From security-by-design to the identification of security-critical deviations in process executions

Mattia Salnitri[1], Mahdi Alizadeh[2], Daniele Giovanella[3], Nicola Zannone[2], and Paolo Giorgini[3]

[1] Polytechnic University of Milan, Milan, Italy
mattia.salnitri@polimi.it
[2] Eindhoven University of Technology, Eindhoven, Netherlands
{m.alizadeh,n.zannone}@tue.nl
[3] University of Trento, Trento, Italy
daniele.giovanella@alumni.unitn.it, paolo.giorgini@unitn.it

Abstract. Security-by-design is an emerging paradigm that aims to deal with security concerns from the early phases of the system development. Although this paradigm can provide theoretical guarantees that the designed system complies with the defined processes and security policies, in many application domains users are allowed to deviate from them to face unpredictable situations and emergencies. Some deviations can be harmless and, in some cases, necessary to ensure business continuity, whereas other deviations might threat central aspects of the system, such as its security. In this paper, we propose a tool supported method for the identification of security-critical deviations in process executions using compliance checking analysis. We implemented the approach as part of the STS-Tool and evaluated it using a real loan management process of a Dutch financial institute.

1 Introduction

Security-by-design is a key emerging principle driving research innovation and industry development and maintenance of today's systems [2]. It includes methods, languages, techniques and tools to deal with security concerns since the initial phases of the system development and to perform verification and certification activities. In this context, business process modeling languages, such as BPMN [22], are currently used to represent the behavior of large and complex systems in terms of humans' and technical components' activities and their interactions. Extensions of such languages have demonstrated to be also effective in capturing security constraints and security policies [8, 26]. Automated reasoning techniques are then used to verify whether business process models satisfy specific security and privacy policies [4, 12, 30], such as those imposed by national or international regulations or simply needed to guarantee a certain level of security.

Security-by-design can guarantee compliance with security policies as far as business processes are executed according to their respective (predefined) models. However, there are many application domains where deviations of the executed processes from such models is the norm. For example, in a hospital it may occur frequently that internal processes are not executed as they were designed to deal with emergency. This is perfectly natural, since when humans face unpredictable situations they require a certain level of flexibility in

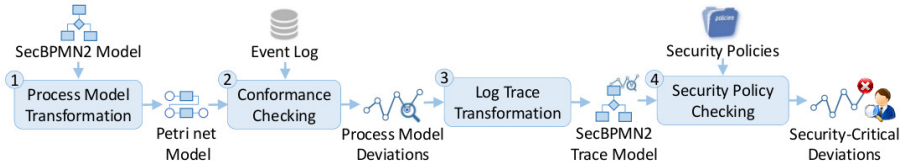


Fig. 1: An overview of the approach

the execution of their activities [28, 32]. It may also happen that deviations become habits for specific processes and they are never executed as they were originally designed and verified. From a security perspective, this may become a problem and it introduces the need to complement methods used to realize secure-by-design processes with methods for checking the conformance of process executions with the prescribed behavior.

Although the literature offers a number of approaches aiming to ensure that business processes comply with a set of security policies given at design time (forward compliance checking) [4, 8, 16, 37] and techniques to evaluate the conformance of process executions with the defined process models (backward compliance checking) [1, 3, 6, 23–25, 34], what is still missing is a comprehensive framework that can close the gap between verified and certified models and their actual executions, providing an effective support to deal with security within the entire lifecycle of the system.

In this paper, we propose a tool supported method for the identification of security critical deviations in process executions, where forward and backward compliance checking are fully integrated. We adopt a state-of-art BPMN-based modeling language (i.e., SecBPMN2 [26]) to represent and verify security concerns in business processes and off-the-shelf techniques for compliance checking analysis of process executions. We show how the combination of these techniques can benefit the identification and analysis of security-critical deviations. We present an application of our approach to a real loan management process of a Dutch financial institute along with an evaluation of its capabilities. The approach is fully supported by an extended version of the STS-Tool [27].

The paper is organized as follows. Section 2 presents an overview of our approach for the identification of security-critical deviations in process executions. Section 3 demonstrates the approach in a real loan management process. Section 4 presents the tool supporting our approach and its evaluation. Finally, Section 5 discusses related work and Section 6 concludes the paper.

2 Identifying security-critical deviations in process executions

This section describes our approach for the identification and analysis of security-critical deviations in process executions. Starting from secure-by-design processes, the approach aims to: (i) automatically analyze process executions, recorded in log traces, and identify deviations from the process specification; (ii) determine which deviations are security-critical; and (iii) visualize the identified deviations along with their security impact on the business processes. An overview of the approach is shown in Figure 1.

The approach takes as input a set of business processes, a set of security constraints on the business processes, hereafter called *security policies*, and a set of log traces. For the

definition of business processes and security policies, we rely on SecBPMN2 [26], which offers a means to define secure-by-design business processes. In particular, it provides a modeling language for the specification of business processes, called SecBPMN2-ml, that extends BPMN 2.0 [22] with security annotations, and a modeling language for the specification of security policies in the form of procedural patterns, called SecBPMN2-Q, along with capabilities to verify whether a given business process satisfies the defined policies.

An off-the-shelf backward conformance checking technique is used to identify log traces that deviate from the process specification. Deviating traces are then transformed into SecBPMN2 models to determine which deviations are security-critical. These deviations are graphically projected onto SecBPMN2 business processes for visual inspection. It is worth noting that most state-of-the-art (backward) conformance checking techniques use Petri nets for the representation of business processes. Thus, we have introduced an additional step for the automated transformation of BPMN2 process models into Petri nets. Below, we provide an overview of the main steps of our approach and underlying techniques. Then, in the next section, we illustrate these steps on a real loan management process.

