Security Compliance Tracking of Processes in Networked Cooperating Systems

Roland Rieke[1, 2*], Maria Zhdanova[1], and Jürgen Repp[1]
[1]*Fraunhofer Institute SIT, Darmstadt, Germany*
{roland.rieke, maria.zhdanova, juergen.repp}@sit.fraunhofer.de
[2]*Philipps-Universität Marburg, Germany*

Abstract

Systems of systems that collaborate for a common purpose are called cooperating systems. Typical examples of novel cooperating systems are electronic health systems and electronic money transfer systems but also critical infrastructures, such as future vehicular ad hoc networks and distributed air traffic management systems. Business processes and technical workflows control the cooperation of the networked systems. Important safety and security goals of the applications, business goals, and external compliance requirements create security obligations for such processes. These processes must not only be secure, they must be demonstrably so. To support this, we present an approach for security compliance tracking of processes in networked cooperating systems using an advanced method of predictive security analysis at runtime. At that, operational models are utilized for: (a) tracking conformance of process behavior with respect to the specification, (b) detection of behavior anomalies which indicate possible attacks, (c) tracking compliance of process behavior with respect to safety and security requirements, and (d) prediction of possible violations of safety and security policies in the near future. We provide an extensive background analysis, introduce the model-based conformance tracking and uncertainty management algorithm, and describe security compliance tracking and model-based behavior prediction. We demonstrate the implementation of the proposed approach on a critical infrastructure scenario from a European research project.

Keywords: Predictive Security Analysis, Model-based Process Behavior Analysis, Security Modeling and Simulation, Security Compliance Monitoring, Security Information and Event Management, Governance and Compliance, Security of Critical Infrastructures

1 Introduction

This article is based on previous publications [1, 2, 3] about our Predictive Security Analysis at Runtime (PSA@R) approach. PSA@R observes the operation of processes and triggers mitigating actions in case of deviations from the pre-planned process behavior, violations of compliance requirements, and predicted critical states.

Systems of systems that *collaborate for a common purpose* are called Cooperating Systems (CS). In general, CS are specific networked systems of systems which are characterized by freedom of decision and loose coupling of their components [4]. The cooperation between these systems is often controlled by technical workflows and business processes. Business processes as such are the most important assets of enterprises and thus security compliance tracking of these processes is required. Furthermore, security breaches might violate safety properties, which are, in particular, in case of technical processes in critical infrastructures, of utmost importance for the operators and users of these infrastructures, e.g., in vehicular

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, volume: 6, number: 2, pp. 21-40
*Corresponding author: Fraunhofer Institute SIT, Rheinstrasse 75, 64295 Darmstadt, Germany, Tel: +49-6151-869-284, Web: http://rieke.link/

ad hoc networks and air traffic management systems. Processes in such CS must not only be secure, they must be *demonstrably* so. *Formal security models* are thus needed to be able to convince others of the security of a system [5].

Therefore, our PSA@R approach is based on a formal security model. A runtime check, whether the observed processes in CS behave according to the given rules, allows a timely decision and thus enables proactive reactions for risk and impact mitigation. The knowledge on expected process behavior in process-aware CS can be utilized to detect anomalies. Anomalies can be found by tracking the conformance of a running process to a formal operational model of the process control flow based on the observation of the events from the CS. A deviation of observed process behavior compared to the planned (prescribed) behavior, given by the process model and current state, indicates an anomaly. In this case, *uncertainty management* is important to classify the anomaly, which can be caused by malicious activities but also by mistakes in the process model, adaptations of process behavior or faulty event communication. For the prediction of close-future critical states, the process perspective on behavior of CS taken by our approach is very important, because the limitations on control flows given by process specifications [6] limits the nondeterminism and thus the state space explosion problems in behavior prediction for such process-aware CS. Therefore, possible close-future process actions can be predicted based on the operational process specification and the current process state as reflected in the model. This knowledge about the expected behavior is used to predict upcoming critical states regarding given security compliance rules.

The conceptual idea of PSA@R was first introduced in [1] and results of a first validation in context of business process security have been reported in [2]. This article mainly builds on [3], which refined PSA@R by detailing the monitoring formalism, implementation, and evaluation of the approach. Here, we provide important extensions to the previous work, in particular: (1) an extensive background analysis, (2) a detailed description of the algorithms for process conformance tracking, uncertainty management and security compliance tracking, taking into account unexpected events as well as missing events, and (3) new results on attack patterns detection by PSA@R in co-action with external complex event processing and attack status evaluation components. In order to demonstrate how our model-based runtime analysis is applied, we have chosen processes concerned with the control activity of a hydroelectric power plant in a dam that was analyzed in a European research project [7]. We show how compliance monitoring can be used to observe system and process behavior, detect anomalies, and provide situational awareness not only on the infrastructure level but also up to business process level and thus to implement a holistic security strategy.

This article is structured as follows: Section 2 provides the background on model-based security compliance tracking. Section 3 introduces the abstract representation of running processes from CS, while Section 4 presents the process conformance tracking and uncertainty management algorithms. Section 5 discusses the security compliance tracking algorithm used to identify and predict critical states. Section 6 describes the adaptation of the approach to a specific industrial scenario as well as application experiences. Concluding remarks and the discussion of future research are given in Section 7.

2 Background and Related Work

What you can not measure, you can not manage. Thus, security management not only requires model-based methods during the development of secure systems but also security compliance tracking at runtime. 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 Security Information Event Management (SIEM) technology and process security analysis.

Security Information and Event Management. SIEM technology provides log management and compliance reporting as well as real-time monitoring and incident management for security events from networks, systems, and applications. To that end, one objective of the work presented in this article is, to bridge the gap between high-level security measurements and data gathered by SIEM engines, like OSSIM [8] and Prelude [9]. OSSIM detects events at the network layer and stores respective attributes like IP address or port number in a relational database. While it is possible to link these attributes to a security information model, neither OSSIM nor Prelude support reasoning over gathered data or model extension. A concise overview of current SIEM systems functionalities is presented in [10]. In [11], current threats are identified and advanced monitoring techniques such as file integrity monitoring, database activity monitoring, application monitoring, identity monitoring, and user activity monitoring are discussed. In [12], some challenges with respect to collecting and analyzing additional data sources and forensic analysis are outlined.

In the case study of this article we consider a critical infrastructure which is controlled by a Supervisory Control and Data Acquisition (SCADA) system. Compared to traditional IT systems, securing SCADA systems poses unique challenges. In order to understand these challenges and potential dangers, [13] provides a taxonomy of possible cyber attacks – including cyber-induced cyber-physical attacks on SCADA systems. Specific SCADA related security problems are discussed in [14].

In general, SIEM systems usually do not have knowledge of the application processes, thus lacking the connection between the reported security problems and the affected critical processes. The main feature of our approach is that we use cross-level process models to evaluate the security compliance of networked CS at runtime foresighted, thus enabling proactive response adapted to the situation.

Process Security Analysis. The business process perspective on security has to face cross-cutting issues and also fairness considerations. Responsibilities are not always centralized and infrastructures are often multi-purpose and partly not physically controlled by the process owner. Models of processes are often not available or outdated.

A formalized approach for security risk modeling in the context of electronic business processes is given in [15]. It also concerns simulation aspects, but does not incorporate the utilization of runtime models. A model-driven approach focusing on access control for business process models is provided in [16]. It allows for the annotation of security goals to business process models, the validation of annotated process models using model checking and the generation of configuration artifacts for runtime components. Runtime analysis of security properties is not addressed by this approach.

Two approaches that focus on security models at runtime are given in [17] and [18]. The first approach proposes 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 second approach discusses the integration of runtime and development time information on the basis of an ontology to engineer industrial automation systems.

Business Activity Monitoring (BAM) applications process events from networked CS in real-time in order to identify critical business key performance indicators, get a better insight into the business activities, and thereby improve the effectiveness of business operations [19]. In contrast to business process monitoring and BAM - which is usually applied directly by the Business Process Management (BPM) system itself - PSA@R uses (correlated) events from all directly and indirectly involved systems cross logical layers and thus can also evaluate cross-cutting concerns.

