Model-based Situational Security Analysis

Jörn Eichler and Roland Rieke

Fraunhofer Institute for Secure Information Technology SIT, Darmstadt, Germany
{joern.eichler,roland.rieke}@sit.fraunhofer.de

Abstract. Security analysis is growing in complexity with the increase in functionality, connectivity, and dynamics of current electronic business processes. To tackle this complexity, the application of models in pre-operational phases is becoming standard practice. Runtime models are also increasingly applied to analyze and validate the actual security status of business process instances. In this paper we present an approach to support not only model-based evaluation of the current security status of business process instances, but also to allow for decision support by analyzing close-future process states. Our approach is based on operational formal models derived from development-time process and security models. This paper exemplifies our approach utilizing real world processes from the logistics domain and demonstrates the systematic development and application of runtime models for situational security analysis.

Keywords: security requirements elicitation, predictive security analysis, analysis of business process behavior, security modeling and simulation, security monitoring

1 Introduction

Electronic business processes connect many systems and applications. This leads to an increasing complexity when analyzing distinctive properties of those business processes. Additionally, frequent changes to business process models are applied to address changing business needs. Current approaches apply changes to those models at runtime [4]. This situation challenges operators and participants in electronic business processes as the assessment of the status of business process instances at runtime becomes difficult. An example for these difficulties is the assessment whether instances of business processes violate security policies or might violate them in the near future.

Traditionally, approaches to security analysis of electronic business processes are executed at development-time. In this perspective, the analysis of possible violations of security policies is part of the requirements engineering process [13]. To cope with the growing complexity of the electronic business processes, the application of security models in the course of the requirements engineering process is becoming a common strategy [5]. Nevertheless, the requirements engineering process is generally limited to development-time.

Contributions. To support security analysis at runtime we utilize formal models based on development-time process and security models. On the basis of

sound methods for the elicitation and modeling of security requirements provided in [7] and an architectural blueprint described in [18], we document in this paper our approach to analyze the security status of electronic business processes. The security analysis consumes events from the runtime environment, maps those events to security events and feeds them to our runtime model, an operational finite state model. This allows to match and synchronize the state of the real process with the state of the model. Annotations of security requirements to the states of the model can now be used to check for security violations and possibly generate alarms. These alarms are then in turn converted to events and sent to the running business process. Furthermore, a computation of possible close-future behavior, which is enabled by the model of the business process, is used to evaluate possible security critical states in the near future at runtime. This knowledge about possibly upcoming critical situations can be used to raise respective predictive alarms.

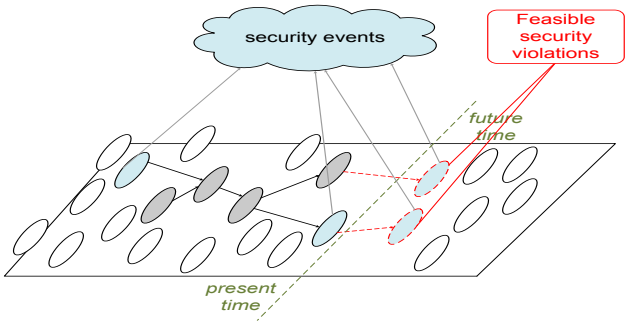

Fig. 1. Predict feasible security violations

In section 2 we provide an application scenario from the logistics domain and elicit security requirements. The formalization of the scenario is given in section 3. Section 4 analyzes the runtime operation and exemplifies generated security alerts. Section 5 reviews shortly related work to our approach. Concluding remarks and further research directions are given in section 6.

2 Application Scenario

In order to demonstrate what kind of security requirements we are able to consider and how our model-based runtime analysis is applied, we have chosen a small part of a "*Pickup*" target process which is analysed in the project Alliance Digital Product Flow (`http://www.adiwa.net/`).