Process Model Transformation The first step of the approach aims to transform a SecBPMN2 business process into the corresponding Petri net. This transformation is required for conformance checking (second step). Petri nets [13] provide a formal representation of business processes for which several conformance checking techniques and tools have been proposed and are currently available (see, e.g., [1, 24, 33]). The transformation consists in generating a Petri net that reflects the control flow specified by the given SecBPMN2 business process. We base this process model transformation on the work of Dijkman et al. [14]. In particular, Dijkman's work specifies transformation rules for most BPMN 2.0 elements such as tasks and gateways. However, such rules do not support the transformation of inclusive gateways due to limitations on the synchronization of multiple control flows. To this end, we extend the transformation rules proposed in [14] along the lines suggested in [35], where inclusive gateways are transformed into (portions of) Petri nets that simulate the inclusive gateway behavior. Although this solution does not fully address the problem of synchronization, this problem is only relevant when process models are simulated or executed. On the other hand, in conformance checking log traces are replayed on the process models and, thus, issues related to the synchronization of multiple control flows do not affect our approach.

Conformance checking The Petri nets generated in the previous step are used to identify process executions, recorded in log traces, that do not comply with the process specification. In this work, we adopt a backward conformance checking technique based on the notion of alignment [33]. In a nutshell, an alignment relates the events in a log trace to the activities in a run of the process, thus pinpointing the deviations causing nonconformity. If a log trace perfectly fits a Petri net, each "move" in the trace, i.e. each event recorded in the log trace, can be mimicked by a "move" in the model, i.e. an instance of a transition fired in the net. In cases where deviations occur, some moves in the trace cannot be mimicked by the net or vice versa. For instance, an activity could be executed when not allowed by the process model, resulting in a so-called *move on log*. Other times, an activity should have been executed according to the model but is not observed in the log trace. This results in a so-called *move on model*.

Log Trace Transformation The third step consists in generating a SecBPMN2 business process for each log trace that does not comply with the process model. This step is required for the identification of security-critical deviations (the next step of the approach). The log traces recorded by IT systems often contain only information on the name of the activities that were executed along with the order of their execution. Although this information covers extremely important aspects of security (e.g., deviations from the prescribed control-flow might affect the correct enforcement of security mechanisms), log traces usually neither contain information on the security mechanisms implemented nor on the data objects used in the execution of activities. Such information, however, is necessary to verify security properties, such as the correct access to data or enforcement of security needs. To overcome this limitation, we exploit the information specified in SecBPMN2 processes. In particular, we extend log traces by including the elements that are not present in the log traces, such as gateways, data objects and security annotations, as specified in the SecBPMN2 process models. Intuitively, the activities of a log trace are mapped to the tasks in the SecBPMN2 business process by matching their labels. After that, a SecBPMN2 business process is created using the original process as a template. Once the SecBPMN2 business processes representing the deviating log traces are generated, they are annotated to highlight where the deviations took place. The identified deviations should be easy to inspect so that proper measures can be taken. The challenge lies in providing a visualization of deviations that scales well with the number of traces, and uses a notation familiar to the person who defined the original business process. We address this issue by aggregating all deviating traces into a single SecBPMN2 model and thus providing an overview of where the deviations happened. We chose SecBPMN2 for the visualization of deviations as this minimizes the learning phase by reducing the amount of information users need to learn.

It is worth noting that our transformation of log traces in SecBPMN2 models assumes that tasks are executed as specified in SecBPMN2 models, i.e. with a correct implementation of security annotations, data access as specified in the model, etc. Such an assumption can be relaxed by considering richer logs or, possibly, different types of logs [1], which however are often unavailable or difficult to obtain. We leave an investigation of a more comprehensive extraction of security elements from logs for future work.

Security policy checking Among the log traces that deviates from the process specification, we are interested in the ones that are security-critical. The identification of security-critical deviations is performed using SecBPMN2. In particular, SecBPMN2 supports the checking of business processes expressed in SecBPMN2-ml against security policies expressed in SecBPMN2-Q. In a nutshell, SecBPMN2 verifies if there exists a path in the process model that satisfies the given security policies (recall that SecBPMN2-Q policies specify constraints on the order of execution of activities). For each path, it is verified whether the security annotations in the process model are of the same type of those in the security policies and whether they are linked to the same SecBPMN2 elements. In this case, the security annotations in the security policies are verified against the security annotations in the SecBPMN2 model. If at least one security policy is not verified, then the corresponding trace is considered security-critical.

3 Approach at work

This section describes an application of our method to the loan management process of a Dutch financial institute, taken from the 2012 BPI challenge [7]. We first introduce the case study along with the SecBPMN2 concepts that are necessary to understand the method and then we present a step-by-step application of the method. We assume the reader to be familiar with BPMN 2.0 [22] and Petri net [13] notations.

3.1 Loan management process

Our case study is about the analysis of an event log recording the loan management process of a Dutch financial institute, which was made available for the 2012 BPI challenge [7]. The event log contains the events recorded for three intertwined subprocesses: subprocess A specifies the handling of loan applications, subprocess O describes the handling of loan offers, and subprocess W specifies how work items are processed. For activities executed within subprocesses A and O, only events at stage complete are recorded, whereas for activities executed within subprocess W, events are recorded at stages schedule, start and complete.

Fig. 2 shows the loan management process in SecBPMN2 notation. This process model is obtained based on the analysis reported in [36]. The process starts when a client submits a credit request. Then, if needed, fraud check (W_Beoordelen fraude) or first assessment (W_Afhandelen leads) are performed. Applications are then finalized by filling in additional information (W_Completeren aanvraag) and an offer is sent to the client. An offer can be canceled or created multiple times during a process execution. In different states of the process, the client might be contacted (W_Nabellen offertes and W_Nabellen incomplete dossiers) for obtaining missing information. After returning an offer, the applications might undergo further assessment (W_Valideren aanvraag). At the end, an application can be denied, approved or canceled. Note that after approving an application, the contract can still be modified (W_Wijzigen contractgegevens).

