SecSy: A Security-oriented Tool for Synthesizing Process Event Logs

Thomas Stocker* and Rafael Accorsi

University of Freiburg, Germany
{stocker,accorsi}@iig.uni-freiburg.de

Abstract. SecSy is a highly flexible log synthesis tool, which allows users to configure their simulation procedures in a detailed way, including control flow, data flow and organizational context of business processes. User friendly dialogs provide rich parameter adjustment and intuitive coupling of simulation components. With the possibility of integrating custom transformers, out-of-the-box functionality of SecSy can be extended to fit custom needs.

1 Introduction

Research into business process security and compliance is concerned with the requirements formalization of security (e.g. secrecy, binding of duties) and compliance requirements (e.g. obligations, interdependencies between activities) [4] and the development of well-founded techniques and tools for analyzing, monitoring and auditing these requirements in business process specifications [1]. Here, a particular challenge arises when it comes to testing the effectiveness of monitoring and auditing techniques and corresponding tools [2, 3]. Specifically, to test these tools one needs controllably generated event logs that contain process executions comprising process flexibility and variability on the one hand, and process non-compliance on the other, thereby mimicking structural vulnerabilities, process dynamics, intentional attacks and user errors [5]. Such event logs can serve as input for monitoring, auditing and mining tools, thereby allowing developers to assess their *kill-rate*, i.e. the precision to identify the violation of the designated security and compliance properties or deviations from an original process model.

The concepts for simulating business process flexibility and noncompliance employed by SecSy are novel, both in the security, compliance and business process management communities. SecSy allows for highly-controllable simulation output incorporating conform process behavior with respect to targeted security properties, as well as random failures that may lead to the violation of selected properties and incomplete/distorted process logs. Overall, the generation of "defect" data has been applied to software process improvement in general [6], but has never been seen in the BPM area. We firmly believe that the controlled,

push-button generation of (large) test data is a promising research direction and application domain in business process testing and improvement.

2 Simulation Approach

The overall approach for synthesizing event logs is depicted in Fig. 1. SecSy takes a series of business process specifications as input and generates a process log that contains traces of these specifications. Based upon the security and compliance requirements, deviations from the defined control flow are generated with the help of *transformers* which encode specific trace properties that are either enforced or violated on a random basis. Generated logs are outputted in MXML (a format for process mining), as well as plain text. Ongoing work adds support to Extended Event Streams format (XES).

Processes subject to simulation are considered to be executed within a *context*. While the control flow of a process defines possible execution traces, subjects authorized to execute process activities and objects used by activities are defined by contexts. SecSy allows to define *simulation runs*, each relating to one process model which are processed one after another according to the number of desired traces for each run. This way, the engine is capable of simulating situations in which there is an initial model for planned process behavior, but a variant of this model was executed for some time, possibly due to the activity of an attacker or process variation/flexibility. During the processing of a simulation run, the engine generates valid log traces according to the control flow of the corresponding process and context and then passes them through the trace transformers which apply transformations in a post-processing manner. Trace transformers can remove or add process activities (simulating skipped activities or incomplete logging), as well as change information within traces in a way business related properties like *separation/binding of duties* or authorization constraints are enforced or violated. Information about which transformers have been applied on which traces and if the enforcement or violation was successful is stored in a separate simulation log which can be used to determine the aforementioned kill-rate of detection-mechanisms.

Fig. 2 depicts the configuration panel of the application. SecSy allows for flexible configuration of all required parameters and the creation and editing of corresponding components. Depending on the simulation type (SIMPLE or DETAILED), generated traces contain only timestamps and activity names or additionally information about executing subjects and data items.

The time generator component contains all timing related simulation properties, including the start time for the simulation (date for first trace), the number of cases per day, office-days and -hours, as well as individual activity durations and delays between succeeding activities. In the case of simulation type EXTENDED, the user has to specify a context and a data container. Data containers generate values for data items used during process execution (e.g. credit amount) and store the values until a trace is completed. This way, the consistent usage of attribute values along a complete trace is ensured. A context holds subjects

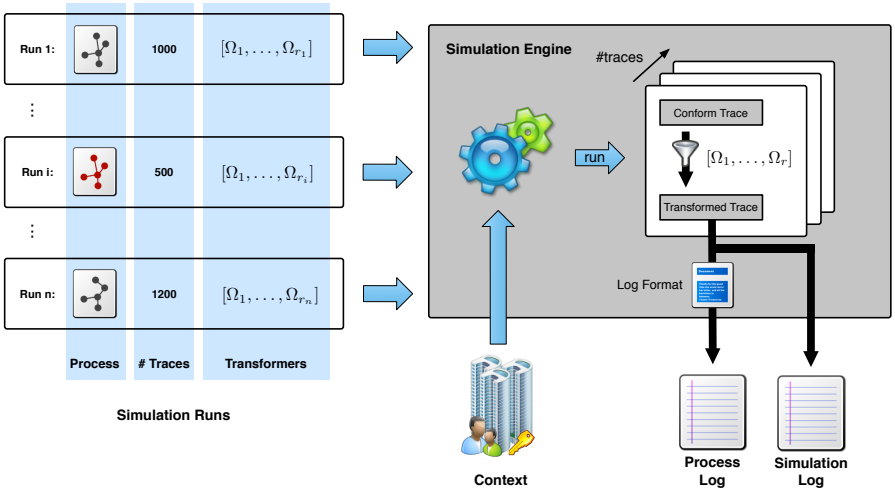

Fig. 1. Overview of the approach.

and their permissions to execute activities and access data elements together with activity data usage (attributes used by activities) information. To specify subject permissions, the user can choose between an access control list or a role based concept, which is particularly helpful for large contexts. Additionally, a context allows to specify constraints on attribute values that can be added to process activities.