2.1 Process and Event Model

The "Pickup" process is initiated when the truck driver is notified about new pickup orders. He accepts the received list of orders and the system calculates a route plan based on the addresses. When the driver arrives at a pickup address, he checks visually the packages for deviations with regard to the description in the order. In case of deviations he consults with the sender whether this package is to be transported. If the truck driver accepts the new package, the package description in the list of orders is updated accordingly. For each accepted package the system receives a confirmation that it has been loaded. The system links each loaded package and its transporting truck using the corresponding radio-frequency identification (RFID) tag identifiers. An Event-driven Process Chain (EPC) flowchart of the considered subprocess is depicted in Figure 2. Rectangles with rounded corners denote actions and chamfered rectangles denote events.

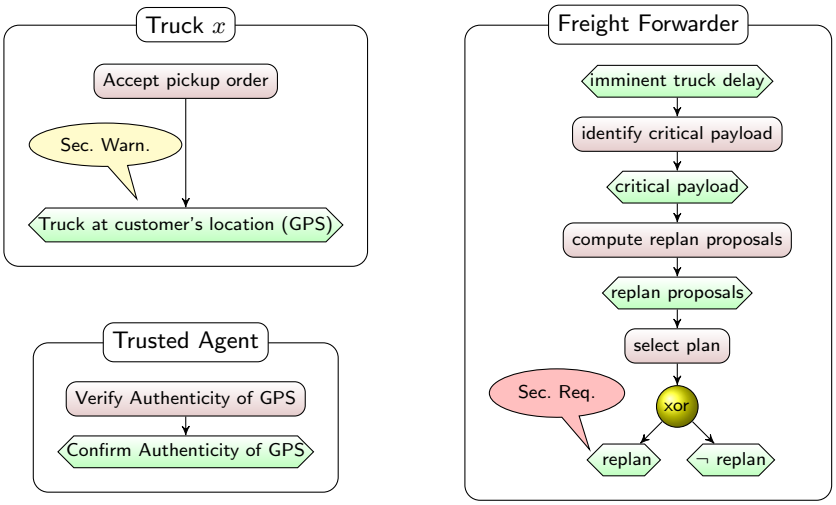

Fig. 2. Model of a part of the Pickup Process (EPC notation)

As an example of a security threatening misuse case, we consider a situation where the system performs a rescheduling because of a delay of one or more trucks on the basis of not confirmed Global Positioning System (GPS) locations. In this case there is a possibility for an attacker to send false GPS data to the system, which may result in ineffective rescheduling and possible time loss in completing the orders.

2.2 Security Requirements Elicitation

In order to derive the security requirements in the given scenario, we follow the scheme described in [7]. We assume that the functional dependencies between the actions in our scenario are given by Fig. 3.

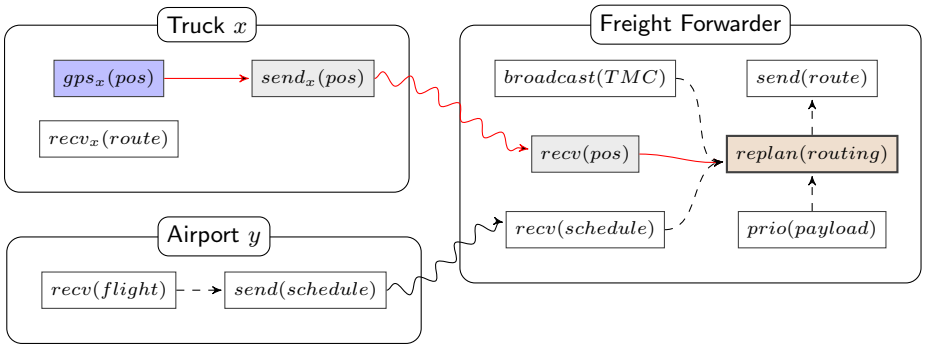

Fig. 3. Functional dependencies

We apply a general security goal: *Whenever a certain output action happens, the input actions that presumably led to it must actually have happened.* As an example for a specific security goal, in the following we will use the authenticity requirement: *Whenever a rescheduling action is performed, the GPS coordinates of each truck should be authentic for the dispatcher in terms of origin, content and time.* The formal syntax to describe these requirements in parameterized form is defined as (see [8]):

Definition 1. *$auth(a, b, P)$: Whenever an action b happens, it must be authentic for an Agent P that in any course of events that seem possible to him, a certain action a has happened.*

Therefore, our selected authenticity requirement can be written as:

$$auth(gps_x(pos), replan(routing), dispatcher). \qquad \text{(Auth 1)}$$

We will use the authenticity requirement (Auth 1) to describe the reasoning process with the help of an appropriate operational model.

