Security-Sensitive Tackling of Obstructed Workflow Executions

Julius Holderer[1], Josep Carmona[2], and Günter Müller[1]

[1] University of Freiburg, Germany
{holderer, mueller}@iig.uni-freiburg.de
[2] Universitat Politècnica de Catalunya, Barcelona, Spain
jcarmona@cs.upc.edu

Abstract. Imposing access control onto workflows considerably reduces the set of users authorized to execute the workflow tasks. Further constraints (e.g. Separation of Duties) as well as unexpected unavailabilty of users may finally obstruct the successful workflow execution. To still complete the execution of an obstructed workflow, we envisage a hybrid approach. If a log is provided, we partition its traces into "successful" and "obstructed" ones by analysing the given workflow and its authorizations. An obstruction should then be solved by finding its nearest match from the list of successful traces. If no log is provided, we flatten the workflow and its authorizations into a Petri net and encode the obstruction with a corresponding "obstruction marking". The structural theory of Petri nets shall then be tweaked to provide a minimized Parikh vector, that may violate given firing rules, however reach a complete marking and by that, complete the workflow.

Keywords: workflow satisfiability, authorization, obstruction, Petri nets

1 Introduction

From the Société Générale scandal with loss of nearly five billion Euro caused by shuffling transactions [8] to more recent scandals, for instance in the automotive industry (e.g. the "Dieselgate" [18]) - the increasing number of corporate fraud cases underline the growing demand for security and control in enterprises and their corresponding information systems. These systems increasingly adapt to a process-oriented view to reach the intended business goals. These so called process-aware information systems (PAIS) [14] can help to mitigate such fraudulent situations by enhancing workflows with authorization constraints. In this respect security in business processes gains more and more importance [22, 1]. Classic computer security [4] usually follows the CIA-triade, trying to achieve or sustain confidentiality, integrity and availability, or simply "keeping bad things from happening". Security in business processes, however, should also consider to "make good things happen" by reaching the intended business goals in completing corresponding processes.

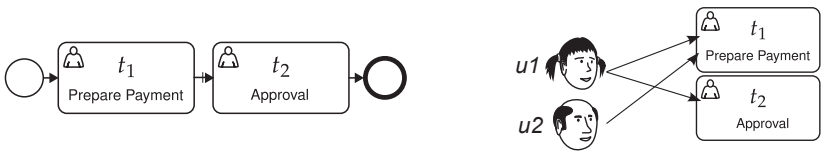
The interplay of security in business processes and this notion of process availability can be shown by analysing the impact of introducing authorization

in PAIS to achieve confidentiality and integrity. First, access control policies are added on top of users contributing in the process, controlling who is authorized to perform which task. On top of that, further constraints are defined, for instance "separation of duties" (SoD) constraints [5] or contrary "binding of duties" (BoD) constraints. Moreover, users can be on vacation or become ill. In this way, the set of authorized users to execute the tasks in a process is drastically reduced and can result in a state where no user can be found to execute the given task at hand, obstructing workflow execution.

An obstruction describes a state of a workflow instance where the enforcement of the authorization policy conflicts with the business objectives. At the control-flow level, the business objectives can be achieved by executing a task t but at the task-execution level there is no user who is authorized to execute t without violating the given authorization policy [2]. The following minimal example depicts this notion of (un-)satisfiability in workflows.

1.1 Running Example

Figure 1 illustrates a simplified payment workflow and a user-task assignment. Now, we add an SoD constraint for $t1$ and $t2$, meaning that the preparation of payment needs to be done by a different user than the one who approves the payment. Given the user-task assignment in Fig. 1(b), if $u2$ executes $t1$, $t2$ can be performed by $u1$. If $u1$ executes $t1$, she can not execute $t2$ due to the imposed SoD constraint, although she basically is authorized to perform this task. $u2$ can neither execute $t2$, since he is not authorized at all. This situation indicates an *obstruction* of the workflow resulting from given authorization constraints [2, 3, 13].