Security requirements defining the correct execution of the loan management process are captured using SecBPMN2 security annotations. These annotations are represented with an orange circle with an icon denoting the type of security annotation. The language supports eleven types of annotations. Here, we only present the four that are used in the case study and refer interested readers to [26] for a complete description of SecBPMN2 security annotations.

Many organizations and, in particular, banks and financial institutes, require that sensitive activities are performed by different users to prevent conflicts of interest [17, 29]. In our case study, it is required that the handling of loan applications and offers is handled by at least two different employees. In SecBPMN2, this can be specified through a Separation of duties security annotation ( ) connecting two pools (or two lanes). This annotation indicates that the activities in the two pools (lanes) cannot be executed by the same person. In the process of Fig. 2, the security annotation specifies that Customer Service and Specialist cannot be the same person in an execution of the process.

Moreover, it should not be possible to challenge the execution of some activities, for example the approval (A_APPROVED) or denial (O_DECLINED) of a loan in Fig. 2, and the validity of the associated contracts. To this end, a legal proof of the execution of those activities is usually requested. Such a proof demonstrates, from a legal point of view, that

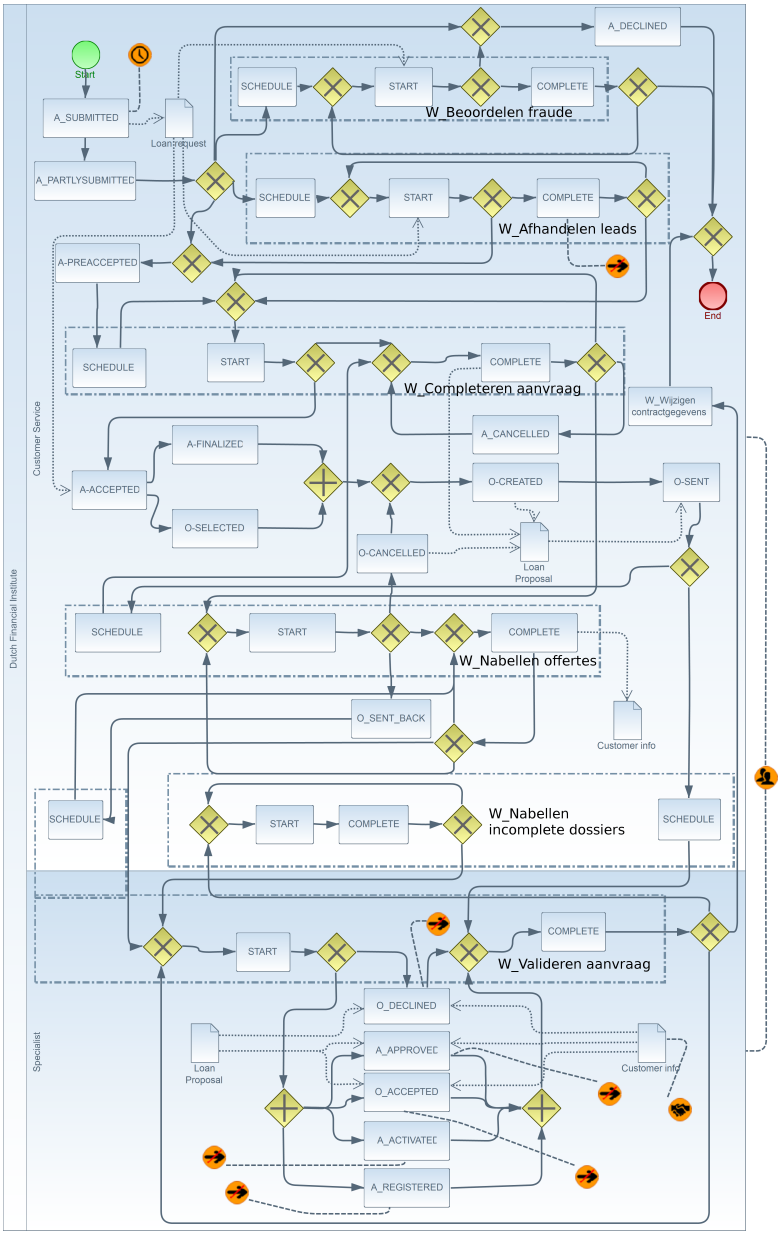
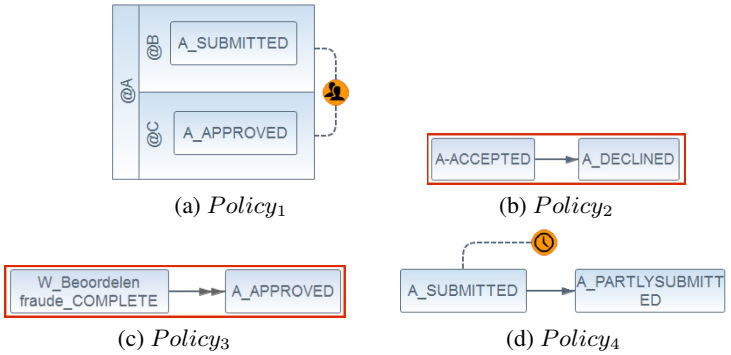


Fig. 2: Loan management process in SecBPMN2-ml notation

the activity has been executed and no one can deny its execution. In SecBPMN2, this is specified with a Non repudiation security annotation (➔), which denotes that a legal proof of the execution of an activity has to be generated.