There are of course many other security requirements necessary in this scenario. For example, *while loading a package on the truck the RFID data and the truck driver should be authentic in terms of content and identification number.* Analysis and application in our situational security analysis follow the same procedure as for (Auth 1). Therefore, we will exemplify only (Auth 1) in the following.

3 Formal Model

In order to analyze the system behavior with tool support, an appropriate formal representation has to be chosen. In our approach, we use an operational finite state model of the behavior of the given process which is based on *Asynchronous Product Automata (APA)*, a flexible operational specification concept

for cooperating systems [16]. An APA consists of a family of so called *elementary automata* communicating by common components of their state (shared memory). We now introduce the formal modeling techniques used, and illustrate the usage by our application example.

Definition 2 (Asynchronous Product Automaton (APA)). *An* Asynchronous Product Automaton $\mathbb{A} = ((Z_s)_{s\in\mathbb{S}}, (\Phi_t, \Delta_t)_{t\in\mathbb{T}}, N, q_0)$ *consists of a family of* state sets $Z_s, s \in \mathbb{S}$, *a family of* elementary automata (Φ_t, Δ_t), *with* $t \in \mathbb{T}$, *a* neighborhood relation $N : \mathbb{T} \to \mathfrak{P}(\mathbb{S})$ *and an* initial state $q_0 = (q_{0s})_{s\in\mathbb{S}} \in \bigtimes_{s\in\mathbb{S}}(Z_s)$. $\mathbb{S}$ *and* $\mathbb{T}$ *are index sets with the names of state components and of elementary automata and* $\mathfrak{P}(\mathbb{S})$ *is the power set of* $\mathbb{S}$. *For each elementary automaton* (Φ_t, Δ_t) *with* Alphabet Φ_t, *its* state transition relation *is* $\Delta_t \subseteq \bigtimes_{s\in N(t)}(Z_s) \times \Phi_t \times \bigtimes_{s\in N(t)}(Z_s)$. *For each element of* Φ_t *the state transition relation* Δ_t *defines state transitions that change only the state components in* $N(t)$. *An APA's (global)* states *are elements of* $\bigtimes_{s\in\mathbb{S}}(Z_s)$. *To avoid pathological cases it is generally assumed that* $N(t) \neq \emptyset$ *for all* $t \in \mathbb{T}$. *An elementary automaton* (Φ_t, Δ_t) *is* activated *in a state* $p = (p_s)_{s\in\mathbb{S}} \in \bigtimes_{s\in\mathbb{S}}(Z_s)$ *as to an* interpretation $i \in \Phi_t$, *if there are* $(q_s)_{s\in N(t)} \in \bigtimes_{s\in N(t)}(Z_s)$ *with* $((p_s)_{s\in N(t)}, i, (q_s)_{s\in N(t)}) \in \Delta_t$. *An activated elementary automaton* (Φ_t, Δ_t) *can execute a state transition and produce a* successor state $q = (q_r)_{r\in\mathbb{S}} \in \bigtimes_{s\in\mathbb{S}}(Z_s)$, *if* $q_r = p_r$ *for* $r \in \mathbb{S} \setminus N(t)$ *and* $((p_s)_{s\in N(t)}, i, (q_s)_{s\in N(t)}) \in \Delta_t$. *The corresponding state transition is* $(p, (t, i), q)$.