(a) Simplified Payment Worfklow (b) User-Task Assignment

Fig. 1: Simplified Payment Workflow based on [6]

1.2 Related Work

Literature regarding satisfiability of authorization constrained workflows mainly offers theoretical approaches so far [13]. In particular, research related to the so called Workflow Satisfiability Problem (WSP) (see Section 2) shows that (under

the assumption that P≠NP) the NP-complete WSP is efficiently solvable for a growing number of constraint types. Regarding access control systems in general, there mainly exist two approaches for the case when no user is available to access a certain object [10]: either alternative constraints are defined ("Break-the-Glass") that override existing policies or another user is empowered to access the object by use of delegation. However, classic delegation requires the delegator to be available to perform the delegation and involves the danger that delegation capabilities are misused (e.g. collusion [21]). Considering these deficits, the approach of Crampton et al. [10] suggests the concept of auto-delegation, in which qualifications are introduced that indicate a potential delegatee. Examples on how the qualification hierarchy may be computed based on an access control-model are given. However, the auto-delegation mechanism only exists as a first concept so far, which seems promising for the use in PAIS. In summary, the state of the art on workflow satisfiability only scarcely solves the consequent practical problems in terms of obstructions in workflow executions at runtime. Therefore, we envisage to develop an approach that caters for the detection of obstructions *and* policy-wise sound workarounds that allow their execution.

1.3 Structure

This paper aims to show our intended solution to this problem. We first state Petri nets, events logs, authorization and structural theory formally and introduce the corresponding terminology in Section 2. On top of that, we present our approach to tackle obstructed workflows based on the model and logs in Section 3 and show its potential applications in Section 4. Section 5 concludes and presents further research steps on the topic.

2 Preliminaries

We first give the definition of a Petri net to model workflows with a clear execution semantics. Then, we introduce users, user-task authorization and define SoD and BoD Constraints. In this way, we are able to grasp unsatisfiability in workflows formally, leading us to introduce structural theory as a way to encounter this (see Section 3.1).

2.1 Petri Nets and Event Logs

Definition 1 (Petri net). *A Petri net [15] is a 4-tuple* $N = \langle P, T, \mathcal{F}, m_0 \rangle$*, where* P *is the set of places,* T *is the set of transitions, satisfying* $P \cap T = \emptyset$ *and* $\mathcal{F} : (P \times T) \cup (T \times P) \to \{0, 1\}$ *is the flow relation, and* m_0 *is the initial marking. A marking is an assignment of a non-negative integer to each place. If* k *is assigned to place* p *by marking* m *(denoted* $m(p) = k$*), we say that* p *is marked with* k *tokens. Given a node* $x \in P \cup T$*, its pre-set and post-set are denoted by* ${}^\bullet x$ *and* $x^\bullet$ *respectively.*

A transition t is enabled *in a marking m when all places in ${}^\bullet t$ are marked. When a transition t is enabled, it can* fire *by removing a token from each place in ${}^\bullet t$ and putting a token to each place in $t^\bullet$. A marking m' is* reachable *from m if there is a sequence of firings $t_1 t_2 \dots t_n$ that transforms m into m', denoted by $m[t_1 t_2 \dots t_n\rangle m'$. A sequence of transitions $t_1 t_2 \dots t_n$ is a* feasible sequence *if it is firable from m_0.*

Definition 2 (System Net, Full Firing Sequences). *A system net defines a set of sequences, each one starting from the initial marking and ending in the final marking. A system net is a tuple $SN = (N, m_{start}, m_{end})$, where N is a WF-net and the two last elements define the initial and final marking of the net, respectively. The set $\{\sigma \mid (N, m_{start})[\sigma\rangle(N, m_{end})\}$ denotes all the full firing sequences of SN.*

