On the Exploitation of Process Mining for Security Audits: The Conformance Checking Case

Rafael Accorsi Thomas Stocker
Department of Telematics
University of Freiburg, Germany
{accorsi, stocker}@iig.uni-freiburg.de

ABSTRACT

Process mining stands for a set of techniques to analyze business process models and logs. However, the extent to which it can be used for security auditing has not been investigated. Focusing on conformance checking and its support in ProM, this paper reports on a case-study in the financial sector applying this technology for the auditing of relevant security requirements. Although the vast majority of requirements could be verified, we notice a large manual effort to carry out the analysis. Moreover, we identify a class of security requirements that demands process discovery for analysis, and elaborate on ways in which process mining could be extended to better suit security analyses.

Categories and Subject Descriptors

H.4.1 [**Information Systems and Applications**]: Office Automation—*Workflow Management*; K.6.5 [**Management of Computing and Information Systems**]: System Management—*Management Audit*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

Business Process Security Audit, Process Mining, Conformance checking, Information Flow Analysis

1. INTRODUCTION

Despite the growing effort in tool design for the secure deployment of business process models, e.g. [1, 5], security incidents and fraud soar [4]. Security audits, whether internal or external, could circumvent these incidents, thereby detecting violations of, e.g., separation of duties or four-eye principle. However, current auditing practices have shortcomings. *Firstly*, audits are largely based on samples, that inevitably provides an incomplete view of the process execution [9]. *Secondly*, there is no specialized tool support for auditing security in workflows systems, leading to largely manual audits [20]. Overall, it is not only the case that audits blatantly fail to detect security violations, they are also time-consuming. (In fact, according to [4], the median time to detect frauds amounts to 18 months.) Auditing automation reduces the corresponding transaction costs, while improving their quality and precision.

Process Mining (PM) stands for automatable techniques to analyze business process models and their execution traces (logs) [21]. PM provides methods for *reconstructing* process models from logs (process discovery), checking the *conformance* of an existing or reconstructed model and logs (conformance checking), and *enhancing* process models based on the results of analysis (process enhancement). Despite the potential to notably improve the quality of security auditing [22], to-date PM has only been modestly employed.

This paper investigates the use of PM for security audits. Focusing on compliance checking and employing its implementation on ProM [25], this paper reports on a case study for the financial sector. The investigation follows the guidelines of [18] for conducting and reporting case studies. In particular, we used interviews to obtain: firstly, the shape of a non-trivial loan application process; secondly, the set of concrete security requirements derived from the set of global business process security requirements [5]; and thirdly, the usual execution characteristics (e.g. number of successful grants and rejected loan applications). Logs are simulated accordingly in an automated manner, including random violations of the security requirements.

In doing so, we obtain the following findings:

- Conformance checking provides a well-founded, powerful tool for the analysis of a relevant set of requirements, including control deviations (skipping and swapping activities), separation of duties, and obligations.
- Showcase technologies, in this case ProM, are not suitable for the industrial size analysis. One does not only encounter scalability problems, but also the manual effort required to interface the different modes and analysis techniques is very high.
- There is a class of properties, namely "non-leak properties" [1], whose analysis requires process discovery.

It should be noted that we employ only a fraction of the existing conformance checking techniques to detect security violations. In fact, we focus on those necessary to detect the most important requirements, leaving out (security-relevant) approaches, such as those analyzing the organizational perspective, which allows for role mining and the circumscription of a subject's social network.

SAC'12 March 26-30, 2012, Riva del Garda, Italy.

Eventually, our goal is to investigate the potential of all PM techniques for conducting security audits, building specialized techniques and tool support. In particular, there is an yet to be unleashed potential to connect powerful information flow techniques, such as InDico [1], RecIF [3], and PDG [27], to security auditing and, thus, business process diagnosis and enhancement.

Related work. The use of process mining for (internal and external) audits has been gaining in momentum. However, to the best of our knowledge there is no previous academic work addressing the intersection of security audits and process mining. In [22], Aalst et al. propose an auditing framework that basically employs process discovery and conformance checking and describe some challenges of applying process mining to auditing. In [13], Jans et al. showed that the use of process mining techniques can be of additional value to reduce the risk of internal fraud in companies by applying their IFR^2 framework on a case company. In order to check if a process functions in a way that corresponds to the designed model they employ process discovery techniques to reconstruct a process model showing the actual process behavior. Besides analyzing relations between process activities and involved persons they use conformance checking for the verification of segregation of duty constraints. This paper is similar to Jans et al. [12], who use process mining for the purpose of internal auditing of a procurement cycle in a financial institution. The main difference is the focus on security.