A simplified model of the part of the "Freight Forwarder" business process shown in Fig 2 contains the APA state components *pstate* and *event* representing the current process state and event. Formally, $\mathbb{S} = \{pstate, event\}$, with $Z_{event} = \{imminent\_truck\_delay, \ldots, replan, \neg replan\}$, $Z_{pstate} = \ldots$.

The *elementary automata* $\mathbb{T} = \{identify\_critical\_payload, \ldots, select\_plan\}$ represent the possible actions that the systems can take. The neighborhood relation between elementary automata and state components of the APA model is depicted by the edges in Fig. 4.

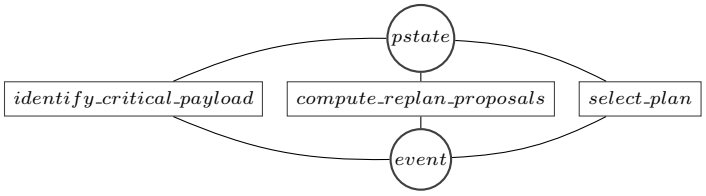

Fig. 4. Elementary automata and state components in the APA process model

Formally, the behavior of our operational APA model of the business process is described by a reachability graph. In the literature this is sometimes also referred to as labeled transition system (LTS).

Definition 3 (Reachability graph). *The behavior of an APA is represented by all possible coherent sequences of state transitions starting with initial state* q_0. *The sequence* $(q_0, (t_1, i_1), q_1)(q_1, (t_2, i_2), q_2) \ldots (q_{n-1}, (t_n, i_n), q_n)$ *with* $i_k \in \Phi_{t_k}$ *represents one possible sequence of actions of an APA. State transitions* $(p, (t, i), q)$ *may be interpreted as labeled edges of a directed graph whose nodes are the states of an APA:* $(p, (t, i), q)$ *is the edge leading from* p *to* q *and labeled by* (t, i). *The subgraph reachable from* q_0 *is called* reachability graph *of an APA.*

We use the *SH verification tool* [16] to analyse the process model. This tool provides components for the complete cycle from formal specification to exhaustive validation as well as visualisation and inspection of computed reachability graphs and minimal automata. The applied specification method based on APA is supported. The tool manages the components of the model, allows to select alternative parts of the specification and automatically *glues* together the selected components to generate a combined model of the APA specification.

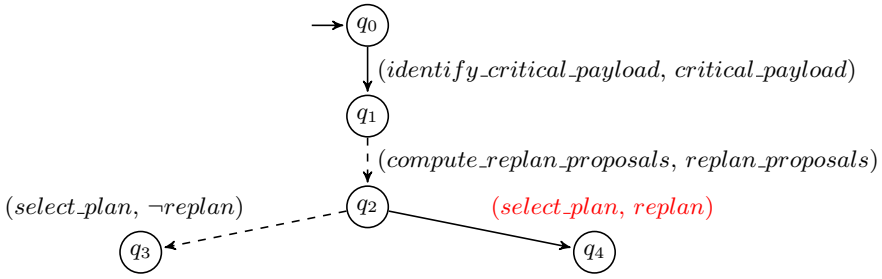

Fig. 5. Close-future (3 Steps) Reachability Analysis

Figure 5 shows the initial part of the reachability graph resulting from the analysis of the model when reaching the part of the business process of the freight forwarder shown in Fig. 2. An example for a state transition of the model in this situation is: (q_0,($identify\_critical\_payload$, $critical\_payload$),q_1). Please note that there are two different transitions from the state q_2 because the interpretation of a variable can have the values $replan$ or $\neg\ replan$, respectively.

4 Runtime Operation and Generated Alerts

During runtime, the events from the business process are used to synchronize the state of the model with the real process. In our exemplary setup, the events are produced by a Complex Event Processing (CEP) engine which is provided by one of the project partners. The events are described by an XML schema and communicated by the Java Message Service (JMS). The events from the event bus are used to provide the information about the state and input to the business process. In our finite state model, this information is represented in the state

components *pstate* and *event* (cf. Fig. 4). This constitutes the initial state of the model from which a simulation is then started. In addition to the predicted system behavior, we also need the information on the security requirements in order to identify critical situations. In [18] we proposed to use APA to specify meta-events, which match security critical situations, to generate alerts. However, since this is slow and not easily usable by end-users, we decided to build the matching algorithm directly into the SH verification tool. We use monitor automata [22] to specify the security requirements graphically. These automata monitor the behaviour of the abstract system during the run of the simulation and provide interfaces to trigger alerts. This concept could be further extended to make use of the built-in temporal logic based reasoning component if more complex reasoning is necessary.