Definition 3 (Trace, Event Log, Parikh vector). *Given an alphabet of events $T = \{t_1, \dots, t_n\}$, a trace is a word $\sigma \in T^*$ that represents a finite sequence of events. An* event log *$L \in \mathcal{B}(T^*)$ is a multiset of traces. $|\sigma|_a$ represents the number of occurrences of a in σ. The Parikh vector of a sequence of events is a function $\widehat{\ }: T^* \to \mathbb{N}^n$ defined as $\widehat{\sigma} = (|\sigma|_{t_1}, \dots, |\sigma|_{t_n})$. For simplicity, we will also represent $|\sigma|_{t_i}$ as $\widehat{\sigma}(t_i)$. The support of a Parikh vector $\widehat{\sigma}$, denoted by supp$(\widehat{\sigma})$ is the set $\{t_i | \widehat{\sigma}(t_i) > 0\}$.*

Workflow processes can be represented in a simple way by using workflow nets (WF-nets) [19]. A WF-net is a Petri net with a place *start* (denoting the initial state of the system) with no incoming arcs and a place *end* (denoting the final state of the system) with no outgoing arcs, and every other node is within a path between *start* and *end*. The transitions in a WF-net represent tasks. For the sake of simplicity, the techniques of this paper assume that models are specified with WF-nets.

2.2 Security in Workflows

To connect WF-nets with users and authorization, we adapt the definitions from [20]. Further constraints regarding workflow satisfiabilty analysis have already been investigated [9]. However, in this paper we focus on the SoD related binary constraints, which are sufficient to reach an obstructed state.

Definition 4 (Authorization). *A configuration is given by a tuple $\langle U, B\rangle$, where $U \subseteq \mathcal{U}$ is a set of users and $B = \{\rho_1, \cdots, \rho_m\} \subseteq B$ is a set of binary relations such that $\rho_i \subseteq U \times U (i \in [1, m])$. Furthermore, we assume that B contains two predefined binary relations $=$ and $\neq$, which denote equality (for BoD) and inequality (for SoD), respectively. A configuration $\langle U, B\rangle$ defines the environment in which a workflow is to be run.*

A workflow is represented as a tuple $\langle N, TA, C\rangle$, where N is a WF-net, $TA \subseteq \mathcal{U} \times T$ is the user-task authorization where $(u, t) \in TA$ indicates that a user u is authorized to perform transition or task t, and C is a set of constraints.

Definition 5 (Constraints, Workflow Satisfiability). *Each of the constraints takes one of the following forms:*

1. $\langle \rho(t_1,\ t_2) \rangle$*: the user who performs* t_1 *and the user who perform* t_2 *must satisfy the binary relation* ρ*.*
2. $\langle \rho(\exists X,\ t) \rangle$*: there exists a task* $t' \in X$ *such that* $\langle \rho(t',\ t) \rangle$ *holds, i.e., the user who performs* t' *and the user who performs* t *satisfy* ρ*.*
3. $\langle \rho(t,\ \exists X) \rangle$*: there exists a task* $t' \in X$ *such that* $\langle \rho(t,\ t') \rangle$ *holds.*
4. $\langle \rho(\forall X,\ t) \rangle$*: for each task* $t' \in X, \langle \rho(t',\ t) \rangle$ *must hold.*
5. $\langle \rho(t,\ \forall X) \rangle$*: for each task* $t' \in X, \langle \rho(t,\ t') \rangle$ *must hold.*

Consider the simplified payment workflow in Figure 1a. Let $t_{1_{prepare}}, t_{2_{approve}}$ denote the two tasks in the workflow. The SoD constraint of the workflow can be represented in tuple-based specification $\langle \neq (t_{1_{prepare}},\ t_{2_{approve}}) \rangle$.

