

Transforming Socio-Technical Security Requirements in SecBPMN Security Policies

Mattia Salnitri and Paolo Giorgini

University of Trento, Italy

{mattia.salnitri, paolo.giorgini}@unitn.it

Abstract. Socio-Technical Systems (STSs) are complex systems composed of both social (i.e., humans and organizations) and technical (i.e., hardware and software) elements. Security requirements for STSs define constraints for the socio-technical interactions and can be specified as a set of security policies that have to be satisfied by the components of the system during their interactions. In this paper, we present how security requirements modeled in STS-ml are transformed in security policies expressed in SecBPMN, an extension of BPMN with security annotations.

1 Introduction

Socio-Technical Systems (STSs) are complex systems where both social and technical components interact one another to achieve shared goals. Smart cities, health care, Air Traffic Management (ATM) systems are only few examples of STSs. In STSs, security requirements identify constraints interactions that have to be satisfied by the components of the system during their activities. STS-ml [2] is a diagrammatic language which allows analysts to model socio-technical security requirements. For example, in a ATM system, a socio-technical security requirement may specify that, if a Flying Object (FO), which is responsible of creating the flight plan to land in airport, delegates the control tower of an airport, the control tower shall not repudiate such a delegation.

STS can be modeled in terms of business processes, where activities are assigned and executed by single components and the control flows regulate their interactions. SecBPMN [5] (Secure BPMN) is a modeling language, based on Business Process Modelling and Notation (BPMN), which allows analysts to (i) model business processes with security annotations; (ii) define security policies, i.e., security constraints in terms of SecBPMN concepts. For example, the flight plan can be negotiated following a SecBPMN business process which specifies the interactions between the FO and the control tower. A SecBPMN security policy may specify that the information about the flight plan shall be exchanged between the FO and the control tower only after the execution of the activity which starts the negotiation.

STS-ml and SecBPMN models express different aspects of STSs, in particular SecBPMN models specify STS-ml models, hence both models are needed to fully characterize STSs. Specifying STS-ml security requirements in terms of SecBPMN security policies is a complex and time consuming activity that calls for automated support. This is particularly true in STSs where the high number of security requirements and their

complexity make very hard and error prone this activity. In this paper, we propose a set of transformation rules to partially automate the specification STS-ml security requirements in SecBPMN security policies.

The paper is structured as follows. Section 2 describes STS-ml and SecBPMN while Section 3 contains the transformation rules. Section 4 concludes the paper and proposes possible future work.

2 Baseline

This Section briefly describes STS-ml and SecBPMN, i.e. the two languages we used to model socio-technical security requirements and security policies.

2.1 STS-ml

Many security requirements modeling languages have been proposed so far. For example, SI* [3] and Secure Tropos [4] aim at modeling security needs. We choose STS-ml [2] (Socio-Technical Security- modeling language), a goal-oriented language for STSs, in which security requirements constrain the interactions between actors.

In STS-ml, requirements models are expressed by three views: (i) the social view describes the main actors, their rationale, goal delegations, and document provisions; (ii) the information view defines the relationship between information and documents; and (iii) the authorization view represents the authorizations about information that actors grant one to another. Figure 1 shows an example of STS-ml model of an airport scenario.

