Situational Secure Web Services Design Methods

Dhafer Thabet, Lamia Hassine, Henda Ben Ghézala
GLOIR Laboratory, Computer Science National School, Tunis, 1004, Tunisia
dhafer_thabet@yahoo.fr, lamia.h@planet.tn, henda.bg@cck.rnu.tn

Abstract

Web Services security is a major concern in the web engineering domain. Several approaches emphasizing Secure Web Services Design (SWSD) process have been proposed. However, they provide a little guidance as to what developers should do exactly to conduct this highly intellectual process. SWSD practices cannot be defined independently of the situation in which they are applied. To address this challenge, we propose an approach for guiding the construction and execution of situational SWSD methods by reengineering and integrating different existing SWSD method components suited to the specific situation on hand.

1. Introduction

A Web Service is a software system conceived to support interoperable machine-to-machine interaction over a network [15]. Web services are thereby exposed to several types of threats and attacks. Despite the importance of security requirement to build efficient Web Services, there has been no efficient guidance as to what developers should do exactly to conduct the Secure Web Services Design (SWSD) process.

Several approaches, trying to ensure Secure Web Service development using a sequence of activities, have been proposed [1] [2] [3] [4] [9] [14]. However, such linear view does not reflect the real use of SWSD process. Due to the continuous increasing complexity of Security Web Services assurance, we should rather attempt to define the SWSD practices suited to the specific situation of the product being developed. To mitigate this problem, we propose an approach for guiding the developer in the construction and execution of Situational Secure Web Services Design Methods by selecting and integrating existing method components which are most suited to the specific situation on hand. We suggest the formalization of SWSD process using a dynamic and flexible process model built on the notion of a map and associated guidelines [5] [6] [12].

Our proposal for SWSD guidelines specification is the reengineering of existing SWSD methods which are useful in specific project situations. A method meta-model is proposed to help the reengineering of SWSD methods in order to produce modular SWSD method components called chunks [12].

This paper is organized as follows: section 2 presents existing works trying to take into account security factor during Web Service Design process. Section 3 highlights motivations for a situational modeling approach in SWSD process and defines SWSD process as a map. Section 4 describes a method meta-model allowing the definition of SWSD method chunks. Section 5 provides an example of SWSD method chunk: System Pattern Method Chunk and illustrates the construction and execution of SWSD situational methods suited to situations on hand. Finally, some conclusions are drawn in section 6.

2. Existing works

Several approaches have been proposed taking into account security factor during the development lifecycle. They are classified in two categories. The first one focuses on the design process whereas the second one proposes a whole development process approaching security at a very early stage of Web service development lifecycle. In this section, we present different approaches belonging to these two categories.

The security design patterns approach [1] proposes a set of security design patterns and a methodology for using them to design a secure system. Software designers can use this approach to produce a system architecture that meets their security requirements.

E. B. Fernández proposes in [9] two design patterns for web services security. The first pattern coordinates authentication and authorization using a role-based control model for access to distributed resources. The second pattern deals with filtering XML messages according to business access control policies.

A process for Web Services Security, called PWSSec, aiming to facilitate security integration into Web services based systems is proposed in the literature [2] [3] [4]. PWSSec is an incremental and iterative process focusing on three stages: WSSecReq (Web Services Security Requirements) [2], WSSecArch (Web Services Security Architecture) [3], [4] and WSSecTech (Web Services Security Technology) [2]. The security requirements of the Web
Service-based system to be constructed are specified in the WSSecReq stage. WSSecArch stage concerns the construction of the system security architecture satisfying the security requirements previously specified. The WSSecTech stage involves defining a set of standards that implement the ASecS identified in the WSSecArch stage.

An AGILE process of secure Web application development is presented in [14]. This process combines two methods. The first one is a general-purpose information system development method which is FDD (Feature-Driven Development). The second one is a security method based on risk analysis. The resulting process is incremental and iterative involving six activities: requirements analysis, security policy decision, use case analysis, content design, security risk analysis, and implementation.

Approaches presented in [2] [14] provide security solutions that cover the whole development lifecycle whereas the rest focus only on the design stage.

3. A situational view of SWSD process

SWSD practices cannot be defined independently of the situation on hand. The question is not just to propose a sequence of improvement actions, but rather to attempt to define SWSD practices suited to the requirements of specific situations.

3.1. Motivations for a situational modeling approach

To be implemented efficiently, the SWSD process specifies what activities to be performed and what they modify. The limited ability to specify the way-of-working that developers must follow is still more hindering in the SWSD process which is particularly characterized by its major importance to ensure secure Web Service development, its highly creative, non-deterministic, flexible and interactive. For such process, there are many ways of doing the same thing and the best way, the one most suited to the situation at hand, has to be found. In this context, alternative generation and selection are very crucial. Decisions concerning which practices to execute in order to secure a design component rely on the situation in which they are applied. Such decisions enable a better correlation between the way-of-working to be followed and the required Web Service security, and thus enhance the efficiency of the Secure Web Service development process.