4.1 Security Reasoning – No Authenticity Approval of GPS Event

In order to demonstrate the use of process models at runtime, let us assume the following situation. We are currently at logical time 0 as depicted on the timeline in Figures, 7, 8, 9, 10. We further assume that the trusted agent inspects the events generated by GPS units of the trucks and sends additional events which attest to the authenticity of each GPS event within a timeframe of 2 logical time units. Please note that it is also a possibility that the trusted agent would filter the events and only let authentic events pass the filter. We furthermore assume that we know from the analysis of dependencies of actions and specifically from the requirement (Auth 1) that whenever a rescheduling action is performed, the GPS coordinates of each truck should be authentic for the process planner in terms of origin, content and time.

We now describe the reasoning process where the authenticity of the GPS event is not approved by the trusted agent. In the diagrams we use pentagon symbols to depict events on the event bus such as GPS information and we use triangles to depict Security Warnings (SW), Predictive Security Alerts (PSA) and Security Alerts (SA) generated by the reasoning process.

Step 1: A GPS event is received

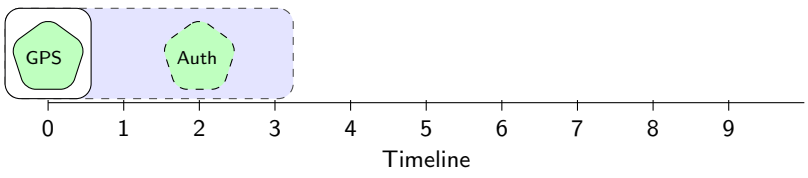

Fig. 6. Security Reasoning - Authenticity Approval of GPS Event - step 1

Figure 6 shows the situation when a GPS event is received. This event is matching a precondition in the requirement pattern: *GPS needs confirmation*

in 2 steps. This requirement (warn-level) is triggered by the GPS event. The reachability analysis reveals no critical actions within the scope (3 steps) of the analysis. We conclude from Fig. 6 that everything is OK at this point. A future event might confirm authenticity of the GPS location received.

Step 2: Confirmation of GPS event not received

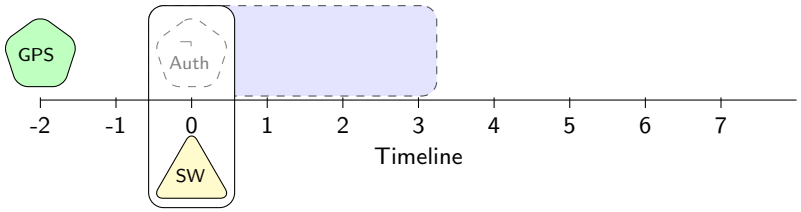

Fig. 7. Security Reasoning - No Authenticity Approval of GPS Event - step 2

Figure 7 shows the situation when an expected event from the trusted agent, namely the authenticity approval of this GPS event is recognized as missing. The missing event indicates a broken security requirement: *GPS needs confirmation in 2 steps*. The reachability analysis in this situation shows that no other security requirement will be triggered within the scope of the analysis. However, some forthcoming security relevant action might require authenticity of this GPS event. Therefore, an alert action associated with a broken warn-level requirement, such as issuing a security warning (SW), is now triggered.

