Modeling and Verification of ATM Security Policies with SecBPMN

Mattia Salnitri
University of Trento
Trento, Italy
Email: mattia.salnitri@disi.unitn.it

Paolo Giorgini
University of Trento
Trento, Italy
Email: paolo.giorgini@disi.unitn.it

***Abstract*—High Performance Computing (HPC) techniques are essential in complex systems such as Socio-Technical Systems (STSs), where humans and organizations are elements of the same system along with technical infrastructures and hardware/software components. For example, several HPC approaches have been successfully applied to support and facilitate distribution or aggregation of computation power among independent and atomic components (e.g., smart meters to solve and/or simulate complex models). However, HPC techniques have to be studied and developed without underestimating the problem of security that, given the interaction-centric nature of STSs, has to be considered not only from the single component perspective but for the system as a whole. In our previous work, we have proposed SecBPMN, a framework to support the design of secure STSs. It is used to model the interaction design and security policies of a STS and it supports their verification through a querying engine. In this paper, we describe how SecBPMN has been successfully used for the study of security in an Air Traffic Management (ATM) system, and we show how it can result also an efficient support when of HPC techniques when applied in complex and heterogeneous environments.**

***Keywords*—Secure System Design; Secure Monitoring and Management**

I. Introduction

Socio-Technical Systems (STSs) are a well know type of systems composed by independent components (both humans and hardware/software) that interact one another to solve complex and distributed tasks. For example, in smart grid systems, end points smart meters, power stations and data management systems are independent nodes that exchange information and work together to distribute energy among users. Literature on HPC techniques offers a wide range of solutions for STSs that have been successfully applied in many different fields, such as energy, healthcare and smart environments where complex models are solved aggregating computational power of single independent units [6].

In STSs, security is a crucial problem that cannot be studied only from a technological perspective (i.e., analyzing and finding security threats and security solutions in terms of the adopted technologies), but rather it has to be analyzed from a wider perspective taking into account how independent components socially interact one another [5]. This socio-technical view allows for a more comprehensive analysis of security and a better understanding of the vulnerabilities of the system.

Security requirements impose a number of restrictions on HPC techniques that have to be properly addressed in the design of a STS. To identify such restrictions, a STS can be conceptualized as a complex organization where interactions among its operative units are represented as business processes and security policies are used to regulate or prevent undesired behaviors. Although, Business Process Modelling and Notation (BPMN) is the de-facto standard for representing the interaction among components of complex systems, it does not allow for representing security restrictions. Moreover, literature on BPMN does not offer any effective solution to easily verify whether a given business process is compliant or not with desired security policies.

In [10], we have introduced SecBPMN, a framework composed of a language for modeling business processes including security concerns, called SecBPMN-ml (SecBPMN- modeling language), a query language for specifying security policies, called SecBPMN-Q (SecBPMN-Query) and a software which permits to verify whether a SecBPMN-Q security policy is satisfied by a SecBPMN-ml business process. In this paper, we illustrate the application of SecBPMN to the SWIM[a] ATM case study. We use SecBPMN-ml and SecBPMN-Q to model, respectively, the business processes and security policies of the SWIM ATM STS, and the SecBPMN software to verify the identified SWIM ATM security policies.

The paper is structured as follows: Section II introduces SecBPMN framework, Section III describes the case study. Section IV details the application of SecBPMN to the case study while Section V describes the verification process we used to maintain security policies satisfied. Section VI contains the conclusions and the plans for future work.

II. SecBPMN

SecBPMN [10] is framework for modeling business processes with security aspects, to model security policies and to verify if such security policies are satisfies by the business processes. It is composed by: SecBPMN-ml, SecBPMN-Q and a software components.