(a) $Policy_1$ (b) $Policy_2$

(c) $Policy_3$ (d) $Policy_4$

Fig. 3: Policies for the loan management process of Fig. 2

Some activities can be central to the core businesses of an organization. Such types of activities are captured using Availability security annotations (), which specify that an activity should always be executed. For example, in Fig. 2, activity A_SUBMITTED is associated with an availability annotation indicating that customers should always be able to request a loan. SecBPMN2, optionally, allows the specification of an availability threshold indicating, for example, an availability of 99% of the time. Business processes may use personal data, stored in data objects, which requires special protection due to its sensitive content. SecBPMN2 uses the Confidentiality security annotation () to specify that the access to a data object must be prevented to unauthorized users. For example, in Fig. 2, data object customer info contains customer personal data and the security annotation specifies that the access to that data object should be limited only to authorized users.

SecBPMN2 also allows the specification of security policies to determine whether the defined process is secure-by-design. Fig. 3 presents four sample security policies for the case study defined using SecBPMN2-Q. $Policy_1$ indicates that the handling of the submission and approval of an application must be performed by two different users. This is specified with a Separation of Duties annotation linking two lanes. The @ wild card specifies there is no constraint on the user performing the activities, i.e. the policy applies to every user. $Policy_2$ and $Policy_3$ are anti-patterns (denoted by a red box), indicating that the behavior described in these policies should not be observed in any process executions. $Policy_2$ imposes that, after an application is accepted, it cannot be immediately declined. This is specified using a sequence flow relation indicating that A_DECLINED cannot be executed immediately after A_ACCEPTED. $Policy_3$ specifies that if an application is inspected for fraud, eventually it must not be approved. This is specified using a walk relation indicating that two activities must not be linked with an arbitrary long sequence of executable elements. Finally, $Policy_4$ indicates that the handling of application submissions has to be provided with high availability (specified using the availability annotation) and its execution should be followed by the execution of A_PARTIALLYSUBMITTED. This is specified using a sequence flow indicating that the target activity must be executed immediately after the source activity.

A verification of the process in Fig. 2 against the policies in Fig. 3 shows that the process is secure-by-design, i.e. all policies are satisfied by the process model. However,

	Activity	Resource
1	A_SUBMITTED-COMPLETE	112
2	A_PARTLYSUBMITTED-COMPLETE	112
3	A_PREACCEPTED-COMPLETE	112
4	W_Completeren aanvraag-SCHEDULE	112
5	W_Completeren aanvraag-START	10982
6	A_ACCEPTED-COMPLETE	10982
7	O_SELECTED-COMPLETE	10982
8	A_FINALIZED-COMPLETE	10982
9	O_CREATED-COMPLETE	10982
10	O_SENT-COMPLETE	10982
11	W_Nabellen offertes-SCHEDULE	10982
12	W_Completeren aanvraag-COMPLETE	10982
13	W_Nabellen offertes-START	10982
14	W_Nabellen offertes-COMPLETE	10982
15	W_Nabellen offertes-START	11001
16	O_SELECTED-COMPLETE	11001
17	O_CANCELLED-COMPLETE	11001
18	O_CREATED-COMPLETE	11001
19	O_SENT-COMPLETE	11001
20	W_Nabellen offertes-SCHEDULE	11001
21	W_Nabellen offertes-COMPLETE	11001
22	W_Nabellen offertes-START	11049
23	O_SENT_BACK-COMPLETE	11049
24	W_Valideren aanvraag-SCHEDULE	11049
25	W_Nabellen offertes-COMPLETE	11049
26	W_Valideren aanvraag-START	10138
27	W_Valideren aanvraag-COMPLETE	10138
28	W_Valideren aanvraag-START	10138
29	W_Valideren aanvraag-COMPLETE	10138
30	W_Valideren aanvraag-COMPLETE	10138
31	A_APPROVED-COMPLETE	112
32	A_ACTIVATED-COMPLETE	112
33	A_REGISTERED-COMPLETE	112

(a)

	Activity	Resource
1	A_SUBMITTED-COMPLETE	112
2	A_PARTLYSUBMITTED-COMPLETE	112
3	W_Afhandelen leads-SCHEDULE	112
4	W_Afhandelen leads-START	11003
5	A_PREACCEPTED-COMPLETE	11003
6	W_Completeren aanvraag-SCHEDULE	11003
7	W_Afhandelen leads-COMPLETE	11003
8	W_Completeren aanvraag-START	10929
9	W_Completeren aanvraag-COMPLETE	10929
10	W_Completeren aanvraag-START	10913
11	W_Completeren aanvraag-COMPLETE	10913
12	W_Completeren aanvraag-START	10929
13	A_ACCEPTED-COMPLETE	10929
14	A_DECLINED-COMPLETE	10929
15	W_Completeren aanvraag-COMPLETE	10929

(b)

	Activity	Resource
1	A_SUBMITTED-COMPLETE	112
2	A_PARTLYSUBMITTED-COMPLETE	112
3	A_PREACCEPTED-COMPLETE	112
4	W_Completeren aanvraag-SCHEDU...	112
5	W_Completeren aanvraag-START	
6	A_ACCEPTED-COMPLETE	11200
7	A_FINALIZED-COMPLETE	11200
8	O_SELECTED-COMPLETE	11200
9	O_CREATED-COMPLETE	11200
10	O_SENT-COMPLETE	11200
11	W_Nabellen offertes-SCHEDULE	
12	W_Completeren aanvraag-COMPLETE	
13	W_Nabellen offertes-START	11049
14	O_SENT_BACK-COMPLETE	11049
15	W_Valideren aanvraag-SCHEDULE	11049
16	W_Nabellen offertes-COMPLETE	11049
17	W_Valideren aanvraag-START	10629
18	W_Beoordelen fraude-SCHEDULE	10629
19	W_Valideren aanvraag-COMPLETE	10629
20	W_Beoordelen fraude-START	10188
21	W_Valideren aanvraag-SCHEDULE	10188
22	W_Beoordelen fraude-COMPLETE	10188
23	W_Valideren aanvraag-START	10629
24	A_REGISTERED-COMPLETE	10629
25	A_APPROVED-COMPLETE	10629
26	O_ACCEPTED-COMPLETE	10629
27	A_ACTIVATED-COMPLETE	10629
28	W_Valideren aanvraag-COMPLETE	10629

(c)

Fig. 4: Examples of log traces.

reality can diverge from the prescribed behavior. Fig. 4 presents three log traces recorded by the financial institute that deviate from the loan management process in Fig. 2. In the next section, we illustrate our approach for the identification of security-critical deviations in process executions. Then, in Section 4, we present an analysis of the log traces in Fig. 4.