Step 3: Replan event in analysis scope

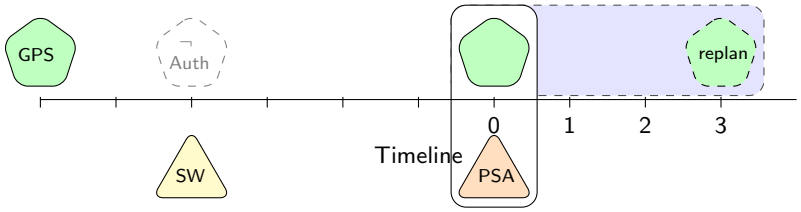

Fig. 8. Security Reasoning - No Authenticity Approval of GPS Event - step 3

In Fig. 8, an arbitrary event is received from which, in one possible execution sequence of the business process, a *replan* event is reachable within the scope of the analysis. In our scenario *imminent_ truck_delay* is such an event. The reachability graph is similar to the one depicted in Fig. 5. It shows that the *select_plan* action may happen in the future if *replan* is chosen. But there

is another possible path in the graph where *replan* is not chosen. The *replan* event in the prediction scope is matching a precondition in a requirement pattern: $auth(GPS, replan, dispatcher)$, but the GPS event is not approved to be authentic. Therefore, a *replan* event with broken security requirement is possible. An action associated with this (possibly) broken alert-level requirement, such as issuing a predictive security alert (PSA), is now executed.

Step 4a: Expected *replan* event received

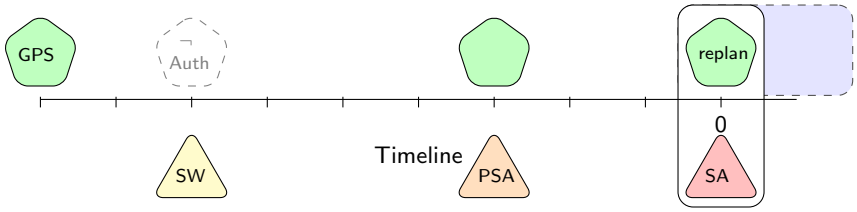

Fig. 9. Security Reasoning - No Authenticity Approval of GPS Event - step 4a

Figure 9 shows the situation when a *replan* event is received as predicted (cf. Fig. 5 transition $q_2 \rightarrow q_4$). At this time we know that the security requirement (Auth 1) is broken. Therefore, an action associated with a broken alert-level requirement, such as issuing a security alert (SA), is now executed.

Step 4b: Predicted *replan* event not received after step 3

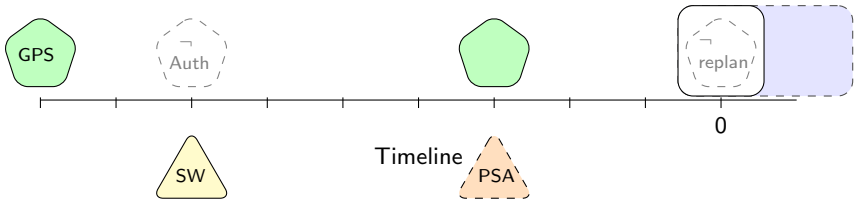

Fig. 10. Security Reasoning - No Authenticity Approval of GPS Event - step 4b

Figure 10 shows the situation when a *replan* event is not received as expected (cf. Fig. 5 transition $q_2 \rightarrow q_5$). In this case, we know that the issued predictive security alert (PSA) was a "False Positive", so a corrective action may be necessary. Corrective actions might be the reduction of a general security warning level or lifting of restrictions on the business process depending on the operating environment. However, the security warning issued in step 2 is still valid because some future event might require authenticity of the GPS event.

5 Related Work

The work presented here combines specific aspects of security analysis with generic aspects of process monitoring, simulation, and analysis. The background of those aspects is given by the utilization of models at runtime [6]. A blueprint for our architecture of predictive security analysis is given in [18].