[a]The System Wide Information Management (SWIM) [1]

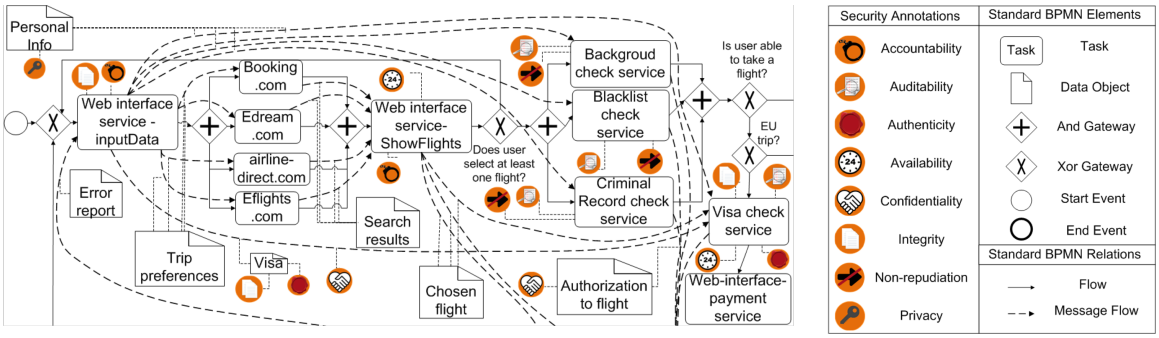

Figure 1. Example of a business process modeled using SecBPMN-ml

BPMN is the de-facto standard for modeling business processes. It is an effective means for expressing the interaction between components in complex information systems, but it does not offer the possibility to represent security concepts in such processes. Various extensions of BPMN for security were proposed, for example SecureBPMN [2] or the extension proposed by Rodriguez at al. [8], but such languages constrain designers to use a fix set of security policies. Other approaches, for example Liu et al. [7] and Rushby [9], consist in formal languages and frameworks to verify custom security policies. But this type of approaches are not usable with real case scenarios because of the high complexity of the languages.

SecBPMN-ml, overcomes this limitation extending BPMN with security concepts about information assurance and security defined in [3]. Figure 1 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-ml models are enriched with a set of security annotations (i.e. icons with a solid orange circle), that represent security aspects defined in [3]. Each security annotation is detailed by a predicate which specifies further details on the security aspects of the business process.

SecBPMN-Q is a graphical query language which permits to define security policies in terms of SecBPMN-ml elements. Figure 2 shows an example of the textual policy "*The visa document must be authenticated and it must be sent through a secure channel, which assures the information will not be sniffed or modified by third parties*", modeled with SecBPMN-Q. The graphical security policy is composed by two activities labeled with "@X" and "@Y", "@" symbol is used to match any activities. The activities are linked with a path relation (the arrow with two slashes) which matches all the business processes where the first activity is executed before the second activity. The security policy is enriched with a message flow (represented as a dashed arrow) which exchange a data object called "Visa". This security policy will match any message flow between two activities which exchange the "Visa" data object. The confidentiality annotation requires the communication channel to assure the data object will be received

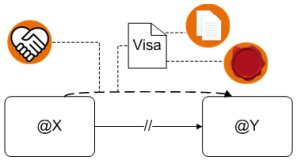

Figure 2. Example of a SecBPMN-Q security policy

only by authorized users. Moreover, the "Visa" data object has to be protected by unauthorized modifications and its originality has to be provable, specified by authenticity and integrity annotations. Details on the security annotations can be expressed using predicates, but are not reported in this paper for the sake of brevity.

The SecBPMN-Q security policy in Figure 2 is satisfied by any path, i.e., any sequence of activities, from "Web interface service - inputData" to "Visa check service" of the business process shown in Figure 1. This is because: (i) "Web interface service - inputData", is linked with a message flow to "Visa check service"; (ii) the message flow is used to exchange the data object "Visa" and it assures confidentiality of the transferred data object; (iii) integrity and authenticity of the "Visa" data object are preserved. Assuming the predicates that details the security annotations of the security policy are less restrictive of the predicates of the business process, the business process satisfies the security policy. For further details, please refer to [10].

In real case scenarios, where business processes can be composed by tens of elements, is not possible to verify security policies manually. The software[b] provided with SecBPMN framework verifies automatically if a SecBPMN-Q security policy is satisfied by a SecBPMN-ml business process.

III. SWIM ATM CASE STUDY

SWIM ATM STS was analyzed, as a case study, as part of the Aniketos[c] European project. It consists of a large number of autonomous and heterogeneous components, which interact

[b] http://www.secbpmn.disi.unitn.it
[c] http://www.aniketos.eu

with each other to enable air traffic management operations: pilots, airports personnel, meteo services, radars, etc.

In such a complex system, ensuring security is critical, for security leaks may result in severe consequences on safety and confidentiality. For instance, a successful attack to the control tower, the core component of every airport, can paralyze an airport for days, with severe consequences for the passengers and consequently, for the company which manages the airport.

