ASPECTUAL PATTERNS FOR WEB SERVICES ADAPTATION

Najme Abbasi Tehrani and Afshin Salajegheh

Department of Computer Engineering, South Tehran Branch,
Islamic Azad University, Tehran, Iran

ABSTRACT

The security policies of an application can change at runtime for some reasons such as the changes on the user preferences, the performance reasons or the negotiation of security levels between the interacting parties. If these security policies are embedded in the services, their modifications require to modify the services, stop and deploy new version. Aspect oriented paradigm provides the possibility to define separated components that is named aspect. In this paper, in order to fulfill security requirements, we will classify required changes of services and for each classifications, how aspects injection will be described. Finally, we will present a pattern for each aspect of each classification.

KEYWORDS

Web Service adaptation, security patterns, aspect oriented programming.

1. INTRODUCTION

Security is considered to be one of the main challenges as regards the widespread application of Service Oriented Architectures across organizations[1]. Software adaptation is a sound solution to overcome the inconsistencies in interface, behavior and security constraints among different services. Inconsistencies of security constraints may be related to difference between provided security level by service provider and required security level of service consumer. Also it may be related to change of management security policies. On the other hand, new attacks may be detected in run time that is not predicted in design time. In order for adaption of provided security level and required security level, services should be changed.

If these security policies are embedded in the services, their modifications require to modify the services, stop and deploy new version. Aspect oriented paradigm provides the possibility to define separated components that is named aspect. Once specified, a security policy is enforced through the deployment of certain security aspects within the application. In this purpose, it's needed to extract security requirements of services and services be faced to their main concerns. Aspect oriented is a tool for concerns separation. By applying the aspect oriented concepts, architectures can prevent of leakage of cross-cutting concerns –e.g. security concerns- in the services.

DOI : 10.5121/ijwsc.2015.6301

In this paper, in order to fulfill security requirements, we aim to classify required changes of services. To realize these changes, for each classification, a method of aspects injection will be described. Finally, we will present a pattern to aspect generation of each classification. Each pattern contains of a set of <point-cut, advice> pairs that detect when and how adaptation be done. In this approach, point-cuts are in the form of queries that check conditions of advice realization.

The classifications and patterns can make easier aspect generations and injections for the service changes. Because of the overhead of aspects checking, by applying the suggested approach, response time of the services will be increased and consequently the performance will be decreased.

The rest of the paper is organized as follows. In Section 2, we discuss the changes in order to modify security level of the services. We characterize these changes in three categories. For each categories in Section 3 we present the some patterns for the aspects that should be injected in to enforce these changes. Related work is discussed in section 4, and Section 5 summarizes our work.

2. CHANGES OF SERVICES IN ORDER TO MODIFY SECURITY LEVEL

In according to the security vulnerabilities and considered security policies in the service oriented environments, it's considered the specified security levels for a system. These security levels may be changed for some reasons such as the changes on the user preferences, the performance reasons or the negotiation of security levels between the interacting parties. These changes can be led tothe reduction or increase security controls. Supplying of requirements in security adaptation, needs to some changes on adaptive system.

We characterize these changes in three categories:

- Replacing a service with the other, in the special conditions
- Processing on input/output parameters, e.g., encryption/decryption of them and limitation of output parameters, in the special conditions
- Related changes to the number of parameters

In the rest of this section, each category will be explained and the method of enforcing of changes by the aspects will be presented.

2.1 Replacing a service with the other

In order to fulfill security requirements, it may be needed to represent a service instead of a requested service. As an example, consider a service provider that presents all the services on the base of trust and does not enforce any security limitation. After some time, managers decide to impose some limitations just for a special service consumer and do not present service unlimitedly. For this purpose, it is decided to provide another service without consumer's

understanding. This applied strategy for these changes is substitution strategy. Fig .1 presents how replacing service by aspect injection.