Security analysis *at development-time* to identify violations of security policies is usually integrated in the security requirements engineering process. An overview of current security requirements engineering processes is given in [5,13]. The security requirements elicitation methods developed in [7] are used in section 2 to derive the requirements which are needed to assess possible security policy violations at runtime. A formalized approach for security risk modeling in the context of electronic business processes is given in [21]. It touches also the aspect of simulation, but does not incorporate the utilization of runtime models. Approaches that focus security models at runtime are given in [14] or in [12]. Morin et. al [14] propose a novel methodology to synchronize an architectural model reflecting access control policies with the running system. Therefore, the methodology emphasizes policy enforcement rather than security analysis. The integration of runtime and development-time information on the basis of an ontology to engineer industrial automation systems is discussed in [12].

Process monitoring has gained some popularity recently in the industrial context prominently accompanied with the term Business Activity Monitoring (BAM). The goal of BAM applications, as defined by Gartner Inc., is to process events, which are generated from multiple application systems, enterprise service buses or other inter-enterprise sources in real-time in order to identify critical business key performance indicators and get a better insight into the business activities and thereby improve the effectiveness of business operations [11]. Recently, runtime monitoring of concurrent distributed systems based on linear temporal logic (LTL), state-charts, and related formalisms has also received a lot of attention [9,10]. However, these works are mainly focused on error detection, e.g., concurrency related bugs. A classification for runtime monitoring of software faults is given in [1]. Patterns to allow for monitoring security properties are developed in [20]. In the context of BAM applications, in addition to these features we propose a *close-future* security analysis as it is detailed in section 4. Our analysis provides information about possible security policy violations reinforcing the security-related decision support system components.

Different categories of tools applicable for simulation of business processes including process modeling tools are based on different semi-formal or formal methods such as Petri Nets [3] or Event-driven Process Chains (EPC) [2]. Some process management tools such as FileNet [15] offer a simulation tool to support the design phase. Also, some general-purpose simulation tools such as CPNTools [19] were proven to be suitable for simulating business processes. However, independently from the tools and methods used, such simulation tools concentrate on statistical aspects, redesign and commercial optimization of the business process. On the contrary, we propose an approach for *on-the-fly* dynamic simulation and analysis on the basis of operational APA models detailed in section 3. This

includes consideration of the current process state and the event information combined with the corresponding steps in the process model.

6 Conclusions and Further Work

In this paper we demonstrated the application of runtime models to analyze the security status of business processes and to identify possible violations of the security policy in the near future. Therefore, we started with a business process model from the logistics domain and analyzed corresponding security requirements. Utilizing both development-time models we derived a runtime model. The runtime model consumes events from the runtime environment, evaluates current violations of the security policy, and identifies close-future violations of the security policy. Within the logistics domain we applied our approach to identify situations in which an attacker might try to disrupt or degrade the process performance. By issuing predictive security alerts, users or operators (in this case: the dispatcher in the logistic process) are able to act securely without the need to understand the security policy or infrastructure in detail.

Other novel uses of such models at runtime can enable anticipatory impact analysis, decision support and impact mitigation by adaptive configuration of countermeasures. The project MASSIF (`http://www.massif-project.eu/`), a large-scale integrating project co-funded by the European Commission, addresses these challenges within the management of security information and events in service infrastructures. In MASSIF [17] we will apply the presented modeling concept in four industrial domains: (i) the management of the Olympic Games IT infrastructure; (ii) a mobile phone based money transfer service, facing high-level threats such as money laundering; (iii) managed IT outsource services for large distributed enterprises; and (iv) an IT system supporting a critical infrastructure (dam).

Acknowledgments. The work presented here was developed in the context of the project MASSIF (ID 257475) being co-funded by the European Commission within the Seventh Framework Programme and the project Alliance Digital Product Flow (ADiWa) (ID 01IA08006F) which is funded by the German Federal Ministry of Education and Research.