Paper structure. Sec. 3 presents the considered workflow, corresponding security requirements and simulation details. In Sec. 4 shows how conformance checking is used to verify the security requirements. We discuss these results in Sec. 5 and conclude in Sec. 6.

2. METHOD AND CASE STUDY DESIGN

We employ the guidelines of [18] to conduct the case study. A case study is the most appropriate research methodology for this setting, as its primary objective is exploratory, with a flexible design, and collecting qualitative (instead of quantitative) data. Concretely, the case study encompasses the following steps:

1. *Case study design*: the objectives and objects of the case study are defined. This is given below.
2. *Preparation for data collection* (Sec. 3).
3. *Evidence collection*: carry out the analysis (Sec. 4).
4. *Analysis of collected data* (Sec. 5).

The objective of the case study is to demonstrate the feasibility of conformance checking as a tool for security auditing. Hence, the primary research question is: Does conformance checking generally allow the testing of traditional security requirements? In addition to this, the study also addresses a secondary research question: What kind of properties can be detected using conformance checking? This subordinated question addresses aims to delimit the scope of conformance checking and circumscribe properties whose testing requires other techniques, such as process discovery. The study is therefore explorative.

The "case under study" is the analysis of a real-life business process model and the log file it produces. The process comes from the financial sector and depicts the steps toward granting a loan application. Fig. 4 depicts the formalization of the process in the Business Process Model and Notation (BPMN), whose design has been obtained from and validated in interviews with the bank managers. (Note that each activity is labeled with a letter, which acts as an abbreviation for the task.) This process is to a largest extent automated within the bank, whereas the execution of non-automatable tasks are to be recorded in process logs.

For the sake of privacy, the case study does not build upon the real process logs, but generates such logs through the process simulation. This simulation is controlled and parameterized by typical business conditions, such as the acceptance rate, number of individuals involved, and the aborting quotes. In this case, we simulate a six-month period, which is the usual time span considered by auditors. To also include deviations of the prescribed process, the simulation automatically adds traces in which violations happen.

3. SECURITY REQUIREMENTS AND SIMULATION SETTINGS

The classes of requirements for business processes are [5, 11]:

- *Authorization*: enforcing access control to ensure that only authorized individuals/roles are allowed to execute activities within a process [19].
- *Usage control*: enforcing constraints after the access to data, e.g. data retention and cardinality of use [16].
- *Separation of duty* (SoD): constraints associated with the process to limit the abilities of agents to perform activities, eventually reducing risk of fraud [7]. *Binding of duties* (BoD) denotes the dual constraint, tighting subject/roles to particular sets of activities.
- *Conflict-of-interest* (CoI): preventing the flow of sensitive information among competing organizations/departments taking part of in a process execution [8].
- *Isolation*: executions of the process do not interfere one to the other.

We employ these general requirements to derive the concrete requirements for the loan application process. To do so, we interview not only the bank personnel but also auditors acquainted with the usual requirements.

3.1 Security Requirements

The main constraints for the loan application process are:

(S1) Authorization: Employees have to stick to the role-based access control constraints.

(S2) Usage control: If the credit amount for registered customers exceeds $10.000, the purpose has to be checked.

(S3) Usage control: Consultants must request an acknowledgement from a backoffice clerk before approving loans with a credit amount higher than $30.000.

(S4) CoI: Backoffice clerks must submit the prepared contract to a manager for signing if the credit amount exceeds $50.000.

(S5) Usage control: Once a loan is approved, the contract has to be prepared within the validity period of the approval, which in turn must not exceed 7 days.

(S6) SoD: Signing and acknowledgment activities must be executed by two different persons each.

(S7) SoD: The tasks of checking the financial status and the external rating of a non-registered customer must not be carried out by the same person.

(S8) SoD: A process must never be executed completely by one single user.

(S9) BoD: Loans must be approved by the acknowledging consultant.

(S10) Isolation: Details on the automatically conducted blackbox calculation of the credit parameter must not be leaked.

Authorization constraints, as well as SoD/BoD are typical security controls used in practice to prevent process misuse. Note that **S10** differs from the other requirements in that it ranges not only over the particular value, but also on how the value is computed. Technically, whereas the other requirements stand for the so-called "safety properties", **S10** is a *hyperproperty*. Hyperproperties can be used to define information flow policies specifying how much information can be learned by users of a system [10] or in our case participating actors of a process.