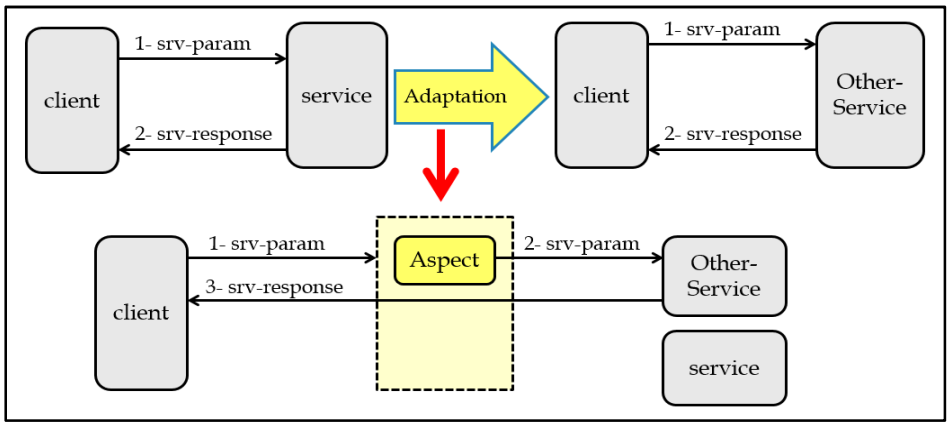

Fig.1 Aspect injection for replacing requested service to another one

As shown in Fig. 1, for replacing the requested service with another one, an aspect has been injected. Considering the mentioned example, before receiving request by the target service, the injected aspect monitors whether the consumer is the expected consumer. If the condition was satisfied, the aspect redirects request to another service without the consumer is noticed.

2.2 Processing on input/output parameters

Adapting system with the provided security requirements may be needed to process on input/output parameters. As an example, suppose a service that presents the output without any encryption. After some time, a consumer wants to use this service. But this consumer is not in the secure connecting network with the provider and requires to get result of service, as encrypted. While presenting service to the old consumers should not be changed, this requirement should be supplied by the provider. Applied strategy for these kinds of changes is re-composition strategy. Fig .2 presents how processing output parameter by aspect injection.

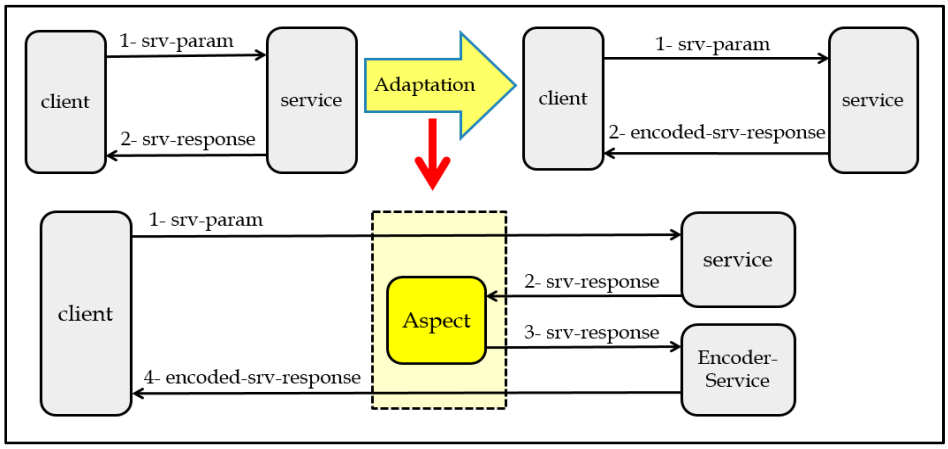

Fig. 2Aspect injection to process output parameter service

Considering the mentioned example, as shown in Fig. 2, before receiving the result of service by the consumer, an aspect has been injected that monitors whether the current consumer is the expected consumer or not. If the expected condition is satisfied, the aspect sends the output of service to the Encoder-Service. The output of the Encoder-Service is returned to the consumer, as the result of the target service.

On the other hand, provided security requirements may be led to processing input parameters. For this purpose, an aspect is injected before receiving input parameter by the service. To process input parameter, this aspect redirects parameters to the other service, e.g., Decoder-Service. After returning result of processor service, another aspect gets this result and sends it to the target service. Fig. 3 shows an example to implement this kind of requirements by the aspect injection.

