modules and the API they offer to provide low-level implementation of security property is captured in a model referred to as the *security base model* in the proposed Framework. Figure 6 below shows the *security base model*. Using confidentiality component as an example, the component contains the actual programming logic to encrypt message using a given algorithm suite to ensure message protection.

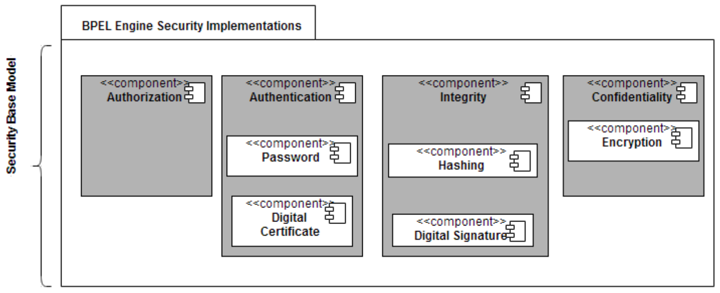

BPEL Engine Security Implementations
Security Base Model
<<component>> Authorization
<<component>> Authentication
<<component>> Password
<<component>> Digital Certificate
<<component>> Integrity
<<component>> Hashing
<<component>> Digital Signature
<<component>> Confidentiality
<<component>> Encryption

Figure 6: Security base model

Step 4: Generation of security reconfiguration model

One of the advantages of CVL is that it is executable. This makes it possible to automatically generate new models from an existing model. The *security variability model* provide information about security properties and the *security base model* provide information about security components that enforce the security properties. By executing the *security base model* and *security variable model* together and incorporating the information from the *security properties changes stored in the file* created in step 2, a *security reconfiguration model* is generated. Figure 7 below shows the process to generate the security reconfiguration plan using model-to-model transformation techniques by delegating to an Atlas Transformation Language (ATL) engine (Jouault & Kurtev, 2006).

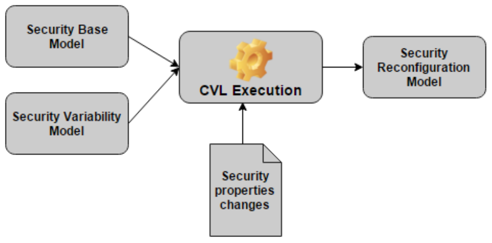

Security Base Model
Security Variability Model
CVL Execution
Security Reconfiguration Model
Security properties changes

Figure 7: Security reconfiguration model generation

The generated *security reconfiguration model* provides information that is used to adapt WS-BPEL based process of a web services composition. If a web service is to be replaced, the *security reconfiguration* *model* will contain the applicable WS-BPEL fragments specifying the selected *partner link* to be replaced and the context conditions which specifies the conditions that determine the instantiation of the WS-BPEL fragments to be added. Use of the security configuration model to adapt the composition is discussed in the next step.

Step 5: Creation of security adaptation plan

A plan of action referred to as the *security adaptation plan* is required to adapt a WS-BPEL web services composition. The security adaptation plan defines actions that need to be performed to transition a WS-BPEL process configuration to a new configuration that guarantees security. The initial step to prepare a WS-BPEL process is to use CVL to augment WS-BPEL processes with variability so that the concept of a DSPL is used. The process variables, activities and partner links necessary to support the mandatory business process features define the WS-BPEL process *base model*. The partner links elements on the business process represents the BPEL *variability model.* Adaptation of the WS-BPEL process can then be viewed as substitution of WS-BPEL process elements. By executing the *WS-BPEL base model* and *WS-BPEL variability model* and incorporating the information from the *security reconfiguration model* created in the previous step, a new WS-BPEL model referred as the *resolved WS-BPEL model* is generated. The *resolved WS-BPEL model* represent the expected WS-BPEL process model after adaptation.