3.2 Managed Process Simulation

To provide a basis for analyzing security properties within a realistic setting, a managed simulation approach was chosen. Based on expert knowledge, a mid-size city-located bank handling a large number of consumer credits was chosen as simulation scenario. The simulation assumes the following constraints:

- Applications within a 6-month period: 21.000. City-located banks typically have a large number of loan applications compared to banks in rural environments with a low percentage of registered customers. (As mentioned above, six-months provide for the usual sample of transaction data for auditors.)
- Rejection rate: 67%. Due to the high number of applications and applicants with bad solvencies, most of the loans are refused.
- Credit division size: 50 employees organized as in Fig. 1. It contains four different roles that be taken during the execution of process activities. Each role within the lattice subsumes the competences of all lower roles. These roles correspond to the BPMN-lanes in Fig. 4.

Generating process logs. We employ the simulation module of the Security Workflow Analysis Toolkit (SWAT) [2] to automatically simulate process executions and generate the logs. For this, we translate the BPMN specification into a Petri net, as they provide a formal execution semantics. This translation is semi-automated and follows [14].

The simulation procedure consists of the two steps depicted in Fig. 2. Step 1 transforms a given BPMN diagram into a Petri net model. In Step 2, process executions are simulated based on additional execution parameters in form of participating task originators, data items/-values to support resource-oriented branching conditions.

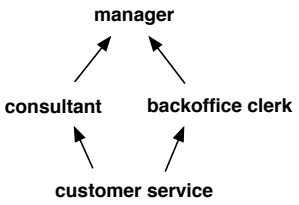

Figure 1: Lattice of business roles.

We chose appropriate activity transition probabilities to meet the simulation constraints stated on the beginning of section. To provide context information in generated log entries, e.g. the activity originator or data attributes, the simulation engine also requires the organizational model (i.e. which employees have which roles) to derive activity execution permissions. Based on given average task execution durations, the simulation engine randomly generates timestamps for log entries. A resultant log entry has the shape:

timestamp	activity	originator	input data	output data

Using a trace-oriented approach, generated workflow executions are stored in form of traces containing subsequent activity executions (so-called *cases*). Generated execution logs contain a list of cases ordered by the starting time. To provide a basis for subsequent analyses within ProM, generated workflow logs are stored in the MXML format, an XML format for the representation of process logs [26].

Simulating violations. During the simulation, we also consider behavior that deviates from the prescribed process model, thereby violating the requirements in Sec. 3.1. To do so, we employ a security testing approach to generate significant test cases, which has been prototypically implemented in SWAT. For example, one violation is assigning the same individuals to tasks that must otherwise be separated (SoD violation), another violation lets the loan approval time delayed for a period greater than 7 days.

Together with the automated simulation of violations, log files were also modified after the generation. In this case, interview partners were made aware of the log format and randomly added violations (without violating the well-formedness of entries). The goal here was to insert security incidents we were not aware of.

For both the automated and manual violation generation, the traces containing a violation were marked as such in order to keep track of where they take place. To simulate a real audit situation, we were using a log file where violation information has been stripped off.

3.3 Conformance Checking and ProM

Conformance checking approaches measure the adequacy of a process log and a process model, or verify constraints on execution traces alone. Conformance checking can be used to find traces that deviate from the prescribed process, and to check if execution traces comply with resource-oriented constraints. This section summarizes the main techniques for conformance checking and their support in ProM [25].

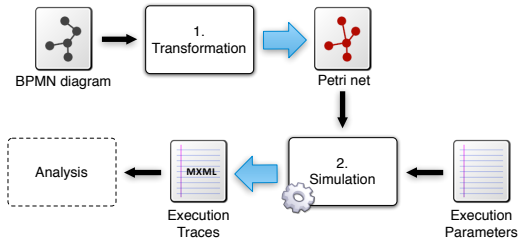

Figure 2: Simulation procedure.

Activity	Count	Affected Traces
C	5	308, 2835, 5818, 15731, 14103, 14915
E	4	13585, 4221, (4842, 12693)
F	2	(9420, 17459)
H	2	8223
C or E	4	(12872, 17437, 16184, 13911)

Table 1: Identified control flow deviations.

Employee ID	Business Role
1 - 6	Manager
7 - 21	Backoffice Clerk
22 - 36	Consultant
37 - 50	Customer Service

Table 2: Business roles of process executors.

Control flow deviations in process logs represent unintended process executions and, thus, can be of relevance regarding security auditing. A common method for finding such deviations is to *replay* logged process executions on an existing Petri net model. [17] propose an approach that besides providing detailed information on different fitness and appropriateness metrics, also supports replaying and the selection of deviating traces. This approach is also available within ProM as the *Conformance Checker* plugin. The use of replaying techniques to find skipped activities or other control flow deviations can effectively assist auditors in distinguishing between executions that comply with the model from those that do not. In particular, one can single out and focus on "outliers" possibly encompassing violations of security requirements.