The guidance capability of a way-of-working specification depends highly on its underlying process approach. The existing process models can be classified in four categories [5] [6]: activity-oriented models, product-oriented models, decision-oriented models and contextual process models. In activity-oriented process models, the process is defined as a related set of activities conducted for the specific purpose of product definition. The product-oriented process models also focus on the activities but have the advantage of linking the activities to their output. In the decision-oriented category, the successive transformations of the product are looked upon as consequences of decisions. Finally, a more recent category of process models follows a contextual paradigm. Here, a process is defined as a sequence of contexts causing successive product transformations under the influence of a decision taken in a specific situation. Considering the characteristics of the SWSD process presented above, we deduce that the contextual approach fit the most to the process modeling goals.

3.2. The concept of a map

The proposed modeling approach is composed of a map and associated guidelines (Fig. 1). To represent it graphically, we use UML formalism. The map is a navigational structure which provides the dynamic selection of the intention to be reached next and the appropriate strategy to reach it.

The map is a labeled directed graph with intentions as nodes and strategies as edges. The directed nature of the graph indicates which intentions can follow which. An intention is a goal that can be achieved by the performance of the process. It is expressed as a natural language statement consisting in a verb and several parameters. A strategy is an approach, a manner to achieve an intention [7].

A Guideline aims at supporting intentions achievement. It embodies method knowledge to guide the developer in fulfilling an intention in a given situation. There are three kinds of guidelines: Intention Achievement Guideline (IAG), Intention Selection
Guideline (ISG) and Strategy Selection Guideline (SSG). An IAG gives advice on how to achieve the intention. It can be strategic represented as a map, tactic represented as a context or informal represented as natural language. Nevertheless, ISG and SSG are necessarily tactic represented as contexts (section 4).

This contextual approach is based on the description of situations that could arise during the process and which require decision-making from the developer. Each situation refers to one or a set of product parts. The product parts are results of transformation actions whereas situations are results of decision-making [12].

3.3. SWSD map

SWSD process mainly involves the definition of functional and security requirements, the design of Web services and the secure of Web services design. To conduct the modelling process of SWSD map (Fig. 2), we define a section. Then, we either define related guidelines or we identify other sections from the existing section [12]. The intentions used are necessary to carry out the SWSD process [1] [2] [3] [4] [14].

A non-deterministic ordering of intentions and strategies has been included in SWSD map. It does not specify what must be done but includes some specification of what can be done. SWSD map captures alternatives of intentions and strategies so as to provide developers with efficient guidance.

The construction of a situational SWSD method is performed under the control of guidelines. There are three kinds of guidelines: ISG, SSG and IAG. An ISG identifies the set of intentions that can be reached after the achievement of a given one and helps the developer to select the intention which is the most appropriate to the situation on hand.

As shown in SWSD map, each intention could be achieved using one or several strategies. The selection of the appropriate one is done under the control of SSG (section 5.1.). For every section (source intention, target intention, strategy) in the SWSD map, there exists one IAG represented by a SWSD chunk (section 5) which supports the developer in the achievement of target intention from the source intention according to the selected strategy.

Topos [8] strategy and Roy [11] strategy are integrated in the SWSD map to reach the intention: Design a Web Service. Security requirements should be defined to secure the Web Service Design. To achieve this intention, SWSD map provides Agile Strategy [14] and WSSecReq strategy [2]. The selection of the appropriate strategy relies on the situation on hand.

While guided by SWSD map, the developer selects and integrates SWSD chunks to construct the appropriate SWSD situational method. Following this approach, a SWSD method is viewed as a collection of SWSD chunks selected dynamically, depending upon the facing situation of the product. As it can be seen, SWSD map approach can be continuously improved by adding other new efficient strategies to design high secure Web Services.

4. SWSD method chunks

To specify the guidelines associated to the SWSD map, we propose to define existing SWSD method using a modular modeling approach. In this way, a chunk is based on the decomposition of a given SWSD method process model into reusable guidelines. The required product parts are associated to each of these guidelines to define them as complete method chunks. For this purpose, we present a method meta-model (Fig. 3) allowing the reengineering of existing SWSD methods and the definition of any SWSD method as an assembly of SWSD method chunks. This method meta-model is based on situational method engineering approach presented in [7], [12], [13], [14].

Each guideline which can be reused apart from its original method is defined as a chunk. A guideline has an interface and a body. The interface is defined by the couple <situation, intention> which describes the
A simple guideline provides informal explanation indicating how to handle the situation. A tactical guideline is represented by a tree structure of contexts (sub-guidelines) following the NATURE process modeling formalism [10]. There are two structures of tactical guidelines: the choice and the plan [10] [13].