Formal methods, such as Linear Temporal Logic (LTL), state-charts, and related formalisms have been used for runtime monitoring of CS in [20, 21]. However, this work is mainly aiming at error detection, for example, concurrency related bugs. A classification for runtime monitoring of software faults is given in [22]. [23] analyzed a class of safety properties and related enforcement mechanisms that

work by monitoring execution steps of some target system, and terminating the target's execution, when an operation would violate the security policy. Extensions of this approach are discussed in [24], where it is proposed to integrate a local monitor component into a grid computational service architecture. Patterns and methods to allow for monitoring security properties are developed in [25, 26, 27].

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 or Event-driven Process Chain (EPC) [28, 6]. Also, some general-purpose simulation tools such as CPNTools [29] were proven to be suitable for simulating business processes. The process mining framework ProM [30] supports plug-ins for different types of models and process mining techniques [31]. Process mining techniques for analysis of large data sets and data streams aim to extract valuable process information building on techniques from data mining and machine learning, where knowledge discovery from data is mostly based on various statistical methods [32]. Process mining can be seen as a specialization of data mining for the discovery of implicit process knowledge in process flows. Many research efforts in this area are concerned with identification of unknown control-flow of processes. In addition, conformity of operating records to existing process models and possible improvements in processes are examined. Security topics in BPM seem orthogonal to the use cases and key concerns [33].

Conversely, the approach proposed here builds on on-the-fly dynamic simulation and analysis on the basis of operational Asynchronous Product Automata (APA) models introduced in [4]. This includes consideration of the current process state and the event information combined with the corresponding steps in the process model. In [1], the first publication on PSA@R, it was proposed to use APA to specify meta-events that match security critical situations in order to generate alerts. However, since this turned out to be slow and not easily usable by end-users, PSA@R now uses monitor automata [3] to specify the operational security requirements for compliance tracking at runtime.

Randell [34] introduced the term *judgemental system* for a system that makes a judgement that the activity or inactivity of an observed system constitutes failure. With respect to the exhaustive survey of approaches in the field of BPM given in [33], the compliance tracking approach of PSA@R supports the *check conformance using event data* approach, where information from the process model and the event data is used to identify deviations of runtime behavior of observed CS from expected behavior. A similar approach is described in [35] but the focus is on quantification of inconsistencies by the formation of metrics. The framework presented in [36] on runtime compliance verification for business processes is considered as complementary to the work on security compliance tracking presented here.

The term *security-by-design* is often used for model-based methods that aim to enhance the security of systems. However, in the currently used systems for run-time security monitoring such a model-based way of thinking is not yet established. Thus, PSA@R forms a novel supplement and a link between the above mentioned methods. We show that models can be utilized during runtime for the following purposes: (1) Anomaly detection with respect to behavior anomalies which indicate possible attacks, (2) compliance control with respect to explicit violations to security policies, and, (3) prediction of critical situations with respect to safety or security in the near future.

3 Abstract Representation of Running Processes

In our PSA@R approach, the operation of processes in CS is observed by analyzing events received from these systems. The core idea is, to utilize the knowledge about the specified control flow of the observed processes to detect anomalies and compliance violations. This complements existing methods which normally use only statistical informations derived by historical behavior analysis. Events from the observed systems are first filtered according to their relevance to the analysis and then mapped to an abstract representation that fits to the process model. Modeling - in this context - means to represent the

events and processes in CS in terms of mathematical objects that reflect the observed properties [37].

3.1 Event Model

A stream of events characterizes one specific execution trace of the observed system. This trace is a shuffle [38] of the traces of the executed process instances. The event model determines the internal mapping for the runtime events defined by an event schema. To reduce the complexity, only data required for the analysis or in generated alarms should be used in the model. Formally, it is assumed that an event represents a letter of the alphabet that denotes the possible actions in the system.

Definition 1 (process instance projection)**.** *Let P denote a finite set of process instances i of some process with $i \in P$ and let Σ_i denote pairwise disjoint copies of Σ. The elements of Σ_i are denoted by e_i and $\Sigma_P := \dot{\bigcup}_{i \in P} \Sigma_i$. The index i describes the bijection $e \leftrightarrow e_i$ for $e \in \Sigma$ and $e_i \in \Sigma_i$. Now the projection π identifies events from a specific process instance i. For $i \in P$, let $\pi_i^P : \Sigma_P^* \to \Sigma^*$ with*

$$\pi_i^P(e_r) = \begin{cases} e \mid & e_r \in \Sigma_i \\ \varepsilon \mid & e_r \in \Sigma_P \setminus \Sigma_i \end{cases} .$$

This is similar to the notion of a *correlation condition* [39] that defines which sets of events in the service log belong to the same instance of a process.

For effective use of PSA@R, it is necessary that a process instance projection is possible for each event. In many applications, a process instance identification is directly available as an attribute of the event. Sometimes, a set of attributes identifies the process instance. However, - particularly in *process-unaware systems* - the assumption about pairwise disjoint alphabets is not always valid. If the event data contain redundant or irrelevant attributes, a proper subset of attributes for use in model construction has to be selected. In order to avoid state space explosion problems, the *coarsest* abstraction that still contains all security relevant information should be used [3].

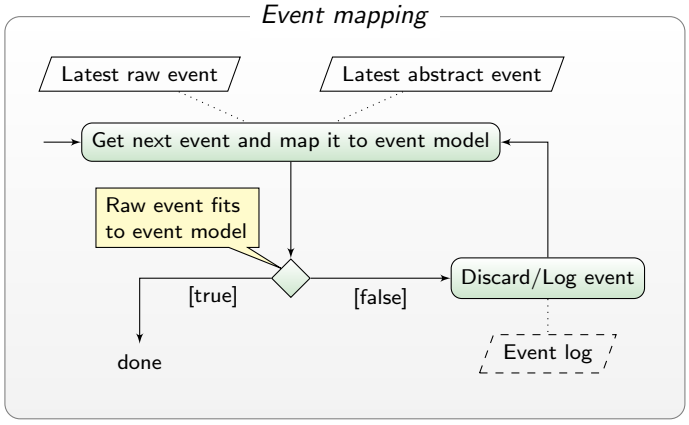


Figure 1: Event mapping algorithm (rectangles with rounded corners denote actions; rhombs denote conditions; trapezia denote data)

The algorithm used by PSA@R to filter the events that is shown in Figure 1 first reads the next event e from the observed system's event stream. If e does not fit to the event model, that is, $e \notin \Sigma_P$ (cf. Definition 1), then the event e is *unknown*, that is, it does not fit to the event schema used for the mapping. Depending on the audit requirements, the event will be discarded or logged. If the event fits to the model, it is transformed to the respective abstract event. In [40], it has been shown that more complex mapping schemes, e.g., duplication of abstract events, can be useful for specific evaluations.

3.2 Process Model

The behavior of a discrete system can be formally described by the set of its possible sequences of actions. This allows to define security requirements as well as to verify such properties, because for these purposes sequences of actions of the system have to be considered [41, 42]. PSA@R does not depend on a specific formal method chosen for model representation. However, the method must allow to compute the possible behavior (Reachability Graph (RG) or Labelled Transition System (LTS) [37]) from the model. For example, APA [4, 43], Petri nets [37], π-calculus [44], or B [45] meet this requirement.

In this work, we considered the system behavior at the abstraction level of the processes executed by the CS. In particular, we used a formal process model given by an APA representation, which is utilized to reflect the current state of the CS. This model provides the basis for the prediction of close-future actions. Formally, the behavior of an operational APA model is described by a RG.

Definition 2 (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*

$$(q_0,(e_1,i_1),q_1)\ (q_1,(e_2,i_2),q_2)\ \ldots\ (q_{n-1},(e_n,i_n),q_n)$$

with $i_k \in \Phi_{e_k}$, where Φ_{e_k} is the alphabet of the elementary automaton e_k, represents one possible sequence of actions of an APA.

State transitions $(p,(e,i),q)$ may be interpreted as labeled edges of a directed graph whose nodes are the states of an APA: $(p,(e,i),q)$ is the edge leading from p to q and labeled by (e,i). The subgraph reachable from the node q_0 is called Reachability Graph *of an APA.*

The operational models can be derived from existing process specifications, such as EPC specifications, Business Process Modeling Notation (BPMN) specifications, or Petri nets. In process aware systems such a specification can also be derived by process mining tools [30].