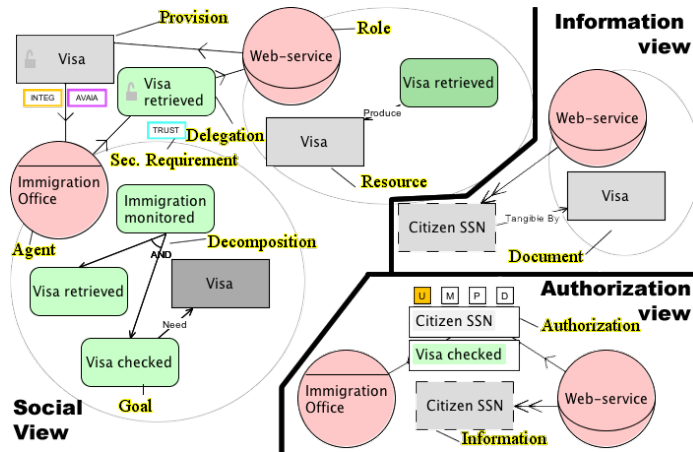


Fig. 1. STS-ml model of an ATM scenario

Social view. Two types of actors are modeled: agents—concrete entities that are known at design-time (e.g., *Immigration office*)—and roles—that can be played by different

agents at runtime (e.g., *Web-Service*). An actor's rationale is an AND/OR goal tree (e.g., the root goal of the *Immigration Office* is to be *Immigration monitored*). To achieve their goals, actors need, modify, and produce documents (e.g., *Immigration Office* needs document *Visa* to achieve goal *Visa checked*). Two social relationships link couples of actors: goal delegation and document provision. Both relations can be subject to security requirements (e.g., integrity, availability, ...).

Information view. Documents and information are linked one to another. The “Tangible by” relation indicates that an information is represented by a document. In Figure 1, *Citizen SSN* (Social Security Number) is made tangible by document *Visa*. Ownership relates an actor to the information that she owns.

Authorization view. It represents the permissions on information that actors grant one to another. An authorization box details the granted permissions (Use, Modify, Produce, Distribute) on documents representing specific information. Authorizations can be limited by defining a scope: one or more goals for whose fulfillment the permissions are granted. In Figure 1, the *Web-Service* authorizes the *Immigration Office* to use *Citizen SSN* in the scope of goal *Visa checked*.

2.2 SecBPMN

SecBPMN [5] is a framework aimed to model business processes with security aspects, to model security policies and to verify if such security policies are satisfied by the business processes. It is composed by: SecBPMN-ml (SecBPMN-modeling language), SecBPMN-Q (SecBPMN-Query) and a software component.

SecBPMN-ml extends BPMN with security concepts about information assurance and security defined in [1]. In the literature there are a number of modeling languages that extend BPMN with security concepts, but they can represent only a restricted set of security concepts, while SecBPMN-ml covers, as far as our knowledge goes, the most comprehensive set of security concepts. Each SecBPMN-ml security annotation is represented with an icon with a solid orange circle, and is detailed by a predicate which specifies further details on the security aspects represented by the annotation. Security annotations can be used as patterns or anti-patterns, depending on the exigencies of security designers.

Figure 2 shows part of a SecBPMN-ml model of a business process used in an airport information system where users can use different web-interfaces to select the best option for a flight, buy tickets and perform most of the bureaucratic processes required to take the flight.

SecBPMN-Q is a graphical query language which permits to define security policies in terms of SecBPMN-ml elements. Figure 3 shows an example of the textual policy “*The visa document must be sent through a secure channel, which assures the information will not be modified by third parties*”, modeled with SecBPMN-Q. The graphical security policy is composed by two activities labeled “@X” and “@Y”, “@” symbol is used to match any activities. The two activities are linked with a path relation (the arrow with two slashes in the middle) which matches all the paths between the source and target activity. The security policy is enriched with a message flow (represented as a dashed arrow) which transports data object called “Visa”. When executed, this security