A strategic guideline is complex. It can be formalized using the NATURE or the map process modeling formalisms [13]. It can provide a strategic view of the process indicating which intention can be fulfilled following which strategy. Such strategic guidelines use map formalism to relate their sub-guidelines. Each sub-guideline can be strategic, tactic or simple. The next section illustrates an example of a strategic guideline defined using map modeling approach.

5. System pattern method chunk

To achieve the intention: Secure a design, SWSD map (Fig. 2) provides 3 strategies: patterns strategy, WSSEcArch strategy, and agile strategy. For example, for the patterns strategy, an Intention Achievement Guideline called IAG5, was defined by the reengineering of the security design patterns approach [1]. The guideline interface is: < (Security Requirements: state (Security Requirements) = defined), Secure a design with Patterns strategy>. The situation refers to the defined security requirements.

According to our approach, any method chunk body comprises two components: a product model and process model. IAG5 is a complex guideline which provides several strategies to reach a given intention, so it is defined as a map (Fig. 4) and a set of sub-guidelines. The required product parts, such as pattern and design component are assigned to IAG5 map to produce Pattern method chunk. The aim of this chunk is to secure a given design component.

5.1. System Pattern Process Model

Security design patterns approach provides a set of security design patterns and a process for using these patterns to design a secure software system [1]. The proposed process is linear involving three main steps:

1. Choice of a design which satisfies the functional requirements of the system and identification of the design components including their interfaces, data repositories, communication links and protocols.

2. Application of the available system sequence.

3. Application of the protection system sequence.

As it can be seen in Fig. 4, the IAG5 map provides a number of paths for going from Start to Stop. This graphic representation shows that the guidance of IAG5 method chunk construction and execution consists of navigating the IAG5 map under the control of IAG5 guidelines. For instance, after the achievement of the intention: Choose an initial design component with Priority strategy, SSG1.1 helps in the selection of the appropriate strategy to Secure the selected design component (Fig. 5). To achieve this intention, IAG5 map provides two strategies: Availability patterns strategy and Protection patterns strategy. SSG1.1 is characterized by an interface and a body. The body of

![Component-based method meta-model](image-url)

![IAG5 map](image-url)
SSG1.1 guideline suggests two alternatives to Secure a design component. The available arguments help the developer in his choice. For instance, IAG5.1 is selected if the developer wants to begin by addressing availability requirements or if only such requirements have to be satisfied. This guideline is detailed in section V.

Patterns are divided into two types: availability patterns and protection patterns. Availability patterns can be applied to a critical design component whereas protection patterns can be applied to either critical resources or critical actors.

5.3. Availability patterns chunks

The reengineering of guidelines continues until the entire method chunks are generated. This section presents the IAG5.1 chunk method.

5.3.1. Availability patterns process model.

Security design patterns approach defines an Available System Sequence to secure a given component design [1] indicating to:

1. Identify critical design components.
2. Apply the Error Detection/Correction pattern to the defined component if data corruption is a likely cause of system failures.
3. Use the Comparator-Checked Fault-Tolerant System pattern to address the availability requirement. Otherwise, continue to the next step.
4. Use the Standby pattern to address the availability requirement if single component failures are acceptable but service outage is unacceptable. Otherwise, continue to the next step.
5. Use the Replicated System pattern to address the availability requirement if partial or localized service outages are acceptable, as long as service is available from alternate components.

The analysis of this sequence shows that different strategies can be used to Secure a component design and the choice of the best relies on the situation on hand. Thus, IAG5.1 is a strategic guideline and its body is defined as a map (Fig. 7). IAG5.1 interface is:

< (Initial design component: state (initial design component) = chosen), Secure a design component with availability patterns strategy>

Fig. 6. IAG5 product model

The product model represents the class of products obtained as outputs of the execution of SWSD method chunks process models. It is composed of a set of concepts which have properties and can be related through composition, association links and Is-a links.

The System Pattern product model is centred on the concept of a component design which can be a critical component, a critical resource or a critical actor (Fig. 6). A critical design component is mainly a critical repository or a critical processing element.
5.3.2. Availability patterns product model.

As shown in Fig. 8, a critical design component can be either a critical repository or a critical processing element. To secure it, availability patterns are applied. These patterns are divided into four types: Error Detection/Correction patterns, Comparator-Checked Fault-Tolerance patterns, Standby patterns and Replicated system patterns.

![Fig. 8. IAG5.1 product model](image)

6. Conclusion

In this paper, we addressed the problem of securing Web Service Design. Our approach provides, to Web service designers, guidance in constructing and executing situational SWSD methods. To respond to the dynamically changing situation, interactivity, backtracking, creativity and flexibility that characterize SWSD process, we formalize it using a situational process model including a map and associated guidelines. The construction and execution of situational SWSD methods is performed by navigating the SWSD map under the control of SWSD guidelines.

In this way, new SWSD methods can be constructed by selecting method chunks that are most suited to a given situation. This leads to an integration of the strengths of different SWSD methods.

7. References