Example 1. *An EPC process specification provides the control flow structure of a process as a sequence of* events *and* functions *[6]. In an APA model that is derived from such a process specification, the set of possible output events of a process function can be used as the alphabet of the elementary automaton representing the function [2]. So the interpretation i is the output event. An example for a state transition is*

$$(p,(transfer,event = 'critical'),q).$$

The parameters of this state transition are the state p, the tuple composed of the elementary automaton transfer and its interpretation event $= 'critical'$, and the follow-up state q.

PSA@R uses the RG to *predict the close-future behavior of the process instance*. A subgraph of the RG starting with the current state of the process instance can always be computed on-the-fly based on the formal process model. The *prediction depth* is the depth of this subgraph starting from the current state. In [6] Mendling presented an extensive metrics analysis on 2003 EPC process specifications which revealed that the number of nodes a connector is in average connected to resulted in 3.56 for the mean value μ and 2.40 for the standard deviation σ. This limits the effects of possible state space explosion in process behavior prediction for such process-aware information systems.

Remark 1. *If it is possible to compute the complete RG of a process model, then it is sufficient to compute the RG only once, because all process instances of a specific process type have the same operational model and so the RG of all process instances is the same up to the process instance identifier.*

Remark 2. *"Underfitting" the process description [31] would allow for behavior that can not actually happen and thus would not only limit the prediction depth because of state space explosion problems but would also lead to too many predictive alerts (false positives).*

4 Process Conformance Tracking and Uncertainty Management

Process conformance tracking is the capability to detect deviations of observed events from expected events with respect to the current state of a process. Since in PSA@R the process knowledge is provided by the process models, the specified model control flow can be compared to the actual observed behavior. At runtime, the latest state of the process behavior model of the process instance i is synchronized with the running process using the projection of the measured events to the respective state transitions $(p,(e,i),q)$ of the RG. Using this approach, operational models can reflect the state of process instances in the observed CS and provide an abstract view of their runtime behavior. An *anomaly* is a deviation of observed events from expected events with respect to the current state. Such anomalies in the behavior can be caused by attacks on the observed CS but there are also other possible causes, such as changes in process behavior, an incomplete process model or errors in the transmission of events.

Figure 2 depicts the PSA@R algorithm for conformance tracking by detection of anomalies such as unknown, unexpected and missing events. If the process behaves not conformant to its model, then a semi-automatic adaptation of the *de-jure* process model based on *de-facto* measured behavior at runtime is applied. The steps in this algorithm are now described more formally.

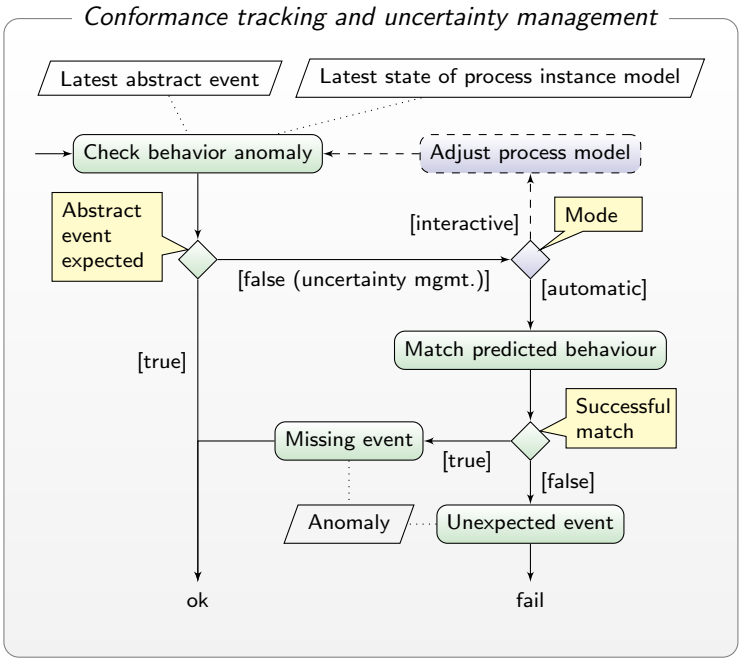


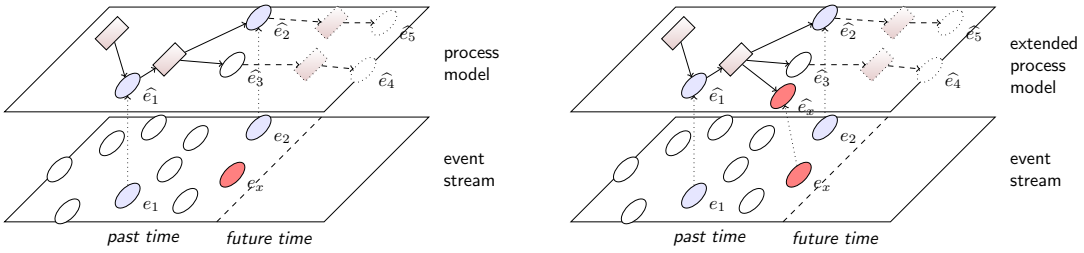
Figure 2: Conformance tracking and uncertainty management algorithm (rectangles with rounded corners denote actions; rhombs denote conditions; trapezia denote data)

Check behavior anomaly. Let $RG_P(q,d)$ denote the possible behavior B of the process P given by a RG starting with current state q and prediction depth d, that is, the set of all possible coherent sequences of state transitions of length less or equal d starting at q. It is further assumed that $d > 0$. The event e is *expected* for process instance i, iff there is a transition $(q,(,event = h(\pi_i^P(e))),q')$ in $RG_P(q,1)$.

Adjust process model. The user is expected to change the process specification, so that the event e fits to the possible behavior in the current state q. For example, the user can decide to insert the abstract

representation of the unexpected event e as well as necessary connections to the process specification. This constitutes a *belief change* with respect to the *de-jure* process model.

Example 2. *Figure 3(a) shows a situation where an event e_x is received. It has already passed the check, whether it fits to the event model, but it is not part of the process behavior in scope of the analysis. Figure 3(b) shows a possible adjustment of the process model from Figure 3(a). In this case, the user has inserted the new abstract event $\widehat{e_x} = h(\pi_i^P(e_x))$ to the process model along with a connecting edge from the current function.*



(a) Event not expected in *de-jure* process model (b) Adapt *de-jure* process model to *de-facto* behavior

Figure 3: Treatment of unexpected events (ellipses in event stream denote observed events, whereby filled ellipses e_1, e_2 denote events that belong to the specific process instance i, i.e., $e_1, e_2 \in \Sigma_i$; ellipses in process model denote abstract events w.r.t. an abstraction h used in the event model, e.g., $\widehat{e_1} = h(\pi_i^P(e_1))$; dotted arrows denote this mapping; rectangles denote process functions; solid arrows denote transitions; dashed arrows denote possible transitions)

Match predicted behavior. Find a transition $(p, (, event = h(\pi_i^P(e))), p')$ in B. If this is successful, then it is assumed that one or more events have been missed and the current process state is adjusted, that is, $q := p$.

Example 3. *Figure 4 depicts a situation where an event e_4 is received but not expected in this state of the process. However, an abstract event $\widehat{e_4} = h(\pi_i^P(e_4))$ is part of a possible continuation of the process. Thus, it is assumed that some event e with $\widehat{e_3} = h(\pi_i^P(e))$ has been missed.*

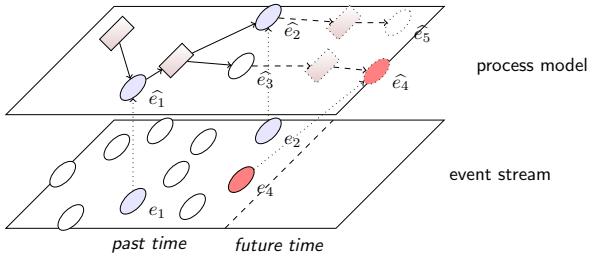


Figure 4: Map event to future state (filled ellipses e_1, e_2, e_4 denote observed events of process instance i; ellipses in process model denote abstract events w.r.t. an abstraction h, e.g., $\widehat{e_4} = h(\pi_i^P(e_4))$; dotted arrows denote this mapping; rectangles denote process functions; solid arrows denote transitions; dashed arrows denote possible transitions; an event that could be mapped to the abstract event $\widehat{e_3}$ is missing)

Missing event. An anomaly which has been classified as a missing event has been detected and resolved. However, a *missing event warning* is raised.