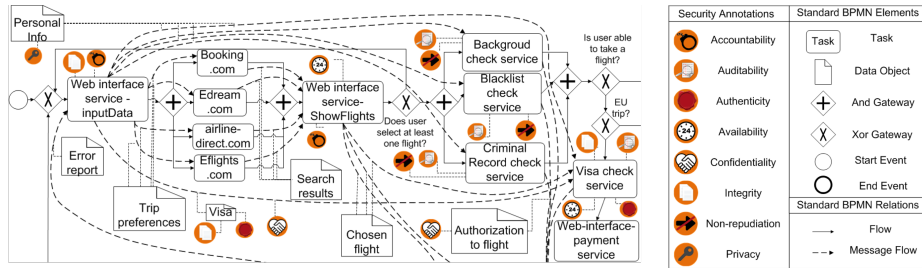


Fig. 2. SecBPMN-ml model of an ATM scenario

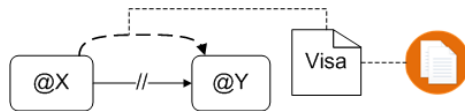


Fig. 3. Example of a security policy modeled using SecBPMN-Q

policy will match any message flow between any two activities which exchange the “Visa” data object. The “Visa” data object is linked to an integrity annotation, which means that has to be protected by unauthorized modifications.

3 Transformation rules

This section describes a set of transformation rules, that are used to specify all types of STS-ml security requirements in SecBPMN security policies. STS-ml and SecBPMN modeling languages have different scopes and, hence, they model different set of concepts. Therefore, for each transformation rule we define one or more mapping relations which link STS-ml concepts with SecBPMN concepts.

In the rest of the section, for each STS-ml security requirement, we describe its semantic (the first sentence) and the transformation rule we defined. Table 1 shows the transformation rules and the mapping needed for the transformations. The first four transformations create anti-patterns, i.e. the security policy shall not be satisfied in any business processes, the other transformations create patterns, i.e. the security policy shall be satisfied by all business process. Such transformation rules can be used to partially automate the transition from STS-ml to SecBPMN models, in order to help security designers in the specification of SecBPMN models and security policies from STS-ml models. For example the security policy in Figure 3 is derived from the transformation rule of integrity security requirement, with the mapping $g \Rightarrow Visa$.

Non-disclosure(Goal g, Document d) The document d shall not be disclosed in order to achieve g, to unauthorized actors. The transformation consists in defining a security policy that matches all the message flows, from ACT and transmit the document d.

Non-usage(Goal g, Document d) The document d shall not be used (i.e. read) to achieve goal g. We created a transformation rule which creates a security policy which the data object mapped to d, i.e. DO, is read by the activity mapped to g, i.e. ACT.

Non-modification(Goal g, Document d) The document d shall not be modified in order to achieve g . We transformed it in a security policy which contains an activity which reads and writes the same data object, i.e. it modifies the data object.

Non-production(Goal g, Document d) The document d shall not be created in order to achieve g . It's transformed in a policy which contains an activity which writes the data object DO.

Non-repudiation(Delegation(Goal g)) The acceptance of the delegation of goal g shall not be repudiated. It is transformed in a policy which contains an activity ACT, that is mapped to goal g , linked to a non-repudiation security annotation.

Redundancy(Goal g, Type t) Goal g shall be achieved with a redundancy strategy t . STS-ml defines two types of redundancy strategies: fall-back redundancy and true redundancy. We specified it in two security policies. For fall-back redundancy we define a policy which is satisfied by a path where there is at least an exclusive gateway between the main and the backup activities, which represents the decision to use the backup activity. For the true redundancy we define a policy which is satisfied when the two activities are executed in parallel paths. These policies require a mapping of g to two activities: a main activity the backup one.

Integrity(Document d) The document d shall not be modified by unauthorized users. We transformed it in a security policy which requires the integrity security annotation attached to the data objects that corresponds to the document d .

Availability(Goal g, Float a) The percentage of time g is achieved over of the total times that is requested, shall be higher than a . We transformed it in a policy which contains an activity linked to an availability security annotation. The level of availability a , is specified in the predicate which details the security annotation.

Trustworthiness(Goal g, Int t) Goal g shall be achieved by an actor with a level of trustworthiness greater than t . This is transformed in a policy which contains an activity linked to an authenticity security annotation. The trustworthiness level t is specified by the predicate which details the security annotation.