3 Trace-Transformations

Trace transformers are used to transform traces in a way specific trace properties are enforced or violated. Transformers operate on valid traces generated during the simulation process and generate modified versions of these traces in a post-processing manner, before they are added to the output log file. Modified fields are locked to prevent further transformers from corrupting the already enforced properties coming from previous applications of transformers. Currently, SecSy comprises the following trace transformers:

1. *delay* inserts a delay in the process execution and can be used to simulate situations in which work is piling up and process execution is slowed down by unusual long delays between single activities.
2. *skip* removes specific activities from generated valid log traces so simulate skipped (mandatory) activities. This transformer adjusts timestamps in a way as if the removed activity never happened.
3. *silent* captures a situation in which a particular activity has not been logged onto the file. In contrast to *skip*, timestamps are not adjusted.
4. *authentication* mimics an access control policy and its violation.

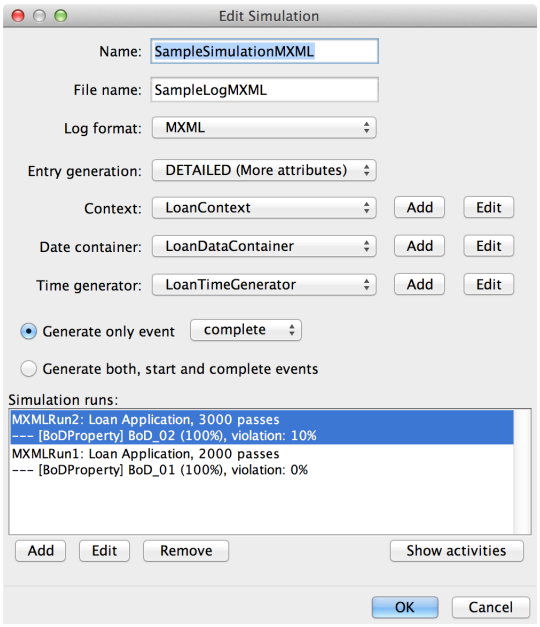

Fig. 2. Screenshot of the configuration panel.

5. *binding of duty* (BOD) mimics the compliance with or the violation of a binding of duty requirement.
6. *separation of duties* can be seen as the opposite of BOD: it states that a particular set of activities must be performed by *different* subjects.

The list of predefined transformers shipped with SecSy can be extended with or refined to other domain-specific transformers. For this, users can define their own transformers (implementing specific interfaces) and, subsequently, add them to SecSy. The procedure of including custom transformers requires writing code for the transformer functionality and providing a graphical frontend for parameter setting. On startup, SecSy looks for custom transformer code and adjusts dialogs for simulation run configuration appropriately. This method allows highly flexible transformer definition and sharing and does not require users to compile their own tweaked version of SecSy. Users can make their custom transformers available in form of class-files other users can readily employ.

4 Availability

SecSy is a standalone, extensible Java application and has been made available on the Open-Source platform Sourceforge (`http://sourceforge.net/projects/secsy/`). Downloads include a runnable jar-version, an OSX application bundle and a sample simulation directory which can be used to get started with SecSy

on basis of preconfigured simulation procedures. This directory also contains an example of a custom trace transformer (DesignatorTransformer). A detailed user guide explaining how to set up and use SecSy can be found on `http://doku.telematik.uni-freiburg.de/SecSy/`. This website also allows users to issue feature requests and bug reports in an interactive way.

References

1. R. Accorsi. Sicherheit im prozessmanagement. digma *Zeitschrift für Datenrecht und Informationssicherheit*, (2):72–76, 2013.
2. R. Accorsi and T. Stocker. On the exploitation of process mining for security audits: The conformance checking case. In *ACM Symposium on Applied Computing*, pages 1709–1716. ACM Press, 2012.
3. R. Accorsi and T. Stocker. On the exploitation of process mining for security audits: The process discovery case. In *ACM Symposium on Applied Computing*. ACM Press, 2013.
4. T. Breaux, A. Antón, and E. Spafford. A distributed requirements management framework for legal compliance and accountability. *Computers & Security*, 28(1-2):8–17, 2009.
5. L. Lowis and R. Accorsi. Finding vulnerabilities in SOA-based business processes. *IEEE Transactions on Service Computing*, 4(3):230–242, August 2011.
6. A. Raninen, T. Toroi, H. Vainio, and J. J. Ahonen. Defect data analysis as input for software process improvement. In O. Dieste, A. Jedlitschka, and N. J. Juzgado, editors, *Conference on Product-Focused Software Process Improvement*, volume 7343 of *Lecture Notes in Computer Science*, pages 3–16. Springer, 2012.