Unexpected event. An anomaly is detected which could not be resolved. An *unexpected event alert* is raised.

Remark 3. *In order to cope with the case of major deviations of observed behavior from the behavior model it can be useful to add another step to the algorithm, namely, to search for the observed event e in the process model and not only in the close-future behavior model. This could reduce the number of alerts. This step is not added here because it is assumed that it is necessary to adjust the process model in such cases.*

Results of an application of this algorithm for fraud detection in mobile payment processes have been reported in [46]. For further work, it is considered to integrate methods using metrics to quantify deviations from process specifications, such as the one described in [47], which is - like the work presented in this chapter - based on the concept to represent system behavior by the set of traces generated by a process.

5 Security Compliance Tracking and Prediction of Critical States

Security compliance tracking is the capability to apply a security model at runtime in order to identify violations of security requirements. In order to provide this novel capability, [1] and [2] introduced an approach for the validation of the actual security status of business process instances. [3] substantially refined this concept by detailing the monitoring formalism, implementation, evaluation, and context of the approach. Security requirements to be satisfied during process execution are formalized by a *security model*, which must be derived systematically [48, 49, 50]. As a means for compliance tracking, we use *monitor automata* [3].

Definition 3. *A monitor automaton $\mathbb{M}$ consists of a set $\mathscr{M}$ of labeled states, an alphabet Λ of predicates on RG state transitions, a transition relation $\mathscr{T}_{\mathscr{M}} \subseteq \mathscr{M} \times \Lambda \times \mathscr{M}$, a set of initial states $\emptyset \neq \mathscr{M}_0 \subset \mathscr{M}$, and a set of accepting states $\mathscr{M}_f \subset \mathscr{M}$.*

Predicates of $\mathbb{M}$ are applied to state transitions $(p_i,(e_j,i_k),q_l)$ of the RG.

Example 4. *The predicate $(,(,event = 'login\_root'),)$ is true if 'login_root' is bound to the interpretation variable event of the interpretation i_k. No condition for the predecessor state p_i, successor states q_l and the elementary automaton e_j is given.*

Monitor automata specify security requirements with respect to the behavior of a process. Accepting states of monitor automata refer to *security critical states*, which trigger the generation of respective alerts. Whenever the latest abstract event from the observed system matches an active state transition of a monitor automaton this transition is executed. Each state of the RG of a process instance has an associated monitor automaton state set that is computed during simulation. Security critical states are reached whenever an accepting monitor automaton state becomes a member of such state set.

An external security status can be involved in the definition of a monitor automaton for exploiting self- and environment awareness in the security reasoning. This supports the collaboration of several security strategy processing components [51].

Prediction of critical states is the capability to predict violations of security requirements in the near future. More precisely, if a state transition leads to a critical state in a monitor automaton within the prediction scope of the process behavior then a so-called *predictive alert* will be raised. Figure 5 illustrates the respective part of the PSA@R approach that enables the runtime assessment of security critical behavior. The steps of this algorithm are now described in detail.

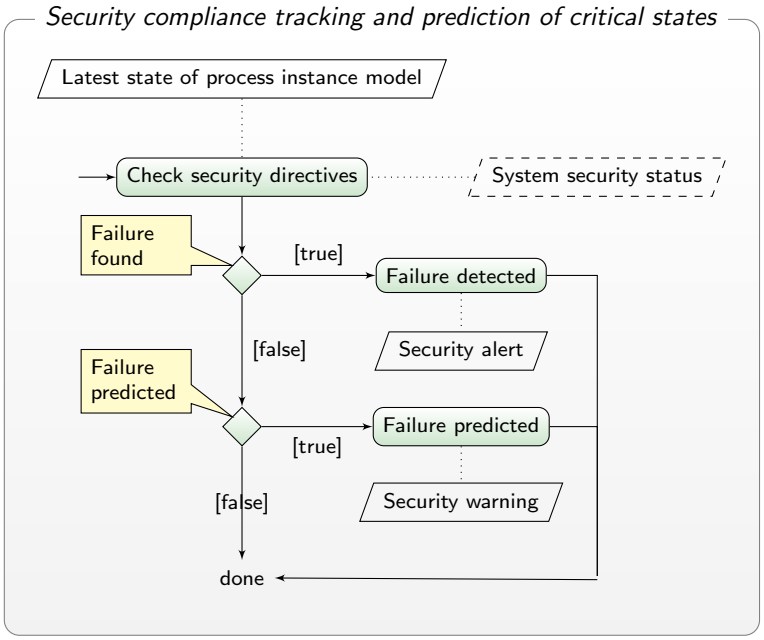


Figure 5: Security compliance tracking and prediction of critical states (rectangles with rounded corners denote actions; rhombs denote conditions; trapezia denote data)

Check security directives. For runtime identification of security critical states and the prediction of possible failures in the near future, the security compliance tracking algorithm uses on-the-fly checks of predicates that express the required security properties in terms of state transitions in the current or predicted behavior. Basically, predicates annotated at the edges of a monitor automaton are applied to state transitions of the RG. Optionally, this check takes into account security status information provided by external components.

Failure detected. If the predicate that expresses compliance requirements applicable in the current state matches the current state transition and the state reached in the monitor automaton is marked as *critical state*, then a *security alert* is raised because a security critical situation has been detected.

Failure predicted. If an applicable predicate matches the current state transition and some state of the monitor automaton that is reachable by a path in the predicted behavior (within the prediction depth) is marked as *critical state*, then a *security warning (predictive alert)* is raised. This signals that the current situation might escalate to a security critical situation in the close future.

5.1 Alert Model

The alert generation used by PSA@R applies a mapping from state transitions of the security analysis model onto security events, such as warnings, alerts or predictive alerts. The mapping can be configured and currently supports Intrusion Detection Message Exchange Format (IDMEF) [52], OSSIM [8], and the format used by the MASSIF project [53].

The security alerts are enriched by the information needed for further processing and fed into the runtime environment for delivery to decision support and reaction systems.

6 Case study: Security Compliance Tracking in Critical Infrastructure

The PSA@R approach has been implemented in a prototype called Predictive Security Analyzer (PSA). In this section, we summarize our experience and lessons learned from an application of the PSA in a critical infrastructure process control (on the example of a storage dam in a hydroelectric power plant) [7]. The PSA was deployed as a model management component of the next-generation SIEM architecture MASSIF [53], to perform security event processing and anomaly detection on the business application (service) layer. As shown in Figure 6, collection, normalization, and preprocessing of raw events from the observed and managed CS were carried out by other MASSIF components, namely, Generic Event Translation (GET) [7] and Complex Event Processing (CEP) [54]. The Attack Modelling and Security Evaluation Component (AMSEC) [55] provided additional security status information. Alerts produced by the PSA were forwarded to the Decision Support and Reaction (DSR) [56] system which implemented countermeasure selection and response mechanisms.

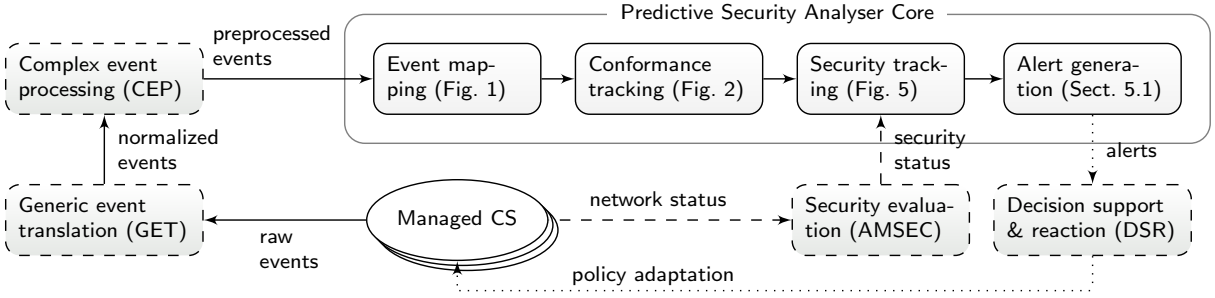


Figure 6: PSA process monitoring and security compliance tracking in next-generation SIEM (dashed boxes denote optional external components, solid arrows denote event flow, dashed arrows denote context information, dotted arrows denote alert processing)

The PSA consists of two main parts: the PSA Modeler that supports the specification of event models, process models, and security models and the PSA Core that actually performs process security analysis at runtime. The box "Predictive Security Analyser Core" in Figure 6 shows the integration of the algorithms for event mapping, process conformance tracking, uncertainty management, security compliance tracking, prediction of critical states, and alert generation, which have been described in the previous sections, in the PSA Core.