Non-delegation(Goal g, Actor a) Goal g shall not be delegated, by a , to other actors. This security requirement cannot be transformed in a security policy, because in SecBPMN there is no concept that can be mapped to actor.

Need to know(Document d, Goal g) All operations on document d shall be executed only for the purpose of achieving g . As the previous security requirement, this cannot be specified as a security policy, but it is implicitly verified when the the security policies of non-usage, non-modification and non-production are verified.

Bind of duty(Role r1,r2) and Separation of duty(Role r1,r2) All actors that play $r1$ shall (not) play also $r2$. These security requirements cannot be specified in security policies, because in SecBPMN there is no concept that can be mapped to actor.

4 Conclusions and future work

In this paper, we have proposed a set of rules to transform security requirements, expressed with STS-ml, into security policies, expressed with SecBPMN. These transformation rules are intended to automatize the specification activity and then support the analyst in the security requirements engineering process for STSs.

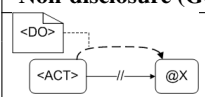
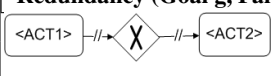
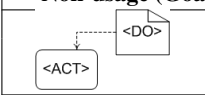
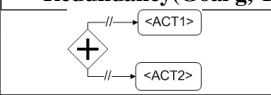
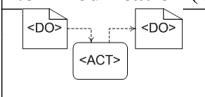
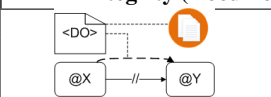
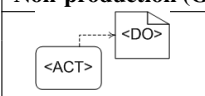



Non-disclosure (Goal g, Document d) 	$g \Rightarrow \langle \text{ACT} \rangle$ $d \Rightarrow \langle \text{DO} \rangle$	Redundancy (Goal g, Fall-back red.) 	$g \Rightarrow \langle \text{ACT1} \rangle$ $g \Rightarrow \langle \text{ACT2} \rangle$
Non-usage (Goal g, Document d) 	$g \Rightarrow \langle \text{ACT} \rangle$ $d \Rightarrow \langle \text{DO} \rangle$	Redundancy (Goal g, True red.) 	$g \Rightarrow \langle \text{ACT1} \rangle$ $g \Rightarrow \langle \text{ACT2} \rangle$
Non-modification (Goal g, Document d) 	$g \Rightarrow \langle \text{ACT} \rangle$ $d \Rightarrow \langle \text{DO} \rangle$	Integrity (Document d) 	$d \Rightarrow \langle \text{DO} \rangle$
Non-production (Goal g, Document d) 	$g \Rightarrow \langle \text{ACT} \rangle$ $d \Rightarrow \langle \text{DO} \rangle$	Availability (Goal g, Float a) 	$g \Rightarrow \langle \text{ACT} \rangle$
Non-repudiation (Delegation (Goal g)) 	$g \Rightarrow \langle \text{ACT} \rangle$	Trustworthiness (Goal g, Int t) 	$g \Rightarrow \langle \text{ACT} \rangle$

Table 1. Transformation rules

As part of our future work, we will develop a software tool to automatically transform security requirements into security policies and we will define a language to support security analysts in defining their own transformation rules.

Acknowledgement

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no 257930 (Aniketos).

References

1. Y. Cherdantseva and J. Hilton. A reference model of information assurance and security. In *Proc. of ARES '13*, pages 546–555.
2. F. Dalpiaz, E. Paja, and P. Giorgini. Security Requirements Engineering via Commitments. In *Proc. of STAST'11*, 2011.
3. P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling Security Requirements through Ownership, Permission and Delegation. In *Proc. of RE'05*, pages 167–176, 2005.
4. H. Mouratidis and P. Giorgini. Secure Tropos: A Security-Oriented Extension of the Tropos methodology. *IJSEKE*, pages 285–309, 2007.
5. M. Salnitri, F. Dalpiaz, and P. Giorgini. Modeling and verifying security policies in business processes. In *Proc. of BPMDS'14*.