3.2 Walkthrough application of the approach to the case study

This section describes the application of the method presented in Section 2 to the loan management process introduced in the previous section.

Process model transformation After the loan management process has been modeled in SecBPMN2-ml along with security annotations, it is automatically transformed into a Petri net. Fig. 5 shows a portion of the Petri net generated from the process model expressed in SecBPMN2 notation shown in Fig. 2. Activities are transformed into transitions with an input and output places. For example, activity A_SUBMITTED in Fig. 2 is transformed into a transition with the same name along with two places, one before and one after the transition. Another example of the application of transformation rules is the exclusive gateway after activity A_PARTIALLYSUBMITTED in Fig. 2, which is transformed, in Fig. 5, in a place connected with as many transitions as the outgoing control flows. Our transformation rules provide optimizations over the transformation rules in [14] to reduce the size of the generated Petri net. For example, the transitions after the place encoding the exclusive gateway denote the activities to be executed after the gateway, instead of duplicating places and transitions as indicated in [14].

Conformance checking The generated Petri net is used to assess the conformity of log traces with the model. As discussed in Section 2, we employ an off-the-shelf alignment-based technique [33], which provides a robust approach to conformance checking able

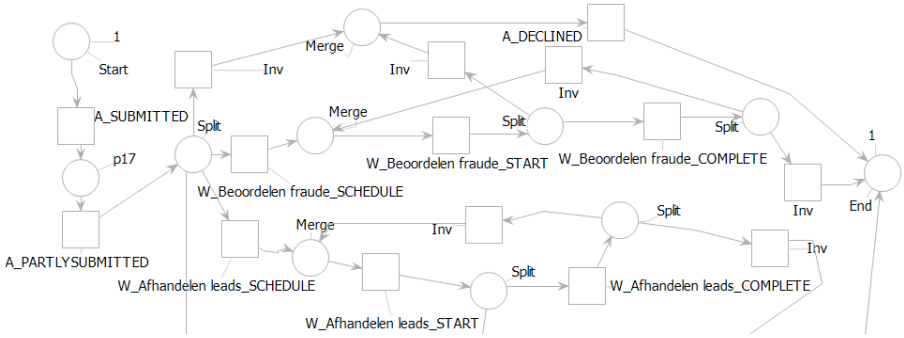


Fig. 5: Portion of the Petri net generated from the SecBPMN2 business process in Fig. 2

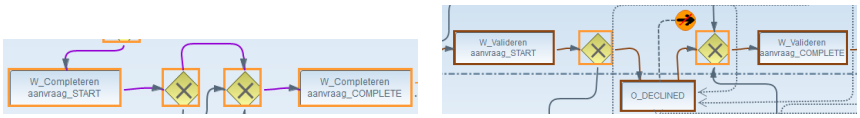
A_SUBMITTED	A_PARTLY SUBMITTED	W_Afhandelen leads_SCHEDULE	W_Afhandelen leads_START	...	W_Completeren aanvraag_START	≫	≫	W_Completeren aanvraag_COMPLETE	...	O_DECLINED ...
A_SUBMITTED	A_PARTLY SUBMITTED	W_Afhandelen leads_SCHEDULE	W_Afhandelen leads_START	...	W_Completeren aanvraag_START	A_ACCEPTED	A_DECLINED	W_Completeren aanvraag_COMPLETE	...	≫ ...

Fig. 6: A (portion of) alignment of the log trace in Fig. 4b and the process model in Fig. 2.

to pinpoint the deviations causing nonconformity between the observed and prescribed behavior. Fig. 6 shows a sample alignment between the log trace in Fig. 4b and the Petri net obtained from the SecBMPN2-ml model in Fig. 2. The top row of the alignment shows the sequence of activities in the run of the net; the bottom row shows the sequence of events in the log trace. Deviations are explicitly shown by columns that contain ≫. For example, the 7th and 8th columns in the alignment show moves on logs for activities A_ACCEPTED and A_DECLINED, indicating that these events occur in the log trace although they are not allowed according to the net. The 11th column shows a move on model for O_DECLINED, indicating that this activity must occur in the log trace according to the net but it was not executed. Other columns show that the events in the log trace match the activities in the run of the net (i.e., *synchronous moves*).

Log trace transformation Log traces that deviate from the process model are transformed into SecBPMN2 models using the original SecBPMN2 business process as a template. Fig. 7 shows two portions of the SecBPMN2 model generated from the trace in Fig. 4b, in which the deviations captured in the alignment of Fig. 6 are highlighted to easy inspection. In the figure, process elements are represented using the following color code: (i) the elements executed according to the process model are in orange; (ii) the control flows indicating the presence of moves on the logs are in purple; (iii) the elements for which a move on mode occurred are in brown. For example, Fig. 7a shows that some activities were executed between activities W_Completeren aanvraag_START and W_Completeren aanvraag_COMPLETE (control flows highlighted in purple). Fig. 7b shows another part of the same model where activity O_DECLINED was not executed in the log trace examined (activity highlighted in brown). Through this view, an analyst can determine where deviations occurred in single traces.