6.1 Critical Infrastructure Process Control

The critical infrastructure process control scenario is concerned with the control activity of a hydroelectric power plant representing a complex CS that consists of a number of components involved into power production. These components include the dam storing the water to feed the hydro-power station, the penstocks and gates enabling the water delivery to the hydroelectric turbine, specialized field sensors controlling the state of the dam and devices, water flows, and other structural and geo-technical instrumentation data [57]. The storage dam of the power plant is remotely controlled by a SCADA system that pushes automated or operator-driven supervisory commands based on information received from remote stations. The SCADA system can be accessed using the control station located in the control room of the power plant. Physical (RFID-based) and logical access controls should normally be deployed to secure the control station.

We explored a combined misuse case encompassing administrator password theft, hazardous water release operations and control station hacking [58] related to an insider attack on the control system of

the storage dam. We considered an adversary as a disgruntled employee of the dam with a non-privilege role (i.e., cleaning staff). The adversary has an RFID-badge allowing her to enter the control room. The RFID badge provides the access control system with the information related to the identity and the role of its owner. The adversary was able to gain in an illegal way administrative credentials needed to login with administrator privileges at the dam control station that operates dam gates. The attack is characterized by the following sequence of steps: (1) The adversary enters the control room using her RFID badge; (2) The adversary logs into the control system using the stolen administrator credentials; (3) The adversary issues a command to open the gate with the intent to discharge the dam's reservoir. Due to the unintended release of water, the attack can endanger people using the dam's reservoir for recreational activities. It also can result in the economic damage, due to the waste of water stored for power generation. The specifics of this scenario is that the correlation of events generated by both physical and logical security systems is required to detect an attack pattern [59, 3].

The emulated testbed that represented a dam being controlled by means of a SCADA system was provided for the simulation of this attack [58]. The event stream - reflecting processes that run in the dam - was created using the DaMon SCADA application installed in the testbed [58]. An attack injector was used to shuffle event traces representing attacks into the normal synthetic traffic generated by the workload generator. The GET components collected different events from the dam testbed, such as events from physical sensors (e.g., *gate_open*, *gate_closed*, *SeepageFlowCheck_ALARM*), RFID access control system (e.g., *administrator_authorized_access*, *employee_unauthorized_access*), and operating system logs (e.g., *login_root*), and parsed them for further processing by CEP and the PSA.

The PSA monitored the processes taking place in the control room to identify deviations between the actual workflow and the expected one. Any deviation were seen as a possible path toward an attack [60]. In order to detect any undesirable deviations, we developed the process model incorporating various security-relevant events collected by the GET framework. The behavior of the process was represented by a RG (cf. Definition 2) consisting of 41 states and 294 state transition. The initial part of the RG is depicted in Figure 7.

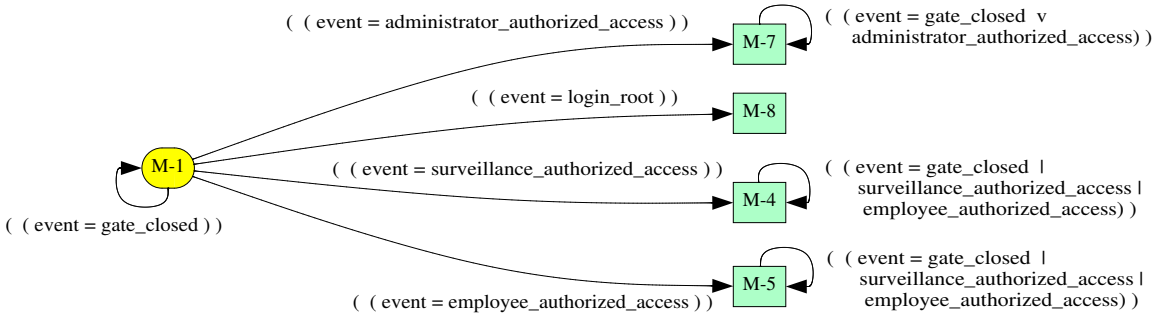


Figure 7: Initial part of the RG representing process behavior in the critical infrastructure (nodes denote states of the RG (please note that the PSA prints the state q_0 as `M-1`); edges denote transitions of the RG)

In order to define and monitor security conditions related to the misuse case using the PSA, we employed monitor automata (cf. Definition 3). The box "Security compliance monitor" in Figure 8 depicts an example of a monitor automaton with four critical states, which indicate security compliance violations in the critical infrastructure process control. In this scenario, the security compliance check was supported by security status assessments provided by the AMSEC, which monitored the network layer underlying the critical infrastructure management. It observed events representing changes in the infrastructure and, based on the analysis of such events, provided a measure of the risk related to the network configuration.

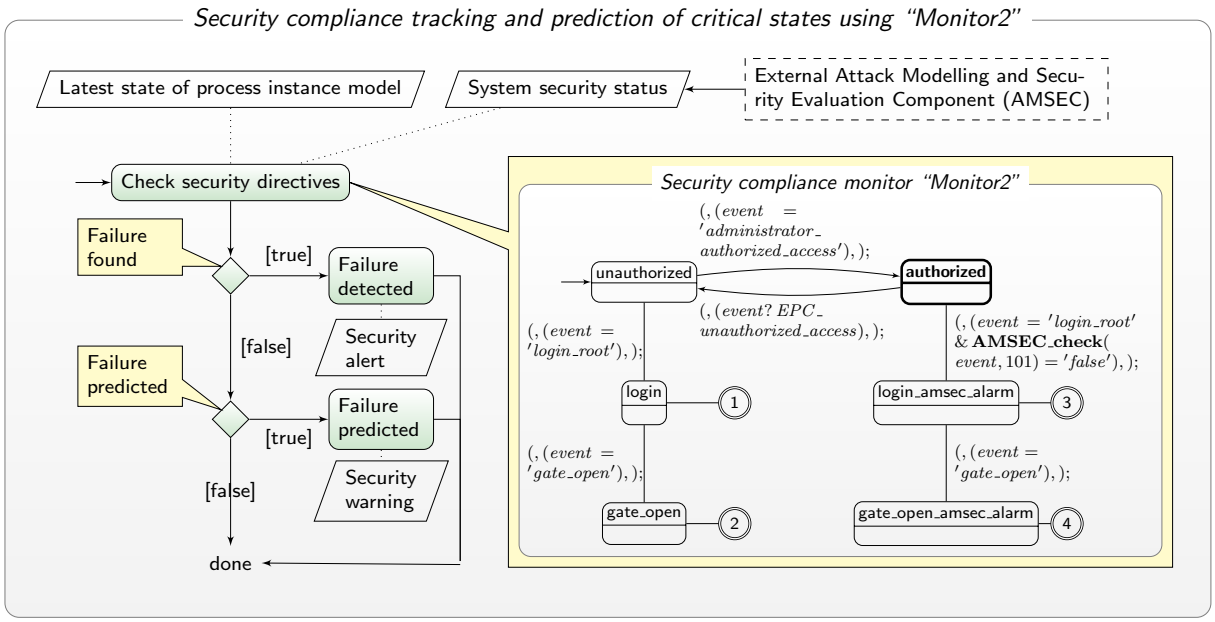


Figure 8: Security compliance tracking and prediction of critical states in PSA (rectangles with rounded corners denote actions; rhombs denote conditions; trapezia denote data; rectangles with rounded corners and split line denote states in compliance monitor (current state is bold); double circles mark critical states; edge inscriptions in compliance monitor denote predicates)

Example 5. *We now consider a situation where the security compliance monitor "Monitor 2" has reached the state* `authorized` *(cf. bold state in Figure 8). The predicate at the edge from* `authorized` *to* `login_amsec_alarm` *represents the situation that a login as root happens during an insecure network status. Formally, the predicate*

$$(,(,event = 'login\_root') \& AMSEC\_check(event, 101) =' false'),) \text{ is true}$$

if $'login\_root'$ *is bound to the interpretation variable event and the function* $AMSEC\_check(event, 101)$ *that provides specific information about the attack status of the observed system is* $false$*. In this case the attack status has been evaluated by the AMSEC. The AMSEC used 101 as an identifier for the metric of the network security status and the function* $AMSEC\_check(event, 101) =' false'$ *indicates an insecure network status. In this case, the compliance monitor reaches the state login_amsec_alarm, which is marked as critical state number 3.*

By analyzing the received events, the PSA was able to predict possible failure scenarios and trigger respective alarms. Response actions have been provided by the DSR system [61]. The reaction strategies were defined as prompting administrator dual factor authentication (RFID re-authentication), triggering an acoustic alarm, or forcing an automatic command on the dam, such as closing a gate. Besides the standard fields, the IDMEF alarm event provided the monitor state, in this case $'gate\_open'$, and an information if the critical state was predicted or already reached (see Listing 1). Optionally, backward references to the event that caused the alarm and the associated high level security goal could be included.