There is limited tool support the verification of authorization constraints so far. Within ProM, the plugin *Originator by Task Matrix* can provide information on which originators were involved in which process activities to decide on unauthorized executions, but still requires high manual effort in finding violations. Baumgrass et al. [6] proposed an approach for directly checking RBAC policies on process execution traces. Analysis tools accepting an access control matrix together with a process log that can give evidence on access control violations are helpful for auditors in evaluating the efficiency of installed enforcement mechanisms. Unfortunately such a tool is not available yet.

Using formal descriptions of security properties, model checking techniques are available to formally verify (non)-conformance. Van der Aalst et al. [24] developed a LTL based language to formulate properties in the context of event logs. Together with a verification engine, this language has been made available in form of the *LTL Checker* plugin within ProM. BoD, SoD and control flow constraints can be verified by specifying appropriate LTL formulas.

Timely completion of successive tasks and validity periods of incorporated/generated data items can be critical regarding proper process completion. Similar to the logical definition of other resource-oriented constraints, there are frameworks for timed constraints too. With the help of the *CLIMB* framework developed by Montali [15], business constraints can be modeled and verified using computational logic. The included proof procedure *SCIFF* is available in form of the ProM plugin *SCIFF Checker* and capable to classify a set of MXML traces as (non)-compliant with respect to a declarative business rule.

4. CHECKING SECURITY PROPERTIES

In this section, different kinds of security properties are checked using built-in tool support of the ProM framework.

4.1 Finding Control-Flow Deviations

Replaying is employed to find control flow deviations that lead to usage control violations. As a preprocessing step for replaying the simulated process traces on the loan application process, the BPMN model in Fig. 4 is transformed into the Petri net, which is omitted here for space constraints. The resultant Petri net can be used for replaying. The *Conformance Checker* plugin determines a fitness value of 0.999922. Perfect fitness indicates that all log traces could be successfully replayed on the model, so in this case there are some "faulty" traces that have to be examined in detail.

Based on the input Petri net, the *Conformance Checker* provides information about missing and remaining tokens during unsuccessful replays which can be used to identify the deviation type. The selection of traces that could not successfully be replayed on the input model can be done directly in the plugin by selecting the "fitting" traces first and then inverting the selection. Table 1 shows all skipped task deviations together with their occurrence number and the affected process traces. Grouped instances represent identical process instances. (Note that the activities in the process model are labeled with capital letters.)

Fig. 3 depicts the case 15731 that shows an unintended process execution by skipping activity C. The specification in Fig. 4 requires activity D (check credit purpose) after activity C. Since checking the internal rating of a customer is a mandatory part of a loan application process to minimize the risk of nonpayment, the reasons for the trace 15731 have to be detected within an internal audit to estimate potential losses or financial risks. In case of refusal, skipped tasks do not necessarily have consequences, but as Fig. 3 shows, the loan has been approved. Possible reasons for such control flow deviations can be erroneous process handling within the process engine that allows executions without checking preconditions or intended process misuse.

4.2 Testing Authorization Constraints

To check the compliance of the workflow log to constraint **(S1)**, each process trace and activity therein must be checked as to whether the user was authorized to perform the activity. The membership of employees working on the loan application process is listed in Table 2.

Using the *Originator by Task* matrix, information on which users (so-called "originators") executed which tasks is available. To check whether there are executions in which an unauthorized user performs an activity, we have to manually analyze the matrix with respect to the predefined business roles. Table 3 shows the identified violations for backoffice employees together with their frequency and corresponding process instances and the total number of violations per ac-

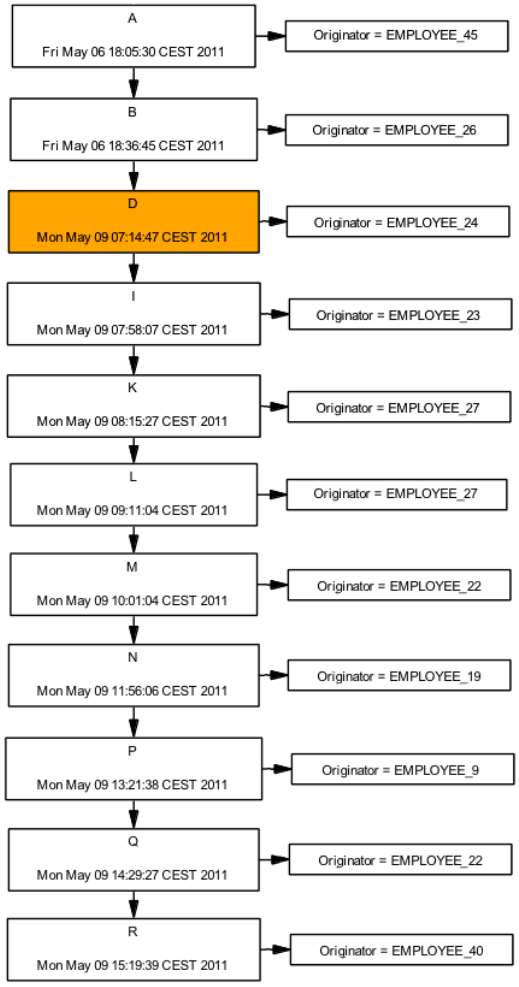

Figure 3: Process execution skipping activity *C*.