References

1. Delgado, N., Gates, A., Roach, S.: A taxonomy and catalog of runtime software-fault monitoring tools. IEEE Transactions on Software Engineering 30(12), 859–872 (2004)
2. Dijkman, R.M.: Diagnosing differences between business process models. In: Business Process Management (BPM 2008). LNCS, vol. 5240, pp. 261–277. Springer (2008)
3. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information and Software Technology 50(12), 1281–1294 (2008)

4. Döhring, M., Zimmermann, B., Karg, L.: Flexible workflows at design- and runtime using BPMN2 adaptation patterns. In: Business Information Systems (BIS 2011), LNBIP, vol. 87, pp. 25–36. Springer (2011)
5. Fabian, B., Gürses, S., Heisel, M., Santen, T., Schmidt, H.: A comparison of security requirements engineering methods. Requirements engineering 15(1), 7–40 (2010)
6. France, R., Rumpe, B.: Model-driven development of complex software: A research roadmap. In: Future of Software Engineering. pp. 37–54. IEEE (2007)
7. Fuchs, A., Rieke, R.: Identification of Security Requirements in Systems of Systems by Functional Security Analysis. In: Architecting Dependable Systems VII, LNCS, vol. 6420, pp. 74–96. Springer (2010)
8. Gürgens, S., Ochsenschläger, P., Rudolph, C.: On a formal framework for security properties. Computer Standards & Interfaces 27, 457–466 (2005)
9. Kazhamiakin, R., Pistore, M., Santuari, L.: Analysis of communication models in web service compositions. In: World Wide Web (WWW 2006). pp. 267–276. ACM (2006)
10. Massart, T., Meuter, C.: Efficient online monitoring of LTL properties for asynchronous distributed systems. Tech. rep., Université Libre de Bruxelles (2006)
11. McCoy, D.W.: Business Activity Monitoring: Calm Before the Storm. Gartner Research (2002)
12. Melik-Merkumians, M., Moser, T., Schatten, A., Zoitl, A., Biffl, S.: Knowledge-based runtime failure detection for industrial automation systems. In: Workshop Models@run.time. pp. 108–119. CEUR (2010)
13. Mellado, D., Blanco, C., Snchez, L.E., Fernndez-Medina, E.: A systematic review of security requirements engineering. Computer Standards & Interfaces 32(4), 153–165 (2010)
14. Morin, B., Mouelhi, T., Fleurey, F., Le Traon, Y., Barais, O., Jézéquel, J.M.: Security-driven model-based dynamic adaptation. In: Automated Software Engineering (ASE 2010). pp. 205–214. ACM (2010)
15. Netjes, M., Reijers, H., Aalst, W.P.v.d.: Supporting the BPM life-cycle with FileNet. In: Exploring Modeling Methods for Systems Analysis and Design (EMMSAD 2006). pp. 497–508. Namur University Press (2006)
16. Ochsenschläger, P., Repp, J., Rieke, R., Nitsche, U.: The SH-Verification Tool Abstraction-Based Verification of Co-operating Systems. Formal Aspects of Computing 10(4), 381–404 (1998)
17. Prieto, E., Diaz, R., Romano, L., Rieke, R., Achemlal, M.: MASSIF: A promising solution to enhance olympic games IT security. In: International Conference on Global Security, Safety and Sustainability (ICGS3 2011) (2011)
18. Rieke, R., Stoynova, Z.: Predictive security analysis for event-driven processes. In: Computer Network Security, LNCS, vol. 6258, pp. 321–328. Springer (2010)
19. Rozinat, A., Wynn, M.T., van der Aalst, W.M.P., ter Hofstede, A.H.M., Fidge, C.J.: Workflow simulation for operational decision support. Data & Knowledge Engineering 68(9), 834–850 (2009)
20. Spanoudakis, G., Kloukinas, C., Androutsopoulos, K.: Towards security monitoring patterns. In: Symposium on Applied computing (SAC 2007). pp. 1518–1525. ACM (2007)
21. Tjoa, S., Jakoubi, S., Goluch, G., Kitzler, G., Goluch, S., Quirchmayr, G.: A formal approach enabling risk-aware business process modeling and simulation. IEEE Transactions on Services Computing 4(2), 153–166 (2011)
22. Winkelvos, T., Rudolph, C., Repp, J.: A Property Based Security Risk Analysis Through Weighted Simulation. In: Information Security South Africa (ISSA 2011). IEEE (2011)