To provide an overview of the deviations in the process executions recorded in the log, we aggregate all deviating traces in a single SecBPMN2 model, thus provide analysts with



(a) (b)

Fig. 7: Portion of the SecBPMN2 model corresponding to the log trace in Fig. 4b

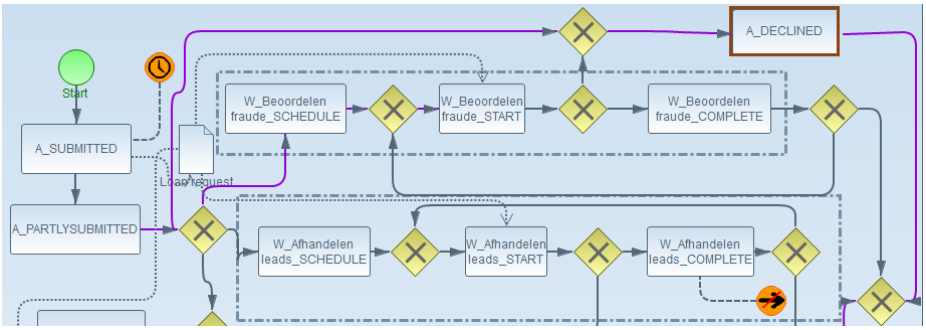


Fig. 8: Visualization of the aggregated deviating traces on the SecBPMN2 process model

an easy understanding of where deviations took place in the process. The visualization consists of the original SecBPMN2 process where the process elements for which a deviation happened in at least one process execution are highlighted in purple or brown depending on the type of deviation as shown in Fig. 8.

Security policy checking The annotated SecBPMN2 model generated in the previous step shows all deviations form the prescribed behavior. In this work, we are particularly interested in security-critical deviations. To this end, deviating traces, represented as SecBPMN2 models, are verified using the SecBPMN2 verification engine to identify which log traces violate any security policy. Security-critical traces, and their aggregation, are shown to analysts using the same color code used in the previous step (see Fig. 7 and Fig. 8). An analysis of the log traces in Fig. 4 is presented in the next section.

4 Evaluation

We have implemented the proposed approach as an extension of STS-Tool [27]. STS-Tool is a software tool that permits to graphically design business processes and security policies using SecBPMN2, and to verify business processes against security policies. We have extended the tool by: (i) implementing the transformation of SecBPMN2 models into Petri nets; (ii) integrating alignment-based techniques to identify process executions that deviate from the process specification; (iii) adapting its verification engine to identify security-critical traces; (iv) extending the graphical editor to show aggregated traces in order to easily understand where security-critical deviations happened in business processes.

To evaluate the capabilities of our approach in the identification of security-critical deviations in real-life settings, we used the event log recording the loan management

# complaint log traces	# noncomplaint log traces	# synchronous move	# move on log	# move on model	# $policy_1$ violation	# $policy_2$ violation	# $policy_3$ violation	# $policy_4$ violation
4237	8850	232,438	29,762	28,541	3	25	30	0

Table 1: Analysis of real-life data

process described in Section 3. The event log contains 262,200 events recorded in 13,087 log traces. We applied the proposed approach to the event log, the process model in Fig. 2 and the policies in Fig. 3. Table 1 reports the results of the analysis. In total, 8,850 log traces deviated from the process model. Among the identified deviations, 29,762 are moves on log and 28,541 are moves on model. In total, 58 violations of the defined security policies were identified. To illustrate the insights provided by our the method, we next analyze the three log traces shown in Fig. 4.

Fig. 4a shows a log trace that violates $Policy_1$. According to this policy, the submission and approval of a loan application have to be handled by two different users. However, in this trace, both activities were executed by the same user. By analyzing the log, we observed that this user performed many more activities than other users (17.42% of all executed activities were performed by this user) and these activities were performed every day and at any time. We speculate that this user might actually be a software application. A closer look at the log also showed that only a limited number of users (9 over 68 users) had approved a loan and the aforementioned user approved only three loan requests (as average, 249 loan applications by each user). Although we were not able to validate this finding with domain experts,[4] we believe that further investigation is required to understand the role of this user and its responsibilities and, thus, to assess the security impact of this deviation. Fig. 4b shows a log trace that violates $Policy_2$. According to this policy, applications must not be declined immediately after being accepted. However, this log trace shows that after collecting all required information, the application was denied without sending any offers to the client or assessing the application. Note that the amount of the requested loan is equal to €1000. Among 361 submitted applications with equal or lower amount, only 5 applications were approved, while in general 17% of all applications were approved. This may indicate that the financial institute tends to not grant loans with low amounts to clients and declines most of these requests. Fig. 4c shows a log trace that violates $Policy_3$. According to this policy, applications suspected to be a fraud must not be approved. In fact, according to the process model, these applications must be either terminated or declined. However, this log trace shows that a suspicious application was approved.

These results confirm that our tool is able to identify security-critical deviations in real-life settings. It is worth mentioning that analyzing all deviations from the process specification is not a trivial task and can require a significant amount of time and resources. A main advantage of our tool is that it enables analysts to focus on security-critical deviations. In particular, the tool offers visualization capabilities to inspect where security-critical deviations occurred along with information about the type of security concerns (i.e., which security policy was violated). We remark that our tool aims to assist and facilitate

[4] The 2012 BPI Challenge made a real log available but the log is anonymized. Also the name of the company providing the log is not disclosed.

analysts in the identification and analysis of security-critical deviations rather than taking over their responsibilities. As mentioned above, the identified deviations need to interpreted based on domain knowledge, although this task is eased by the insights provided by the tool.