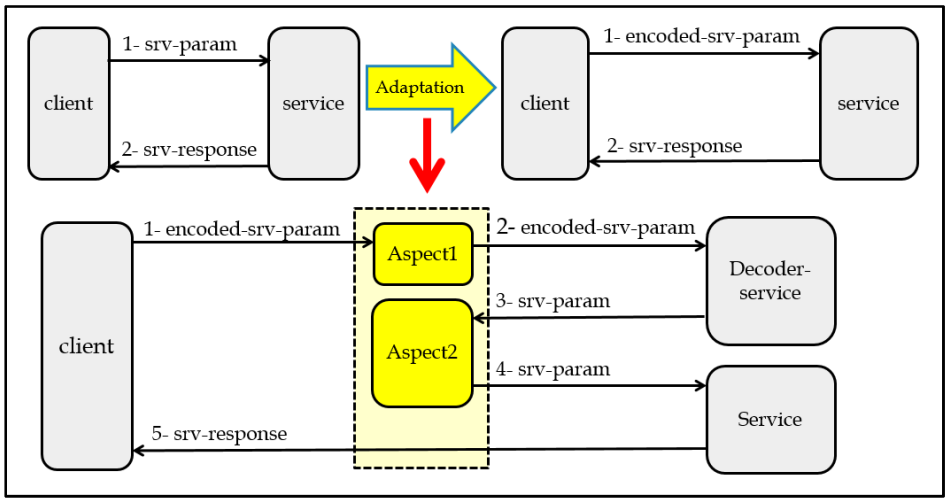

Fig.3Aspect injection to process input parameter service

2.3 Related changes to the number of parameters

Adaptation requests to change security level in a service oriented system may be required to change the number of exchanged parameters between service provider and service consumer. These changes may be on the purpose of increasing or decreasing the number of input/output parameters.

- **Increasing the number of input/output parameters**

In some cases, the service consumer wants to send some parameters that current service does not require to receive. In other word, some input parameters exist in the target service that do not exist in the current service. On the other hand, sometimes, some values exist in the target service response that is not in the current service response.

Fig. 4 presents an example to implement increasing input parameters by the aspect injection. In this example, the consumer requires to send a ticket in addition to parameters of current service. For this purpose, an aspect has been injected before receiving parameters by the service. This aspect sends a ticket to the “Ticket Validator” service. After returning response from “Ticket

Validator" service, another aspect has been injected. This aspect checks the response and if the response of ticket validation is ok, it invokes the main service by the required parameters.

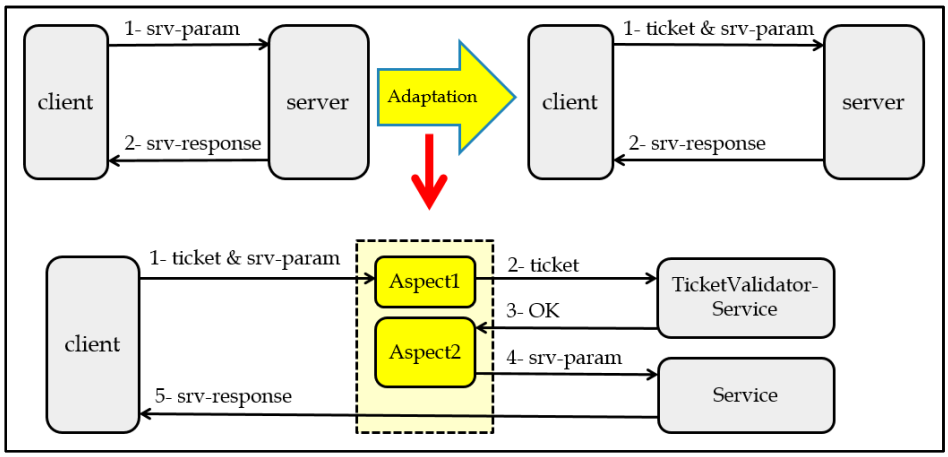

Fig. 4Aspect injection to increase input parameters

On the other hand, the result of analysis of an adaptation requirement may be led to increasing the output parameters. Fig. 5 presents an example to show how aspect injection to fulfill these kinds of requirements. For the purpose of inserting sign parameter in the output of service, an aspect has been injected after returning the response of service. This aspect sends the result of service to the "Sign Generator" service.