It is worth noticing that the PSA has a capability to determine how every observed alert refers to (high-level) security goals set for a monitored system. In the PSA the backward reference of a security monitor (monitor automaton) to the originating security goal is provided by the corresponding project description. Figure 9 shows the PSA project description of the critical infrastructure scenario. In particular, the security monitor "Monitor2" (cf. Figure 8) is linked to the security goal "Supervision". In the

```
<IDMEF-Message>
  <Analyzer analyzerid="0" name="PSA"
    manufacturer="http://www.sit.fraunhofer.de"
    model="PSA" version="3.0.916" class="Concentrator" ostype="Linux"
    osversion="3.1.10-1.19-desktop">
     <Node category="unknown"><name>tux</name></Node>
     <Process><name>psa</name><pid>19302</pid><path>/local/acl90/clim</path>
     </Process>
  </Analyzer>
  <CreateTime ntpstamp="0xd563683d.0x00000000">2013-06-12T23:35:57+02:00
  </CreateTime>
  <Classification text="Monitor Automaton"/>
  <AdditionalData type="xsd:string" meaning="MonitorState">gate_open
  </AdditionalData>
  <AdditionalData type="xsd:string" meaning="Predicted">true</AdditionalData>
</IDMEF-Message>
```

Listing 1: IDMEF alarm event

current implementation of the PSA the security goal is an unstructured file containing a human readable explanation of the goal, in this case: ”All critical actions have do be supervised”. This should be refined to a structured representation of the respective security directive as proposed in [51].

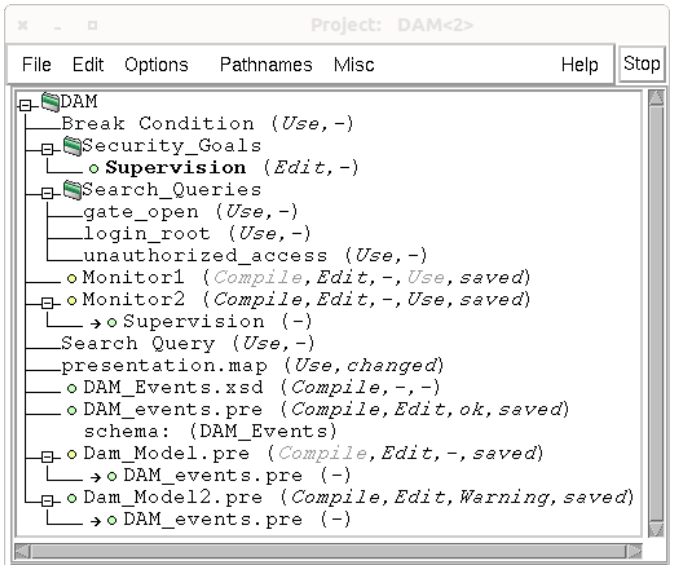


Figure 9: Project tree of critical infrastructure scenario in PSA

6.2 Lessons Learned

Based on the experiences with application of an early prototype of the PSA in a logistics scenario [2], we have applied the PSA tool in several use cases, each of which allowed us to examine a particular aspect of PSA@R in a realistic industrial setup [61]. Besides the critical infrastructure scenario described above, the PSA has been evaluated in other industrial domains, e.g., in managed enterprise service infrastructures [59], mobile money transfer services [46], and Olympic Games IT infrastructure management [62]. The realized test setups targeted different, domain-specific security issues and varied in complexity in terms of process security tasks, but as a whole covered all steps of the security analysis cycle enabled by

the PSA. The main problem we faced during the adaptation of the PSA to the provided use case scenarios is that none of them a priori involved either process-aware information systems or process specifications. Thus, a point of particular interest regarding these industrial setups is exploitation of process-aware security compliance tracking, as provided by the PSA, in "process-unaware" CS that can often be seen in the wild. In the critical infrastructure scenario we solved the problem of the missing process specification together with the application provider by assuming that there exists only one business process and only one process instance and that every abstract event belongs to that process instance. The application provider had access to the system that generated the events and so it was possible to adapt the generated events to the needs of abstract representation of the running processes. The process behavior (cf. Figure 7) was then learned by repeated application of the uncertainty management algorithm described in Section 4 in interactive mode on an event log.

In a simulation of the critical infrastructure misuse case described above, the PSA generated a warning before the water level decreased significantly. The warning could then be used to reconfigure the system to deny the access of the suspicious user to the control machine [61]. In a simulation of another misuse case, where the adversary compromises water level sensors of the dam, the PSA managed to generate a warning before an alarm condition, due to incoherence in measurement data, was reached on the dam control system. In this misuse case, the PSA detection can be used to trigger external reaction systems to send reprogramming commands to the managed system before the attack is completed [61].

7 Conclusion and Research Directions

The approach for runtime security analysis presented in this article enables security compliance tracking of processes in networked CS. This capability is based on models of events, process behavior, and security compliance requirements. Our extensive background analysis has shown that in the currently used systems for runtime security monitoring a model-based way of thinking is not yet established. Our approach PSA@R forms a novel supplement and a link between the established methods in data mining, machine learning, predictive analytics, complex event processing and process mining. PSA@R provides security analysis at runtime exploiting process behavior models, which enables on-the-fly security compliance checks and prediction of close-future violations of security requirements. Building on previous PSA@R publications [1, 2, 3], this article contributes, in particular, the conformance tracking and uncertainty management algorithm describing the behavior of PSA@R in case of unexpected or missing events. Furthermore, the integration of runtime knowledge about the external security status in the security compliance tracking is demonstrated by an industrial application of critical infrastructure process control. This knowledge enriches the security model and enables exploitation of additional security status assessments from other systems in the compliance monitor automata used for security reasoning. The proposed integration of process behavior simulation and runtime monitoring allows for early security warnings and predictive alarms on possible security critical states in close future.

We have shown that utilizing a model of the prescribed process behavior and the respective compliance rules supports an intelligent security management life-cycle. The process owners can: (1) assess the achievement of the process objectives, (2) determine and predict deviations from the planned (prescribed) behavior, (3) monitor and audit the ongoing process regarding the security policies, (4) assess the treatment of incidents, (5) identify weak points in the process flow and so plan corrections of the process flow. At that, we do not intend to diminish the significance of security-by-design. Our work is aimed as a critical add-on in order to address the dynamics of electronic business processes. In combination with other novel applications, PSA@R enables anticipatory impact analysis, decision support and impact mitigation by adaptive configuration of countermeasures to implement holistic security strategy management. In [51, 63], we proposed a security strategy meta model that consolidates the necessary

information. It incorporates business goals and the identified important assets, compliance requirements that create obligations for security management, runtime events, system security status, as well as risk and countermeasure assessments. A prototype of PSA@R has been implemented and evaluated in important industrial scenarios [3, 61]. This has proven that model-based analysis is applicable and fast enough for security compliance tracking of processes at runtime. Further results, where we compared PSA@R with state-of-the-art fraud detection approaches, indicate that we can achieve better recognition performance [40]. For further research, we plan to integrate methods, such as the one described in [47] using metrics to quantify deviations from process specifications.

Acknowledgments

The presented work was developed in context of the project ACCEPT (ID 01BY1206D) being funded by the German Federal Ministry of Education and Research and the project MASSIF (ID 257475) being co-funded by the European Commission within the Seventh Framework Programme.