tivity. With this information, decisions on further violation handling can be made. Not in every case violations are intentional, but they result from process misuses due to unclear definitions of responsibilities. Distinguishing between intended and unintended violation is a crucial part for secure auditing that falls in the area of internal audit.

4.3 Testing BoD and SoD Constraints

Considering the security requirements for the loan application process, there are three SoD constraints **(S6-S8)** and one BoD constraint **(S9)** formulating obligations for task executions. Since each of the constraints **(S6-S9)** can be described as a formula in LTL, the compliance of execution traces can be verified with the *LTL Checker* plugin. Using the predefined formula *exists person doing task X and Y*, constraint **(S6)** and **(S7)** can be checked by adjusting X and Y accordingly. Table 4 depicts the results.

To check constraint **(S8)** the formula $\exists p : \Box(executor == p)$ is used in the *LTL Checker* plugin. Applied to the log, no traces can be found in which every activity was executed by the same person. Generally, adding customized formulae is a straightforward procedure and advisable.

Employee	Task	Number	Instance
ID_7	Q	1	6681
ID_8	K	2	6045, 16918
	S	1	14921
ID_9	C	1	4743
ID_{10}	S	1	6219
ID_{11}	Q	1	13755
ID_{14}	B	1	9480
	S	1	14149
ID_{15}	C	1	3548
	D	1	16114
	Q	1	12052
ID_{16}	B	1	11840
ID_{20}	Q	1	4800

violations	B	C	D	K	Q	S
BO	2	2	1	2	4	3

Table 3: Identified access control violations of back-office (BO) employees.

	X	Y	True	False
(S6)	Ack. BO	Ack. CO	1	20999
	Sign Manager	Sign BO	0	21000
(S7)	Check Status	Check Rating	1	20999
(S9)	Ack. CO	Approve Loan	20992	8

Table 4: Traces violating SoD and BoD constraints.

4.4 Testing Data Constraints

Branching decisions within workflows often depend on data items used within process execution. The loan application process contain various branching situations, where three are considered in the security requirements in Sec. 3.1. Depending on the credit amount, additional checks are scheduled as security measures in **(S2-S4)**.

To find violations of constraint **(S2)**, the credit amounts of all traces that contain a transition from activity C (check internal rating) to activity I (calculate creditworthiness characteristic) are examined. Filtering these traces can be done with the help of the *LTL Checker* plugin. After applying the formula $\Diamond CI$ on the process log, a set of 1189 traces is isolated. In a further analysis step, we check whether this set contains traces where the credit amount is equal or higher $10.000. With the help of additional LTL formulas considering the *creditAmount* data item which is set by activity B (prepare loan application), we find out that merely 157 executions are compliant. Analogously, testing the constraints **(S3)** and **(S4)** leads to portions of 45,92% and 19,79% violating execution traces respectively. This is a considerable rate. If found in a real-life log, this would mean that the bank had serious problems in enforcing security measures.

Within an internal audit it would be interesting why the majority of process executions do not stick to the branching constraint. A possible reason could be that the loan process is not completely automated, thereby allowing users to decide on their own risk which path to take.

4.5 Testing Time Constraints

Time constraint **(S5)** relates to the activities *Approve Loan (L)* and *Prepare Contract (N)* and requires the completion of N within a period of 7 days after L was completed. To check this, the set of traces used as input for the *SCIFF Checker* is first reduced to traces containing the activities L and N which can be done by the *LTL Checker* plugin using the logical formula $\Diamond L \Diamond N$. This reduces the candidate set from 21.000 to 7.036 traces and saves computation time. The *SCIFF Checker* can be applied to MXML logs and allows the specification of user-defined rules. For checking **(S5)**, the rule *"**IF** activity **L** is performed at time t_0 **THEN** activity **N** should be performed before $t_0 + 7days$"* is applied

		Approve Loan (L)	Prepare Contract (N)
1	Time	03.01.2011 14:26	10.01.2011 17:03
	Originator ID	29	7
2	Time	07.03.2011 15:53	14.03.2011 18:57
	Originator ID	23	8
3	Time	06.04.2011 19:15	14.04.2011 10:06
	originator	34	12
4	time	14.04.2011 12:01	21.04.2011 15:19
	originator	28	17

Table 5: Traces violating time constraint (S5).