The application of the tool to the case study also allowed us to perform a preliminary evaluation of its effectiveness and usability. In particular, the visualization of aggregated deviating traces into a SecBPMN2 diagram provides an intuitive view of where deviations occurred and scales well with a large number of traces, as in our case study. The visualization of single deviating traces is effective since the tool shows the business process, as drawn by the analyst, but includes only the process elements occurring in the trace. However, we realized that the usability of the tool could be improved by providing additional features for the filtering and selection of deviating traces.

5 Related work

The goal of this paper is to guarantee the compliance of business processes during the entire lifecycle of the systems. Approaches to compliance checking in the area of business processes can be grouped into two main categories: *forward* and *backward* compliance checking. To the best of our knowledge, this is the first work that aims to reconcile these two orthogonal approaches.

Forward compliance checking aims to ensure that business processes models comply with a given set of requirements at design time. A number of approaches have been proposed to verify the compliance of process models, before they are deployed and executed. For instance, SecureBPMN [8] extends BPMN with access control and information flow constraints. It uses the hierarchic structure of the organization, in which the business process will be executed, to help security designers to define security properties such as binding of duty [17] and separation of duty [17, 29]. However, SecureBPMN is limited in that it does not allow the specification of other fundamental security concepts such as confidentiality, non-repudiation and availability. Beeri et al. [4] propose BP-QL, a pattern-based graphical query language for business processes. They also provide software tooling to determine the compliance of a business process–defined using WS-BPEL [20]– with behavioral patterns. The use of WS-BPEL, a machine-readable standard, hinders the readability of business processes, especially with real case scenarios, where business process easily reach hundreds of elements. APQL [16] is a textual query language, based on 20 predicates that can be composed to create complex queries. This approach suffers scalability issues: the definition of complex queries is challenging and error-prone due to its complexity. Moreover, to the best of our knowledge, this approach is not tool-supported. Wolter et al. [37] propose a modeling language for business processes and business security concepts, to graphically define security specifications. They also develop a framework that transforms security goals in security policies specified in XACML [21] and Rampart [31]. The framework automatically extracts specifications of security mechanisms aiming at the enforcement of security goals, but it does not allow security experts to compose security goals and, therefore, to create complex security policies. FPSPARQL [5] is a query language that allows defining queries using a formal textual language. FPSPARQL focuses on the analysis of business processes mined from action logs, hence making it impossible

to directly define processes; in addition, it does not address security concerns. In this work, we have adopted SecBPMN2 for the definition of secure-by-design processes. It provides a modeling language for the specification of business processes with security annotations, and a modeling language for the specification of security policies along with capabilities to verify whether a given business process satisfies the defined policies.

Backward compliance checking aims to evaluate the compliance of process executions recorded in an event log with the prescribed behavior. Existing approaches [1, 3, 6, 23–25, 34] can be classified in two main streams, depending on the representation of the prescribed behavior. One stream comprises approaches that assess the compliance of process executions with business rules. Ramezani et al. [24] encode business rules into Petri net patterns and employ alignments to identify possible deviations from them. In recent years, many researchers have focused on conformance checking with respect to declarative models. For example, Chesani et al. [10] represent the prescribed behavior using declarative reactive business rules. This approach maps business rules to Abductive Logic Programming, and Prolog is used to check whether business rules are fulfilled. Montali et al. [19] propose to encode business rules in Linear Temporal Logic and evaluate them using automata. However, these approaches do not support time or data perspectives. Declarative constraints have been extended to support the time [18] and data [11] perspectives separately. Burattin et al. [9] propose an approach to support different process perspectives at the same time.

Another stream includes approaches that assess the compliance with respect to a process model. Among these approaches, Petkovič et al. [23] generate the transition system of a process model and verify whether a log trace corresponds to a valid trace of the transition system. Rozinat et al. [25] propose a token-based approach to measure the conformance between an event log and a Petri net. This measurement is based on the number of missing and remaining tokens after replaying log traces on the net. Banescu et al. [3] extend the work in [25] to identify different types of deviations such as replacement and re-ordering by analyzing the configuration of missing and remaining tokens using predefined patterns. These approaches, however, do not provide accurate diagnostics on deviations. In this work, we adopt the notion of alignment [33] for conformance checking. Alignments offer a robust approach to conformance checking able to provide richer and more accurate diagnostics and have been widely used for various purposes such as performance analysis [33], process-model repairing [15] and security analysis [1].

6 Conclusion

This paper proposes a methodological approach for the identification of security-critical deviations in process executions, which leverages off-the-shelf techniques for forward and backward conformance checking. The use of forward compliance checking allows the specification of secure-by-design business processes, whereas the use of backward compliance checking allows the identification of deviations in the process executions. Moreover, we exploit the verification capabilities provided by forward compliance checking to identify those deviations that are security critical. Our approach is fully supported by an extended version of the STS-Tool. The tool offers visualization capabilities to inspect the identified deviations and provide diagnostics on security concerns (i.e., which policy was

violated). This view provides analysts with insights into security issues, thus, helping them in taking the necessary measures to mitigate the impact of security incidents. We evaluated the tool with a case study using a real log recording the loan management process of a Dutch financial institute. The results of the evaluation show that our approach is able to identify a number of security concerns, and the visualization of the single and aggregated deviations help in the understanding of critical-security issues in process executions.

Although the evaluation showed that our approach is able to detect security-critical deviations in real settings, we plan to conduct a more extensive evaluation of the usability and effectiveness of the tool by involving practitioners. Moreover, we plan to extend our method in order to support the analysis of a larger range of security constraints in process executions. This includes the design of methods to extract security-related information (e.g., security mechanisms employed, accessed data objects) from log files.

Acknowledgments This work has been partially funded by the NWO CyberSecurity programme under the PriCE project, by the DITAS project funded by the European Union's Horizon 2020 research and innovation programme under grant agreement RIA 731945.