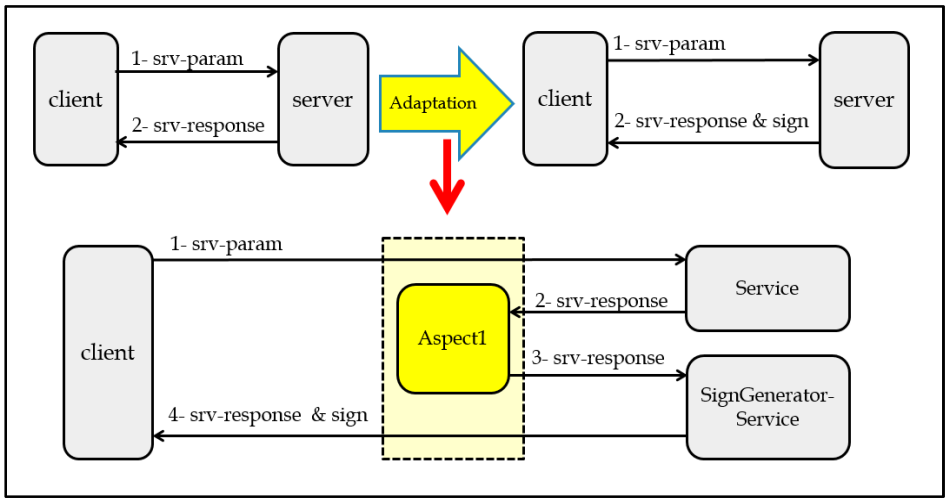

Fig. 5Aspect injection to increase output parameters

The "Sign Generator" service generates a digital sign from output service and sends it in addition to result of the service to the service consumer.

- **Decreasing the number of input/output parameters**

For some reasons, sometimes a consumer might not be determined to send some parameters that the current service requires to receive. In other word, some input parameters can't be provided by a consumer. On the other hand, sometimes, some values exist in the response of current service that should not be sent to the consumer.

Fig. 6 presents an example to implement decreasing the number of input parameters by the aspect injection. In this example, current service requires to receive a ticket in addition to the other parameters while consumer can't provide the ticket. In order to provide the ticket for sending to the service, an aspect has been injected before receiving parameters by the service. This aspect sends input parameters to the "Ticket Provider" service. After returning response from the "Ticket Provider" service, another aspect has been injected. This aspect send sticket and the other required parameters to the considered service.

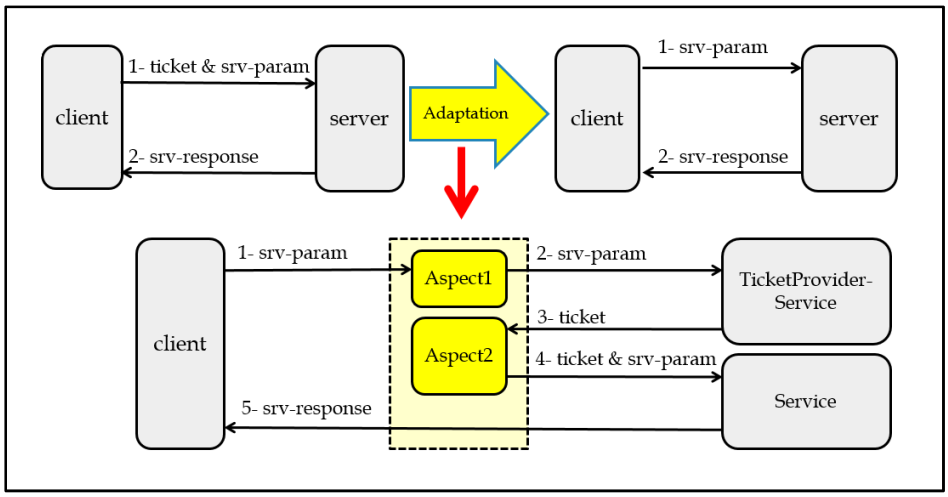

Fig. 6Aspect injection to decrease the number of input parameters

3. ASPECTUAL PATTERNS IN ORDER TO SECURITY LEVEL MODIFICATION

We have been offered a pattern, for each of the injected aspects in the determined categories. Each patterns is specified by a collection of <query, advice> pairs. When an aspectual pattern is instantiated, a collection of adaptation aspects that will be weaved into the service at the runtime. An advice determines which actions should be done to realize adaptation. The query part defines the conditions for triggering an advice. In general, by considering the detected categories, all the aspects will be injected before or after service invocation in order to modify security level. Due to space limitation, below we only discuss three aspectual patterns in detail:

Pattern 1: For replacing a service with another one, the injected aspects are followed by this pattern.