Date	Trace	Activities	Employee ID
2011-01-03	17	A → B → C → I → S	32
2011-01-07	629	A → B → C → I → S	28
2011-01-07	682		
2011-01-07	775		

Table 6: Aborted processes containing activity I that were executed by a single actor.

on the candidate set. The results of the logical verification procedure is displayed in form of a pie chart and shows that 4 traces violate the specified rule. Table 5 shows these traces together with timing information and originators.

To find out where delays occurred and which originators are responsible for them, a *Dotted Chart analysis* can conducted to visualize task execution times and inter-task delays. Together with the originators, this provides valuable information for auditing.

4.6 Testing Isolation Constraints

Constraint **(S10)** claims that no information about the blackbox calculation can be leaked. In contrast to other constraints, it does not relate to specific data elements or other process resources specified in the process. Information leakage in this context can be interpreted as the possibility to gather information on the calculation procedure by analyzing the generated outputs of task I with respect to the given inputs. Concretely, it should not be possible to find out how the credit parameter changes depending on the internal rating, the acceptability report and the credit amount. Obviously it cannot be prevented that consultants and backoffice clerks gather some knowledge about the calculation since they have to prepare the inputs and work with the output, but to prevent manipulation they should not be able to find out calculation details.

A possible approach would be to test if task I is repeated several times within one process execution. Since the loan application process does not allow a repetition of I, these cases represent control flow deviations. The analysis for finding control flow deviations only yielded traces containing skipped tasks, so this possibility is ruled out. Another possibility is to search for cases where the same actor starts a loan application process, triggers the calculation procedure and then aborts the process execution. Repeatedly misusing the process like this can enable users to learn how the inputs influence the output of the credit parameter calculation. To find such cases, all aborted process instances that contain the activity I are identified first using the *LTL Checker* plugin, which results in 2678 process traces. Applying the LTL formula to identify traces where all activities are conducted by the same actor yields the traces in Table 6.

There are two different users that started and aborted a loan application while executing all activities on their own. Especially for the last three cases (all aborted traces were started on the same day), the reasons for such practice should be clarified. Generally, it is impossible to detect and quantify isolation properties using conformance checking.

5. DISCUSSION

Conformance checking approaches could be successfully used to verify the desired security requirements for the loan application process. In fact, they provide an appropriate basis for requirements **(S1-S9)**, where a testing strategy could be found and successfully applied. (See below for **(S10)**.) These requirements correspond to *safety* properties and it seems reasonable that the trace-wise analysis provided by conformance checking suffices to detect their violations. We could indeed detect all the violations of this requirements, both those that were generated automatically and those added by the experts. Still, we identify three major issues, which can be understood as "threats" to the validity of the case study (whenever transferred to real-life):

- The application of existing tools involves extensive adjustments or manual steps. Put another way, toolchains tailored for security are missing.
- We generate logs in a technically very controlled format. In real-life systems, there are several steps necessary to bring them down to mining-aware log format, e.g., MXML, or even generate logs coherent at all.
- Mining results are often presented at a low level and require abstraction. In particular, it is not reasonable to display LTL formulae or Petri net patterns to auditors, who are potentially not literate in these formalisms.

We firmly believe that these are technical shortcomings which can be dealt with a set of purpose-specific tools coupled with techniques to "plug-in" the tool in different process-aware enterprise architectures and systems. They would not enhance the power of tests, but make it considerably handy.

In contrast to the other requirements, **(S10)** – and more generally, isolation requirements – stand for hyperproperties which must be tested on "sets of sets" of traces. In fact, even though we attempt to tackle a special form of isolation in Sec. 4.6, the success is modest. Addressing this point is necessary to enlarge the kinds of constraints to be detected.

The current support for such testing is not sophisticated and may require the analysis of the process model obtained by process discovery (instead of the sole traces). By that we do not only mean the extraction of process models from logs represented as Petri nets, but the extraction of other types of models that allow for reasoning about such hyperproperties. Below we list some promising alternatives:

Information flow nets (IFnet) [1]: IFnet is a dialect of Petri net that allows the reasoning about information flow properties in business process models based upon the concept of "Place-based Non-Interference". In particular, it allow to determine whether there are harmful structural patterns that abet the leak of information from a classified domain to a public domain. The great challenge here is to extract precise models that faithfully reflect those in the log.

Propagation graphs [3]: Propagation graphs denote the flow of data during the execution of the process, thereby zooming in on data-awareness. While we do consider data constraints, propagation graphs could provide for a more expressive and complete basis to reason about authorization, role-base access control, and delegation.