Once the *resolved WS-BPEL model* is generated, the *security adaptation plan* is calculated by applying a difference operator between the model of the currently active WS-BPEL web services composition and the *resolved WS-BPEL model* using the Atlas Model Weaver by applying the approach defined by Del Fabro et al (2006). At the implementation level, model differences are changes in the enforcement mechanisms of the security properties. When the action needed to adapt the composition requires replacement of a web service, the *security adaptation plan* contains information that refers to a *partner link*s in the BPEL process base model where new *partner links* are added to replace an existing process element or fragments. When the action needed to adapt the composition require dynamically configuring new mechanisms to enforce security properties, aspects are defined and populated with information necessary to intercept methods on the respective security component on the *security base model*.

Step 6: Enforcement of the adaptation plan on the WS-BPEL

Replacing a web service in a composition at runtime is not a trivial process because it requires adapting WS-BPEL process at runtime. Blocking of any task on the process to apply changes should not interrupt any running conversations, unless all internal state variables are in a consistent state and the necessary transactions scope that encapsulate the service are completed. The framework adopts DyBPEL, a tool that compliments activeBPEL execution engine with runtime adaptation capabilities (Baresi et al. 2012). Using the *adaptation plan* created in step 5, a new process definition is created at runtime by replacing the *partner links* with new partner links. This is possible since BPEL specifications are XML documents and XPath is a natural choice to identify *partner links* on the BPEL process. After creating a new process definition, the process is deployed and DyBPEL handles the WS-BPEL process migration.

When the web services composition is required to dynamically configure new mechanisms to enforce security properties, Aspect-Oriented Programming (AOP) is used to enforce the adaption plan. AOP is chosen because it was designed explicitly to address the modularization of crosscutting concerns like security. AOP introduces a unit of modularity, called an *aspect*, which can be used to modularize the security properties using *join points, pointcuts* and *advices*. *Join points* is a method defined on the security component in the *security base model* where the aspect will be weaved. A set of *join points* create a *pointcut*. An *advice* is a piece of code that is executed whenever a *join point* is reached. *Advice, joint* *points* and *pointscut* are contained in the adaptation model created in the previous section. To enforce new mechanisms when the BPEL composition invokes a composed service, a SOAP message being sent to a composed service is intercepted by the external security module. Using aspects, security configuration is weaved when the methods that provide security functionality are executed.

7. Conclusion

This paper proposed a high-level framework to engineer dynamic adaption of a web services composition within its security context. The framework uses web services composition models created at design time to abstract the security of a web services composition and to provide security meta-information at runtime. The security meta-information provides rich semantic information to monitor and reason about runtime security changes and is used by the framework to facilitate better web services composition adaptation using an automatically generated adaptation strategy. Mechanisms to transform the composition by weaving new security configuration into the web services composition has also been proposed. Unlike most existing approaches that focus on capturing and enforcing security requirements at design time, the proposed approach contributes a framework that pays attention to how a web services composition could dynamically adapt to cope with unanticipated security requirements that may arise at runtime. Future work will focus on ensuring that the adaptation strategy is coordinated across the web services interface and other layers of the application so that changes on one layer does not affect the security of other layers. Finally, the framework will be implemented in a prototype to determine how viable the framework is.

References

Alferez, G.H. and Pelechano, V (2013) “Facing uncertainty in web service compositions” *Web Services (ICWS), IEEE 20th International Conference on,* pp. 219-226

Alférez, G.H., Pelechano, V., Mazo, R., Salinesi, C. and Diaz, D (2014) “Dynamic adaptation of service compositions with variability models”, *Journal of Systems and Software,* pp.24-47.

Almorsy, M., Grundy, J. and Ibrahim, A.S (2014) “Adaptable, model-driven security engineering for SaaS cloud-based applications” *Automated Software Engineering, 21*(2), pp.187-224.