References

[1] R. Rieke and Z. Stoynova, "Predictive security analysis for event-driven processes," in *Proc. of the 5th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS'10), St. Petersburg, Russia, LNCS*, vol. 6258. Springer, September 2010, pp. 321–328.

[2] J. Eichler and R. Rieke, "Model-based Situational Security Analysis," in *Proc. of the 6th International Workshop on Models@run.time (MRT'11), Wellington, New Zealand, CEUR*, vol. 794. RWTH Aachen, October 2011, pp. 25–36.

[3] R. Rieke, J. Repp, M. Zhdanova, and J. Eichler, "Monitoring security compliance of critical processes," in *Proc. of the 22nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'14), Turin, Italy*. IEEE, February 2014, pp. 525–560.

[4] P. Ochsenschläger, J. Repp, R. Rieke, and U. Nitsche, "The SH-Verification Tool – Abstraction-Based Verification of Co-operating Systems," *Formal Aspects of Computing, The International Journal of Formal Method*, vol. 10, no. 4, pp. 381–404, April 1998.

[5] C. E. Landwehr, "Formal models for computer security," *ACM Computing Surveys*, vol. 13, no. 3, pp. 247–278, September 1981.

[6] J. Mendling, *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*, ser. LNBIP. Springer, 2008, vol. 6.

[7] L. Coppolino, S. D'Antonio, V. Formicola, and L. Romano, "Enhancing SIEM technology to protect critical infrastructures," in *Proc. of the 7th International Workshop on Critical Information Infrastructures Security (CRITIS'12), Lillehammer, Norway, LNCS*, vol. 7722. Springer, September 2012, pp. 10–21.

[8] AlienVault, "AlienVault Unified SIEM," http://alienvault.com/, 2012, [Online; accessed 10-Jun-2015].

[9] CS Group, "Prelude Security Information and Event Management," http://www.prelude-siem.com, 2014, [Online; accessed 15-Jun-2015].

[10] M. Nicolett and K. M. Kavanagh, "Magic Quadrant for Security Information and Event Management," Gartner Reasearch, May 2010.

[11] Securosis, "Monitoring up the Stack: Adding Value to SIEM," Securosis L.L.C., Phoenix, AZ, White Paper, November 2010. [Online]. Available: https://securosis.com/research/publication/monitoring-up-the-stack-adding-value-to-siem

[12] ——, "Applied Network Security Analysis: Moving from Data to Information," Securosis L.L.C., Phoenix, AZ, White Paper, December 2011. [Online]. Available: https://securosis.com/research/publication/applied-network-security-analysis-moving-from-data-to-information

[13] B. Zhu, A. Joseph, and S. Sastry, "A taxonomy of cyber attacks on scada systems," in *Proc. of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing, Dalian, China*. IEEE, October 2011, pp. 380–388.

[14] G. Dan, H. Sandberg, M. Ekstedt, and G. Björkman, "Challenges in power system information security," *IEEE Security & Privacy*, vol. 10, no. 4, pp. 62–70, July-August 2012.

[15] S. Tjoa, S. Jakoubi, G. Goluch, G. Kitzler, S. Goluch, and G. Quirchmayr, "A formal approach enabling risk-aware business process modeling and simulation," *IEEE Transactions on Services Computing*, vol. 4, no. 2, pp. 153–166, April-June 2011.

[16] C. Wolter, M. Menzel, A. Schaad, P. Miseldine, and C. Meinel, "Model-driven business process security requirement specification," *Journal of Systems Architecture*, vol. 55, no. 4, pp. 211–223, April 2009.

[17] B. Morin, T. Mouelhi, F. Fleurey, Y. Le Traon, O. Barais, and J.-M. Jézéquel, "Security-driven model-based dynamic adaptation," in *Proc. of the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE'10), Antwerp, Belgium*. ACM, September 2010, pp. 205–214.

[18] M. Melik-Merkumians, T. Moser, A. Schatten, A. Zoitl, and S. Biffl, "Knowledge-based runtime failure detection for industrial automation systems," in *Proc. of the 5th International Workshop on models@run.time (MRT'10), Oslo, Norway, CEUR*, vol. 641. RWTH Aachen, October 2010, pp. 108–119.

[19] D. W. McCoy, "Business Activity Monitoring: Calm Before the Storm," Gartner Research, 2002, [Online; accessed 15-Jan-2014]. [Online]. Available: http://www.gartner.com/resources/105500/105562/105562.pdf

[20] R. Kazhamiakin, M. Pistore, and L. Santuari, "Analysis of communication models in web service compositions," in *Proc. of the 15th International Conference on World Wide Web (WWW'06), Edinburgh, Scotland*. ACM, May 2006, pp. 267–276.

[21] T. Massart and C. Meuter, "Efficient online monitoring of LTL properties for asynchronous distributed systems," Université Libre de Bruxelles, Tech. Rep., 2006.

[22] N. Delgado, A. Gates, and S. Roach, "A taxonomy and catalog of runtime software-fault monitoring tools," *IEEE Transactions on Software Engineering*, vol. 30, no. 12, pp. 859–872, December 2004.

[23] F. B. Schneider, "Enforceable security policies," *ACM Transactions on Information and System Security*, vol. 3, no. 1, pp. 30–50, February 2000.

[24] F. Martinelli, P. Mori, and A. Vaccarelli, "Towards continuous usage control on grid computational services," in *Proc. of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS/ICNS'05), Papeete, Tahiti*. IEEE, October 2005, pp. 82–82.

[25] C. Serban and B. McMillin, "Run-time security evaluation (RTSE) for distributed applications," in *Proc. of the 1996 IEEE Symposium on Security and Privacy, Oakland, California, USA*. IEEE, May 1996, pp. 222–232.

[26] T. Tsigritis and G. Spanoudakis, "Diagnosing runtime violations of security & dependability properties," in *Proc. of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE'08), San Francisco, USA*. Knowledge Systems Institute Graduate School, July 2008, pp. 661–666.

[27] A. Evesti, E. Ovaska, and R. Savola, "From security modelling to run-time security monitoring," in *Proc. of the European Workshop on Security in Model Driven Architecture (SECMDA'09), Enschede, The Netherlands*. CTIT, June 2009, pp. 33–41.

[28] R. M. Dijkman, M. Dumas, and C. Ouyang, "Semantics and analysis of business process models in BPMN," *Information and Software Technology*, vol. 50, no. 12, pp. 1281–1294, November 2008.

[29] A. Rozinat, M. T. Wynn, W. M. P. van der Aalst, A. H. M. ter Hofstede, and C. J. Fidge, "Workflow simulation for operational decision support," *Data & Knowledge Engineering*, vol. 68, no. 9, pp. 834–850, September 2009.

[30] H. Verbeek, J. Buijs, B. Dongen, and W. Aalst, "Xes, xesame, and prom 6," in *Proc. of the CAiSE Forum on Information Systems Evolution, Hammamet, Tunisia, LNBIP*, vol. 72. Springer, June 2010, pp. 60–75.

[31] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[32] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Magazine*, vol. 17, no. 3, pp. 37–54, Fall 1996.

[33] W. M. P. van der Aalst, "Business process management: A comprehensive survey," *ISRN Software Engineering*, vol. 2013, p. 37, 2013.

[34] B. Randell, "On failures and faults," in *Proc. of the International Symposium of Formal Methods Europe (FME'03), Pisa, Italy, LNCS*, vol. 2805. Springer, September 2003, pp. 18–39.

[35] A. Rozinat and W. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Information Systems*, vol. 33, no. 1, pp. 64–95, March 2008.

[36] F. M. Maggi, M. Montali, M. Westergaard, and W. M. P. van der Aalst, "Monitoring business constraints with linear temporal logic: An approach based on colored automata," in *Proc. of the 9th International Conference on Business Process Management (BPM'11), Clermont-Ferrand, France, LNCS*, vol. 6896. Springer, August–September 2011, pp. 132–147.

[37] D. A. Peled, *Software Reliability Methods*, 1st ed. Springer, 2001.