Dependence graphs [27]: A procedure dependence graph (PDG) is a graph in which the *vertices* represent the individual statements and predicates of the procedure and the *edges* represent the control and data dependences among the procedure's statements and predicates. The collection of PDG for a program is called "system dependence graph". Such program representation has been successfully used to show several information flow properties through slicing. While building a kind of program analysis, it could provide a powerful basis for the analysis of discovered model.

6. SUMMARY AND FURTHER WORK

This paper reports on a case study using conformance checking techniques for security auditing. In particular, it employs a bank scenario and a real-life loan application process to synthesize execution traces which are tested against the common security requirements. Conformance Checking and ProM indeed provide a solid basis for carrying out the main analysis steps. In fact, we could verify the vast majority of requirements by combining different techniques, while informally circumscribing a class of properties for which conformance checking does not suffice.

The focus of further work is twofold: *firstly*, enhance the analysis with conformance checking. For example, in the case study we do not consider the delegation of rights (and their revocation). Attaching this analysis is relevant in several real-life applications. *Secondly*, investigate the use of process discovery for security auditing. In particular, we examine precision of discovered models and devise methods to tackle over- and underfitting. A particular challenge we see here is the consideration of data-flow information from logs.

7. REFERENCES

[1] R. Accorsi and C. Wonnemann. Strong non-leak guarantees for workflow models. In *Symp. on Applied Computing*, pages 308–314. ACM, 2011.

[2] R. Accorsi, C. Wonnemann, and S. Dochow. SWAT: A security workflow toolkit for reliably secure process-aware information systems. In *Conf. on Availability, Reliability and Security*, pages 692–697. IEEE, 2011.

[3] R. Accorsi, C. Wonnemann, and T. Stocker. Towards forensic data flow analysis of business process logs. In *Conf. on Incident Management and Forensics*. IEEE, 2011.

[4] Association of Certified Fraud Examiners. Report to the nations on occupational fraud and abuse. `http://www.acfe.com/uploadedFiles/ACFE_Website/Content/documents/rttn-2010.pdf`, 2010.

[5] V. Atluri and J. Warner. Security for workflow systems. In *Handbook of Database Security*, pages 213–230. Springer, 2008.

[6] A. Baumgrass, T. Baier, J. Mendling, and M. Strembeck. Conformance checking of RBAC policies in process-aware information systems. In *BPM'11 Workshops* (to appear).

[7] R. Botha and J. Eloff. Separation of duties for access control enforcement in workflow environments. *IBM Systems J.*, 40(3):666–682, 2001.

[8] D. Brewer and M. Nash. The Chinese-wall security policy. In *IEEE Symp. on Security and Privacy*, pages 206–214, 1989.

[9] A. Carlin and F. Gallegos. IT audit: A critical business process. *IEEE Computer*, 40(7):87–89, 2007.

[10] M. R. Clarkson and F. B. Schneider. Hyperproperties. *J. of Computer Security*, 18(6):1157–1210, 2010.

[11] G. Herrmann and G. Pernul. Viewing business-processes security from different perspectives. *Int'l J. of Electronic Commerce*, 3(3):89–103, 1999.

[12] M. Jans, B. Depaire, and K. Vanhoof. Does process mining add to internal auditing?. In *BMMDS/EMMSAD '11*, pages 31–45, 2011.

[13] M. Jans, N. Lybaert, K. Vanhoof, and J. van der Werf. A framework for internal fraud risk reduction at it integrating business processes. In *Int'l J. of Digital Accounting Research*, volume 9, pages 1–29, 2009.

[14] N. Lohmann, E. Verbeek, and R. Dijkman. Petri net transformations for business processes - A survey. In *Trans. on Petri Nets and Other Models of Concurrency*, volume 5460 of *LNCS*, pages 46–63. Springer, 2009.

[15] M. Montali. *Specification and Verification of Declarative Open Interaction Models*, volume 56 of *LNBIP*. Springer, 2010.

[16] A. Pretschner, M. Hilty, and D. Basin. Distributed usage control. *Comm. of the ACM*, 49(9):39–44, 2006.

[17] A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Inf. Systems J.*, 33(1):64–95, 2008.

[18] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Soft. Eng.*, 14(2):131–164, 2009.

[19] R. Sandhu and P. Samarati. Access control: Principles and practice. *IEEE Comm. Mag.*, 32(9):40–48, 1994.

[20] A. Sayana. Using CAATs to support is audit. *Inf. Systems Control J.*, 1, 2003.

[21] W. van der Aalst. *Process Mining*. Springer, 2011.