The dimensions of SWIM ATM STS are considerably wide but with similar, redundant, sub-parts. Hence we opted to focus on four representative aspects of such system: the landing process, the negotiation of the Reference Business Trajectory (RBT), i.e., the flight plan, and the external-services management and use. We believe these four aspects are distinctive examples of the salient characteristics of SWIM ATM STS: the management of functionalities typical of any ATM, and the dynamic management of external services.

IV. Modeling SWIM ATM STS with SecBPMN

This section describes the SWIM ATM business processes and an example of security policy we modeled using SecBPMN. We modeled 4 business processes, one for each aspects of the ATM SWIM STS we considered.

The first business process is executed to dynamically add external services to SWIM ATM STS. External services negotiate with ATM controllers the Quality of Service that will be offered to SWIM ATM users.This business process is composed by 28 elements that are performed by 4 participants. It contains 5 message flows and 7 data objects. In this business process 14 security annotations are used to reflect, on the SecBPMN model, the security aspects of each activity.

The second business process is executed when functionalities of external services are used. The SWIM ATM STS grants to internal users the reliability of external services. This is achieved with a trust-basted mechanism. This business process contains 55 elements that are executed by 5 participants. It contains 15 message flows and 16 data objects and 18 security annotations.

The third business process is about the dynamic negotiation of RBT. When a Flying Object (FO) is entering an area controlled by a control tower (TWR), it negotiates with the TWR the part of the RBT to cross the area. This business process is composed by 48 elements that are executed by 3 different participants. It contains 13 message flows and 17 data objects and 31 security annotations.

The forth business process is the landing process of FOs. In this scenario a FO negotiates a RBT to the landing point and the queue position. This business process is compose by 59 elements executed by 4 participants. It contains 14 message flows and 14 data objects and 31 security annotations.

Aniketos experts analyzed the security requirements documents and they identified a set of security policies, i.e. security constraints on the STS, using STS-ml (Socio-Technical Security - modeling language) [5], a goal-based modeling language. For example, Aniketos experts identified a security policy that specifies that sensitive data of customers of an airport can not be disclosed, hence they specified the content of the security policy, i.e. the constraint that costumers' information can not be disclosed, the requester, i.e. the customers themselves, and the responsible, e.g. the crew of the flights.

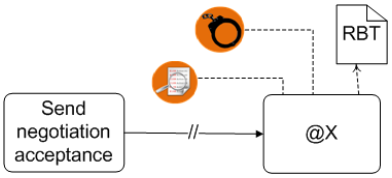

Figure 3. Example of a SWIM-ATM SecBPMN-Q security policy

The experts identified 27 active entities (among all responsible and requesters of security policies) and 60 security policies. We classified the security policies in 7 types, described in Table I with the number of their instances in the SWIM ATM STS.

We transformed all textual security policies in SecBPMN-Q security policies. This result of the transformation is not unique because it is based on the interpretation of the textual security policy, hence other security designers may obtain different security policy. This is one of the salient characteristic of SecBPMN: it does not force the designers to a unique interpretation of textual security policies. An example is the non-production security policy in Table I: it is transformed in the SecBPMN-Q model in Figure 3. The SecBPMN-Q security policy will be satisfied by all models where the activity "Send negotiation acceptance" is executed before any another activity ("@X") which creates the data object "RBT" from scratch (the creation of data objects is modeled with a dashed arrow from an activity to the created data object). The second activity is annotated with accountability and auditability security annotations, meaning that the actions performed to execute the activity are monitored and the involved users will be held for their misbehaviors. Such security annotations are required in order to have one or more security mechanisms to control at runtime the behavior of the service provider, when the activities are executed. For the sake of brevity we omit the predicates which specialize the security annotations.