Query	Advice
Before service Invocation When <condition>	invoke specified Service(srv-param)

This aspect always, checks the determined conditions before every service invocation and when these conditions are satisfied, the specified service will be invoked, instead of main service.

Pattern 2: The required aspects for processing input parameters will be followed by this pattern.

Query	Advice
Before service Invocation When <condition>	Write target ServiceName; Write srv-param; Invoke specified Service(srv-param);
After service Response When <condition>	Read target Service Name; Read srv-param; Invoke target Service Name (srv-param/srv-response);

As described before, two aspects are required to weave into the adaptive system. First aspect, always, checks the determined conditions before every service invocation. If the conditions are satisfied, the specified service will be invoked after writing the specification of service invocation, such as target service name and input parameters.

After returning the result of the specified service, by the second aspect, the specification of the target service will be retrieved and the target service will be invoked, if the determined conditions are satisfied.

Pattern 3: The required aspect for processing output parameters will be followed by this pattern.

Query	Advice
After service Response When <condition>	Invoke specified Service(srv-response);

The determined conditions will be always checked by this aspect after every service invocation. When the conditions are satisfied, the specified service will be invoked, and the result of the service will be returned to the service consumer instead of the target service.
The patterns of the other specified categories, are almost similar to the described patterns and due to space limitation we have ignored presenting them.

4. RELATED WORKS

Web services security is one of the main challenges that attracted the attention of the research community. From definition of standards to the publication of research papers, the goal is to provide policies and mechanisms for enforcing web services security [2] .In this context, several standards such as Security Assertion Markup language (SAML)[3], WS-Security [4] and WS-XACML [5]were proposed.

Moradian et al. [6]presented, Web Services security and security concerns together with analysis of possible attacks on SOAP implementation of XML Web Services over HTTP.

Bhuyant al. [7]addressed choreography security issues while passing messages between the services choreographed in SOA, They also proposed a Verification Model (VM) using Security Assertions Markup Language (SAML 2.0) to provide authentication and authorization.

J.A. Martin et al. [1]proposed to use security adaptation contracts that allow to express and adapt the security requirements of the services and their or chest ration. Security adaptation contracts enable to specify how to adapt signature, behavior and security incompatibilities among services; they describe the security checks that must be performed over the received messages; and, due to their centric role in the conversation, they provide a formal framework to analyze the behavior of the system.

In order to modularize nonfunctional concerns, e.g., logging and security of BPEL processes, in [8] Charfi and Mezini, propose to use AOP for defining workflow Processes, for the first time and present a framework as AO4BPEL. This framework can weave aspects with business process at runtime.

Mourad et al. [2]proposed an approach for the dynamic enforcement of web services security, which is based on a synergy between Aspect-Oriented Programming (AOP) and composition of Web services. They specified the security policies as aspects that are weaved in BPEL process at runtime.

Wohlstadter and Volder[9] intercept and parse the content of SOAP messages to identify pointcuts in to apply advices that handle document-oriented concerns, e.g., encryption or schema transformation. This work focuses on message-level processing. Similarly we have focused on modifying security level by aspect injection on message-flow transformations on the application-level.

Bertino et al.[10] proposed a RBAC-WS-BPEL framework for defining authorization policies and constraints for WSBPEL business processes. They introduced the Business Process Constraint Language (BPCL), which can be used to specify authorization constraints. The users are associated with roles as done in Role Based Access Control (RBAC) models. The main problem with the proposed solutions for the enforcement of Web services security is the static embedding of the security features in the design/code of the Web services. In fact, many security features require run-time verification of the security policies, which may often be modified and updated. Such a mechanism is cumbersome, error-prone and tedious. Our approach relies on the dynamic injection of AOP aspects between service provider and consumer. This allows to easily update the security measures when needed, without modifying the services.