[38] H. Björklund and M. Bojanczyk, "Shuffle Expressions and Words with Nested Data," in *Proc. of the 32nd International Symposium on Mathematical Foundations of Computer Science (MFCS'07), Cesky Krumlov, Czech Republic, LNCS*, vol. 4708. Springer, August 2007, pp. 750–761.

[39] H. R. Motahari-Nezhad, R. Saint-Paul, F. Casati, and B. Benatallah, "Event correlation for process discovery from web service interaction logs," *The VLDB Journal*, vol. 20, no. 3, pp. 417–444, June 2011.

[40] M. Zhdanova, J. Repp, R. Rieke, C. Gaber, and B. Hemery, "No smurfs: Revealing fraud chains in mobile money transfers," in *Proc. of the 9th International Conference on Availability, Reliability and Security (ARES'14), Fribourg, Switzerland*. IEEE, September 2014, pp. 11–20.

[41] S. Schneider, "Security Properties and CSP," in *Proc. of the 1996 IEEE Symposium on Security and Privacy, Oakland, California, USA*. IEEE, May 1996, pp. 174–187.

[42] S. Gürgens, P. Ochsenschläger, and C. Rudolph, "Abstractions preserving parameter confidentiality," in *Proc. of the 10th European Symposium on Research in Computer Security (ESORICS'05), Milan, Italy, LNCS*, vol. 3679. Springer, September 2005, pp. 418–437.

[43] P. Ochsenschläger and R. Rieke, "Abstraction based verification of a parameterised policy controlled system," in *Proc. of the 4th International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security (MMM-ACNS'07), St. Petersburg, Russia, Communications in Computer and Information Science*, vol. 1. Springer, September 2007, pp. 228–241.

[44] P. Sobocinski, "A well-behaved lts for the pi-calculus: (abstract)," *Electronic Notes in Theoretical Computer Science*, vol. 192, no. 1, pp. 5–11, October 2007.

[45] D. Bert and F. Cave, "Construction of finite labelled transition systems from b abstract systems," in *Proc. of the 2nd International Conference on Integrated Formal Methods (IFM'00), Dagstuhl Castle, Germany, LNCS*, vol. 1945. Springer, November 2000, pp. 235–254.

[46] R. Rieke, M. Zhdanova, J. Repp, R. Giot, and C. Gaber, "Fraud detection in mobile payment utilizing process behavior analysis," in *Proc. of the 8th International Conference on Availability, Reliability and Security (ARES'13), Regensburg, Germany*. IEEE, September 2013, pp. 662–669.

[47] S. Banescu and N. Zannone, "Measuring privacy compliance with process specifications," in *Proc. of the 3rd International Workshop on Security Measurements and Metrics (MetriSec'11), Banff, Alberta, Canada*. IEEE, September 2011, pp. 41–50.

[48] B. Fabian, S. Gürses, M. Heisel, T. Santen, and H. Schmidt, "A comparison of security requirements engineering methods," *Requirements engineering*, vol. 15, no. 1, pp. 7–40, March 2010.

[49] D. Mellado, C. Blanco, L. E. Sánchez, and E. Fernández-Medina, "A systematic review of security requirements engineering," *Computer Standards & Interfaces*, vol. 32, no. 4, pp. 153–165, June 2010.

[50] A. Fuchs and R. Rieke, "Identification of Security Requirements in Systems of Systems by Functional Security Analysis," in *Architecting Dependable Systems VII*, ser. LNCS, A. Casimiro, R. de Lemos, and C. Gacek, Eds. Springer, 2010, vol. 6420, pp. 74–96.

[51] R. Rieke, J. Schütte, and A. Hutchison, "Architecting a security strategy measurement and management system," in *Proc. of the 1st Workshop on Model-Driven Security (MDsec'12) at MoDELS 2012, Innsbruck, Austria*. ACM, October 2012, pp. 2:1–2:6.

[52] H. Debar, D. Curry, and B. Feinstein, "The Intrusion Detection Message Exchange Format (IDMEF)," IETF RFC 4765, March 2007, http://www.ietf.org/rfc/rfc4765.txt.

[53] P. Verissimo *et al.*, "MASSIF architecture document," FP7-257475 MASSIF European project, Tech. Rep., April 2012. [Online]. Available: http://www.massif-project.eu/sites/default/files/deliverables/MASSIF_Architecturedocument_v15_final.zip

[54] M. Callau-Zori, R. Jiménez-Peris, V. Gulisano, M. Papatriantafilou, Z. Fu, and M. Patiño Martínez, "STONE: A Stream-based DDoS Defense Framework," in *Proc. of the 28th Annual ACM Symposium on Applied Computing (SAC'13), Coimbra, Portugal*. ACM, March 2013, pp. 807–812.

[55] I. Kotenko and A. Chechulin, "Attack modeling and security evaluation in SIEM systems," *International Transactions on Systems Science and Applications*, vol. 8, pp. 129–147, December 2012.

[56] G. Granadillo, G. Jacob, H. Debar, and L. Coppolino, "Combination approach to select optimal countermeasures based on the rori index," in *Proc. of the 2nd International Conference on Innovative Computing Technology (INTECH), Casablanca, Morocco*. IEEE, September 2012, pp. 38–45.

[57] L. Coppolino, M. Jäger, N. Kuntze, and R. Rieke, "A Trusted Information Agent for Security Information and Event Management," in *Proc. of the 7th International Conference on Systems (ICONS'12), Saint Gilles, Reunion Island*. IARIA, February–March 2012, pp. 6–12.

[58] M. Llanes, E. Prieto, R. Diaz, , L. Coppolino, A. Sergio, R. Cristaldi, M. Achemlal, S. Gharout, C. Gaber, A. Hutchison, and K. Dennie, "Scenario requirements (public version)," FP7-257475 MASSIF European project, Deliverable D2.1.1, April 2011.

[59] R. Rieke, L. Coppolino, A. Hutchison, E. Prieto, and C. Gaber, "Security and reliability requirements for advanced security event management," in *Proc. of the 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS'12), St. Petersburg, Russia, LNCS*, vol. 7531. Springer, October 2012, pp. 171–180.

[60] R. Rieke, "Abstraction-based analysis of known and unknown vulnerabilities of critical information infrastructures," *International Journal of System of Systems Engineering (IJSSE)*, vol. 1, pp. 59–77, 2008.

[61] MASSIF project consortium, "Acquisition and evaluation of the results," FP7-257475 MASSIF European project, Deliverable D2.3.3, September 2013.

[62] E. Prieto, R. Diaz, L. Romano, R. Rieke, and M. Achemlal, "MASSIF: A Promising Solution to Enhance Olympic Games IT Security," in *Proc. of the 7th International and 4th e-Democracy, Joint Conferences, ICGS3/e-Democracy (ICGS3/e-Democracy'11), Thessaloniki, Greece, LNICST*, vol. 99. Springer, August 2011, pp. 139–147.

[63] J. Schütte, R. Rieke, and T. Winkelvos, "Model-based security event management," in *Proc. of the 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS'12), St. Petersburg, Russia, LNCS*, vol. 7531. Springer, October 2012, pp. 181–190.

Author Biography

Roland Rieke obtained his PhD from Philipps-University Marburg in 2014. His thesis was awarded the GFFT prize "Best Dissertation 2015". He works since 1982 as a senior researcher at the Fraunhofer Institute for Secure Information Technology and was deputy head of the department "Trust and Compliance" from 2007-2014. Roland was research director of the project MASSIF (MAnagement of Security information and events in Service InFrastructures), a large-scale European project 2010-2013. He was member of the strategy board of the Effects+ (European Framework for Future Internet Compliance, Trust, Security and Privacy through effective clustering) project and is member of the ERCIM working group on Security and Trust Management.

Maria Zhdanova[1] is researcher at the Fraunhofer Institute for Secure Information Technology in Darmstadt, Germany. She holds a diploma degree in Applied Mathematics. Her research interests are focused on security, resilience and trust in IoT.

Jürgen Repp attended a training as a mathematical-technical assistant in 1981. After joining the SIT predecessor institute in 1983 his field of activity was the development and implementation of switch software and communication protocols. Afterwards he entered the formal methods group in this institute, working in the area of development and application of tools for formal specification and verification of distributed systems. He was the major developer of the SH verification tool. Since 2005 he was also involved in several software security evaluations. Currently he is working on the application of trusting computing technologies and the development of the future TCG software stack.

[1] No photo is available