Asim, M., Zhou, B., Llewellyn-Jones, D., Shi, Q. and Merabti, M (2014) “Dynamic Monitoring of Composed Services” *Cyberpatterns, Springer International Publishing,* pp. 235-245

Avanesov, T., Chevalier, Y., Mekki, M.A., Rusinowitch, M. and Turuani, M (2012) “Distributed orchestration of web services under security constraints”, *Data Privacy Management and Autonomous Spontaneus Security, Springer Berlin Heidelberg,* pp. 235-252

Bocciarelli, P. and D'Ambrogio, A (2011) “A BPMN extension for modeling nonfunctional properties of business processes”, *Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium,* Society for Computer Simulation International*, Society for Computer Simulation International,* pp. 160-168

Cao, Tuan-Dung, and Nguyen-Ban Tran. "Enhance Matching Web Service Security Policies with Semantic." In *Knowledge and Systems Engineering,* pp. 213-224. Springer International Publishing, 2014.

Chhun, S., Moalla, N. and Ouzrout, Y (2014) “QoS ontology for service selection and reuse”, *Journal of Intelligent Manufacturing,* pp.1-13.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N. and Weerawarana, S (2002) “Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI””, *IEEE Internet computing,* (2), pp.86-93.

Del Fabro, M.D., Bézivin, J. and Valduriez, P (2006) “Weaving Models with the Eclipse AMW plugin” *Eclipse Modeling Symposium, Eclipse Summit Europe* (Vol. 2006)

Erl, T., 2008. *Soa: principles of service design* (Vol. 1). Upper Saddle River: Prentice Hall.

Hoque, S., Rahim, A., Llewellyn-Jones, D. and Merabti, M (2013) “Security Property Lifecycle Management for Secure Service Compositions” In *Cyber Security and Privacy, Springer Berlin Heidelberg,* pp. 67-78

Horcas, J.M., Pinto, M. and Fuentes, L (2014) “Runtime Enforcement of Dynamic Security Policies” In *Software Architecture Springer International Publishing,* pp. 340-356

Immonen, A. and Pakkala, D (2014) “A survey of methods and approaches for reliable dynamic service compositions” *Service Oriented Computing and Applications*, *8*(2), pp.129-158

Jouault, F. and Kurtev, I (2006) “Transforming models with ATL”, S*atellite events at the MoDELS 2005 Conference, Springer Berlin Heidelberg,*pp. 128-138.

Kephart, J., Kephart, J., Chess, D., Boutilier, C., Das, R., Kephart, J.O. and Walsh, W.E (2007) “An architectural blueprint for autonomic computing” *IEEE internet computing*, *18*(21).

Khaxar, M. and Jalili, S (2012) “WSCMon: runtime monitoring of web service orchestration based on refinement checking”, *Service Oriented Computing and Applications*, *6*(1), pp.33-49

Menzel, M., Warschofsky, R. and Meinel, C (2010) “A pattern-driven generation of security policies for service-oriented architectures”, *Web Services (ICWS), IEEE International Conference,* pp. 243-250

Rosenberg, J. and Remy, D (2004) “*Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption”,* Pearson Higher Education.

Shah, S.Y. and Szymanski, B.K (2014) “Dynamic policy enforcement using restriction set theoretic expressions (rste)”, *Military Communications Conference (MILCOM), IEEE,* pp. 198-203

Song, W., Ma, X., Cheung, S.C., Hu, H., Yang, Q. and Lu, J (2011) “Refactoring and publishing WS-BPEL processes to obtain more partners” *Web Services (ICWS), IEEE International Conference,* pp. 129-136

Tiwari, S. and Singh, P (2011) “Survey of potential attacks on web services and web service compositions”, *Electronics Computer Technology (ICECT), 2011 3rd International Conference on* (Vol. 2, pp. 47-51). IEEE.

Xiao, Z., Cao, D., You, C. and Mei, H (2011) “Towards a constraint-based framework for dynamic business process adaptation” *Services Computing (SCC), IEEE International Conference,* pp. 685-692

Yahyaoui, H., Mourad, A., Almulla, M., Yao, L. and Sheng, Q.Z (2012) “A synergy between context-aware policies and AOP to achieve highly adaptable Web services”, *Service Oriented Computing and Applications*, *6*(4), pp.379-392

Zhou, B., Llewellyn-Jones, D., Shi, Q., Asim, M., Merabti, M. and Lamb, D (2012) “Secure service composition adaptation based on simulated annealing”, *6 th Layered Assurance Workshop* pp. 49