Horcas et al.[11] presented the design, implementation and evaluation of a runtime security adaptation service. This service is based on the combination of autonomic computing and aspect-oriented programming, where the security functionalities are implemented as aspects that are dynamically configured, deployed or un-deployed by generating and executing a security adaptation plan. This service is part of the INTER-TRUST framework, a complete solution for the definition, negotiation and run-time enforcement of security policies.

Kongdenfha et al. [12]proposed an Aspect oriented framework as a solution to provide support for service adaptation. This framework consists of i) a taxonomy of the different possible types of mismatch between external specification and service implementation, ii) a repository of aspect-based patterns to automate the task of handling mismatches, and iii) a tool to support template instantiation and their execution together with the service implementation. After that, they [13]characterized the problem of Web services adaptation focusing on business interfaces and protocols adapters. They introduced mismatch patterns completely to capture these recurring differences and to provide solutions to resolve them. They leveraged mismatch patterns for service adaptation with two approaches: by developing stand-alone adapters and via service modification and digged into the notion of adaptation aspects that, following aspect-oriented programming paradigm and service modification approach, allowing for rapid development of adapters. Their proposed approach was implemented in a proof-of-concept prototype tool, and evaluated using both qualitative and quantitative methods. Similarly we have presented some patterns for the aspects that should be injected in order to satisfy security requirements. Differently instead of BPEL processes we have focused on any message flow transformation between service provider and service consumer.

5. CONCLUSION

In this paper we used AOP for service adaptation to solve security level mismatches. For this purpose, we extract the security aspects as some cross-cutting concerns. This modularization facilitates the maintenance of the services and every service only evolves its main functionality. Our work can respond to modify security requirements at run time.

We classified the required changes of services, to satisfy security requirements. For each classification, the method of aspect injection was described. However, generally more than one changes may be needed at the same time, which makes service adaptation more complicated. These presented classifications and patterns can make easier aspect generations and injections for these changes.

Because of the overhead of aspects checking, by applying the suggested approach, response time of the services will be increased and consequently the performance will be decreased.

In the future, in addition to presenting a semi-automated aspect generation using the provided patterns, we plan to present an optimized approach for injecting these aspects in the service oriented environment, in order to satisfy the security requirements.

REFERENCES

[1] J.A. Martina and E. Pimentela, (2011) "Contracts for Security Adaptation," Logic and Algebraic Programming, vol. 80, no. 3-5, pp. 154-179.

[2] A. Mourad, S. Ayoubi, H. Yahyaoui and H. Otrok, (2010) "New approach for the dynamic enforcement of Web services security," Ottawa, ON.

[3] N. Klingenstein and T. Hardjono, "OASIS Security Services (SAML) TC," https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

[4] K. Lawrence and C. Kaler, "Web Services Security," https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

[5] B. Parducci and H. Lockhart, "OASIS eXtensible Access Control Markup Language (XACML) TC," https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[6] E. Moradian and A. Hakansson, (2006) "Possible attacks on XML Web Services," International Journal of Computer Science and Network Security, vol. 6.

[7] P. Bhuyan, A. Ray and D. P. Mohapatra, (2014) "A Service-Oriented Architecture (SOA) Framework Component for Verification of Choreography," in Springer India.

[8] A. Charfi and M. Mezini, (2007) "AO4BPEL: An Aspect-oriented Extension to BPEL," vol. 10, no. 3.

[9] E. Wohlstadter and K. De Volder, (2006) "Doxpects: Aspects Supporting XML Transformation Interfaces," in ACM, New York.

[10] F. Paci, E. Bertino and J. Crampton, (2008) "An Access-Control Framework for WS-BPEL," International Journal of Web Services Research(IJWSR), vol. 5, pp. 20-43.

[11] J.-M. Horcas, M. Pinto and L. Fuentes, (2014) "Runtime Enforcement of Dynamic Security Policies," in Springer International Publishing, Vienna, Austria.

[12] W. Kongdenfha, R. Saint-Paul and B. Benatallah, (2006) An Aspect-Oriented Framework for Service Adaptation, vol. 4294, Springer Link.

[13] W. Kongdenfha, H. R. Motahari Nezhad, B. Benatallah, F. Casati and R. Saint-Paul, (2009) "Mismatch Patterns and Adaptation Aspects: A Foundation for Rapid Development of Web Service Adapters," vol. 2, no. 2.