References

1. M. Alizadeh, X. Lu, D. Fahland, N. Zannone, and W. M. van der Aalst. Linking data and process perspectives for conformance analysis. *Computers & Security*, 2017.
2. R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Publishing, 2 edition, 2008.
3. S. Banescu, M. Petkovic, and N. Zannone. Measuring privacy compliance using fitness metrics. In *Business Process Management*, LNCS 7481, pages 114–119. Springer, 2012.
4. C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo. Querying business processes with BP-QL. *Information Systems*, 33(6):477–507, 2008.
5. S. Beheshti, B. Benatallah, H. Motahari-Nezhad, and S. Sakr. A Query Language for Analyzing Business Processes Execution. In *BPM*, pages 281–297. Springer, 2011.
6. D. Borrego and I. Barba. Conformance checking and diagnosis for declarative business process models in data-aware scenarios. *Expert Systems with Applications*, 41(11):5340–5352, 2014.
7. BPI Challenge 2012. Event log of a loan application process. `http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`, 2012.
8. A. D. Brucker, I. Hang, G. Lückemeyer, and R. Ruparel. SecureBPMN: Modeling and Enforcing Access Control Requirements in Business Processes. In *SACMAT*, pages 123–126. ACM, 2012.
9. A. Burattin, F. M. Maggi, and A. Sperduti. Conformance checking based on multi-perspective declarative process models. *Expert Systems with Applications*, 65:194–211, 2016.
10. F. Chesani, P. Mello, M. Montali, F. Riguzzi, M. Sebastianis, and S. Storari. Checking compliance of execution traces to business rules. In *BPM*, pages 134–145. Springer, 2008.
11. R. De Masellis, F. M. Maggi, and M. Montali. Monitoring data-aware business constraints with finite state automata. In *ICSSP*, pages 134–143. ACM, 2014.
12. P. Delfmann, H. Dietrich, J. Havel, and M. Steinhorst. A Language-independent Model Query Tool. In *DESRIST*, pages 453–457. Springer, 2014.
13. J. Desel and W. Reisig. Place/transition Petri Nets. In *Lectures on Petri Nets I: Basic Models*, LNCS 1491, pages 122–173. Springer, 1998.
14. R. M. Dijkman, M. Dumas, and C. Ouyang. Semantics and analysis of business process models in BPMN. *Information and Software technology*, 50(12):1281–1294, 2008.

15. D. Fahland and W. M. P. van der Aalst. Model repair - aligning process models to reality. *Information Systems*, 2014.
16. A. Hofstede, C. Ouyang, M. La Rosa, L. Song, J. Wang, and A. Polyvyanyy. APQL: A Process-Model Query Language. In *Proc. of AP-BPM*, volume 159, pages 23–38, 2013.
17. N. Li, M. V. Tripunitara, and Z. Bizri. On mutually exclusive roles and separation-of-duty. *ACM Transactions on Information and System Security*, 10(2):5, 2007.
18. F. M. Maggi and M. Westergaard. Using timed automata for a priori warnings and planning for timed declarative process models. *IJCIS*, 23(01):1440003, 2014.
19. M. Montali, M. Pesic, W. M. van der Aalst, F. Chesani, P. Mello, and S. Storari. Declarative specification and verification of service choreographiess. *TWEB*, 4(1):3, 2010.
20. OASIS. Web Services Business Process Execution Language. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, Apr 2007.
21. OASIS. eXtensible Access Control Markup Language (XACML)Version 3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html, Jan 2013.
22. OMG. BPMN 2.0, Jan 2011.
23. M. Petković, D. Prandi, and N. Zannone. Purpose control: Did you process the data for the intended purpose? In *Secure Data Management*, LNCS 6933, pages 145–168. Springer, 2011.
24. E. Ramezani, V. Gromov, D. Fahland, and W. van der Aalst. Compliance checking of data-aware and resource-aware compliance requirements. In *On the Move to Meaningful Internet Systems*, pages 237–257. Springer, 2014.
25. A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008.
26. M. Salnitri, E. Paja, and P. Giorgini. Maintaining secure business processes in light of socio-technical systems' evolution. In *RE Conference Workshops*, pages 155–164. IEEE, 2016.
27. M. Salnitri, E. Paja, M. Poggianella, and P. Giorgini. STS-Tool 3.0: Maintaining Security in Socio-Technical Systems. In *Proc. of CAiSE Forum*, pages 205–212, 2015.
28. S. Sarker, S. Sarker, and A. Sidorova. Understanding business process change failure: An actor-network perspective. *Journal of Management Information Systems*, 23(1):51–86, 2006.
29. R. Simon and M. Zurko. Separation of duty in role-based environments. In *Proc. of Computer Security Foundations Workshop*, pages 183–194, 1997.
30. H. Störrle. VMQL: A Visual Language for Ad-hoc Model Querying. *Journal of Visual Languages and Computing*, 22:3–29, 2011.
31. The Apache Software Foundation. Apache Rampart website, last visited April 2016. http://axis.apache.org/axis2/java/rampart/.
32. W. M. van der Aalst. Business process management: A comprehensive survey. *ISRN Software Engineering*, 2013.
33. W. M. van der Aalst, A. Adriansyah, and B. van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Int. Rev. Data Min. and Knowl. Disc.*, 2(2):182–192, 2012.
34. W. M. van der Aalst, H. De Beer, and B. F. van Dongen. Process mining and verification of properties: An approach based on temporal logic. In *On the Move to Meaningful Internet Systems*, pages 130–147. Springer, 2005.
35. W. M. van der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
36. W. M. van der Aalst and H. Verbeek. Process discovery and conformance checking using passages. *Fundamenta Informaticae*, 131(1):103–138, 2014.
37. C. Wolter, M. Menzel, A. Schaad, P. Miseldine, and C. Meinel. Model-driven business process security requirement specification. *Journal of Systems Architecture*, 55(4):211–223, 2009.