[22] W. van der Aalst, K. van Hee, J. van der Werf, and M. Verdonk. Auditing 2.0: Using process mining to support tomorrow's auditor. *IEEE Computer*, 43(3):90–93, 2010.

[23] W. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.

[24] W. M. P. van der Aalst, H. T. de Beer, and B. F. van Dongen. Process mining and verification of properties: An approach based on temporal logic. In *OTM Conferences*, volume 3760 of *LNCS*, pages 130–147. Springer, 2005.

[25] B. van Dongen, A. de Medeiros, H. Verbeek, A. Weijters, and W. van der Aalst. The ProM framework: A new era in process mining tool support. In *Conf. on Applications and Theory of Petri Nets*, volume 3536 of *LNCS*, pages 444–454. Springer, 2005.

[26] B. van Dongen and W. van der Aalst. A meta model for process mining data. In *Workshop on Enterprise Modelling and Ontologies for Interoperability*, volume 16, 2005.

[27] D. Wasserrab, D. Lohner, and G. Snelting. On PDG-based noninterference and its modular proof. In *Workshop on Programming Languages and Analysis for Security*, pages 31–44. ACM, 2009.

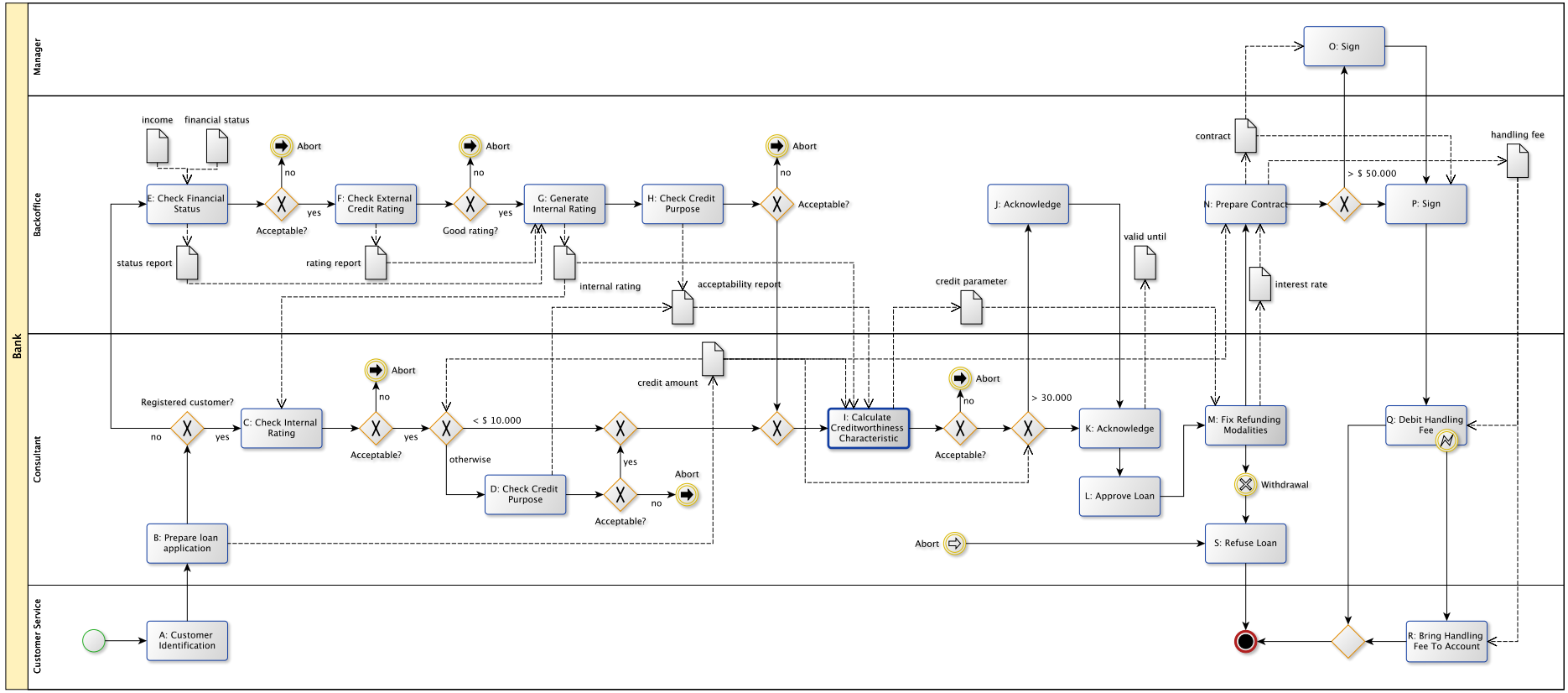

Figure 4: Loan application process as BPMN diagram including data elements.