V. Verifying and Maintaining Satisfaction of Security Policies

If security policies are not satisfied by all the business process of an STS, consequences will be severe [4]. Hence all business processes of a STS shall satisfy all security policies, otherwise they shall be modified in order to be compliant with such policies. For example, if the security policy of non-disclosure, showed in Table I, is not fulfilled by the business processes of the SWIM ATM STS, sensitive data of pilots will be disclosed, with monetary consequences, law consequences

TABLE I
DESCRIPTION OF SECURITY POLICIES TYPES

Type	#	Description
Non-disclosure	8	A data object will not be disclosed to unauthorized participants.
Non-usage	3	A data object will not be used by unauthorized participants.
Non-modification	11	A data object will not be modified by unauthorized participants.
Non-production	11	A data object will not be created by unauthorized participants.
Need to know	13	All operations on a data object are executed only for providing a specified functionality.
Redundancy	1	The functionality requested will be offered with a backup strategy.
Non-repudiation	13	The participant will not be able to deny the fact that he/she performs an action.

and a lost in the credibility of the entire system. Therefore, ion order to avoid such consequences, all the business processes which do not satisfy the security policy must be modified.

Frequently STSs change to adapt to external changes, hence the verification of security policies is required not only when the system is design, but also after the deployment. For example, in SWIM ATM STS the security mechanisms used by the TWR to enforce its communications may need to be substituted because a security bug is found. This implies that all business processes in which the TWR communicates with other entities of the STS are updated and, therefore, all security policies must be verified.

Business processes are not the only part of the STS that can change, also security policies can be modified. For example, if a law about privacy changes, e.g., it requires stricter controls on sensitive data, the security policies must be updated to be compliant with the new law. This change trigger the verification of the updated security policies and, in case it is not satisfied, the business processes are modified.

We used a specific process to maintain the satisfaction of security policies. The process starts every time a business process or a security policy are changed. If security policies are satisfied a new change is waited. Otherwise, the business process or the security policies are changed and, again, all security policies are verified.

The documentation of SWIM ATM STS we analyzed, does not have any historical data that can be used to simulate changes in the business process, therefore we defined a set of changes that are likely to happen in such systems. For example, security issues both in external and internal components, changes in the technologies used by a component and changes in the functionalities offered by internal components. All these changes bring the designers to adapt one or more business processes, i.e., to modify the flow of activities, and/or the security annotation linked to the activities.

VI. CONCLUSIONS

This paper describes an application of SecBPMN framework to the SWIM ATM case study. SecBPMN framework allows designers of a STS to verify whether a set of given security policies are satisfied by a set of security annotated business processes. We demonstrated that SecBPMN is an effective framework to support the design and the maintenance of secure complex systems providing a clear understanding of security policies and the restrictions they impose on the interactions among the components of the system.

As part of future work, we will explore more in detail how SecBPMN can be effectively used with HPC techniques to develop monitoring solutions for the runtime verification of security policies in STSs.

ACKNOWLEDGMENT

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] Federal Aviation Administration. SWIM ATM case study, last visited March 2014. http://www.faa.gov/about/office_ org/headquarters_ offices/ato/service_ units/techops/atc_ comms_ services/swim/.

[2] A. D. Brucker, I. Hang, G. Lückemeyer, and R. Ruparel. SecureBPMN: Modeling and Enforcing Access Control Requirements in Business Processes. In *Proc. of SACMAT'12*, pages 123–126.

[3] Y. Cherdantseva and J. Hilton. A reference model of information assurance and security. In *Proc. of ARES '13*, pages 546–555.

[4] R. Crook, D. Ince, L. Lin, and B. Nuseibeh. Security requirements engineering: When anti-requirements hit the fan. In *Proc. of RE'02*, pages 203–205. IEEE, 2002.

[5] F. Dalpiaz, E. Paja, and P. Giorgini. Security Requirements Engineering via Commitments. In *Proc. of STAST'11*, 2011.

[6] R.C. Green, L. Wang, and M. Alam. Applications and trends of high performance computing for electric power systems: Focusing on smart grid. *IEEE Trans. on Smart Grid*, 4(2):922–931, 2013.

[7] Y. Liu, S. Müller, and K. Xu. A static compliance-checking framework for business process models. *IBM Syst. J.*, 46(2):335–361, April 2007.

[8] A. Rodríguez, E. Fernández-Medina, and M. Piattini. A BPMN extension for the modeling of security requirements in business processes. *IEICE Trans. on Information and Systems*, 90(4):745–752, 2007.

[9] J. Rushby. Using model checking to help discover mode confusions and other automation surprises. *Reliability Engineering and System Safety*, 75:167–177, 2002.

[10] M. Salnitri, F. Dalpiaz, and P. Giorgini. Modeling and verifying security policies in business processes. *In Proc. of BPMDS'14*.