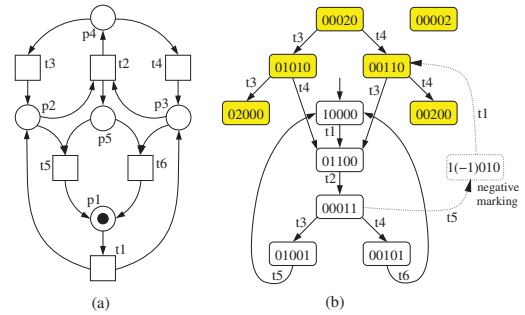
A plan P *for workflow* $W = \langle N, TA, C \rangle$ *is a subset of* $\mathcal{U} \times T$ *such that, for every task* $t_i \in T$*, there is exactly one tuple* $(u_a,\ t_i)$ *in* P*, where* $u_a \in \mathcal{U}$*. Intuitively, a plan assigns exactly one user to every task in a workflow.* Given a workflow $W = \langle N, TA, C \rangle$ and a configuration $\Gamma = \langle U, B \rangle$, we say that a plan P *is valid for* W under Γ if and only if for every $(u,\ t) \in P, u$ is an authorized user of t and no constraint in C is violated. We say that W is *satisfiable* under Γ if and only if there exists a plan P that is valid for W under Γ.

The *Workflow Satisfiability Problem* (WSP) checks whether a workflow W is satisfiable under a configuration Γ. Given configuration $\langle U, B \rangle$, checking whether W is satisfiable under Γ is equivalent to checking whether there is a valid plan for W under Γ. Note that there can be multiple valid plans for a workflow W under a configuration. In fact, it is the existence of multiple valid plans that makes it possible for W to be completed even if a number of users are absent. Therefore, the notion of *resilience* in workflows is introduced [20]: Given a workflow W and an integer $n \geq 0$, a configuration $\langle U, B \rangle$ is *resilient* for W up to n absent users if and only if for every size-n subset U' of U, W is satisfiable under $\langle (U - U'), B \rangle$. In our example, the absence of either $u1$ or $u2$ would result in an unsatisfiable workflow wherefore it is not resilient for $n > 0$ absent users. However, although the regarded workflow is satisfiable, it still contains an obstruction (cf. Section 1.1). We show how we plan to capture such obstructions based on a WF-net with an "obstruction marking" in Section 3.1.

2.3 Structural Theory of Petri Nets

Let $N = \langle P, T, \mathcal{F}, m_0 \rangle$ be a Petri net. Given a feasible sequence $m_0 \xrightarrow{\sigma} m$, the number of tokens for a place p in m is equal to the tokens of p in m_0 plus the tokens added by the input transitions of p in σ minus the tokens removed by the output transitions of p in σ:

$$m(p) = m_0(p) + \sum_{t \in {}^\bullet p} |\sigma|_t \ \mathcal{F}(t,p) - \sum_{t \in p^\bullet} |\sigma|_t \ \mathcal{F}(p,t)$$



$$\begin{matrix} m & & m_0 & & & & N & & & & & & \boldsymbol{\sigma} \end{matrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 & 0 & +1 & +1 \\ +1 & -1 & +1 & 0 & -1 & 0 \\ +1 & -1 & 0 & +1 & 0 & -1 \\ 0 & +1 & -1 & -1 & 0 & 0 \\ 0 & +1 & 0 & 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

(c)

Fig. 2: (a) Petri net, (b) Potential reachability graph, (c) Marking equation.

The marking equations for all the places in the net can be written in the following matrix form (see Fig. 2(c) as an example): $m = m_0 + \mathbf{N} \cdot \widehat{\sigma}$, where $\mathbf{N} \in \mathbb{Z}^{P \times T}$ is the *incidence matrix* of the net: $\mathbf{N}(p,t) = \mathcal{F}(t,p) - \mathcal{F}(p,t)$.

If a marking m is reachable from m_0, then there exists a sequence σ such that $m_0 \xrightarrow{\sigma} m$, and the following system of equations has at least the solution $X = \widehat{\sigma}$

$$m = m_0 + \mathbf{N} \cdot X \tag{1}$$

If (1) is infeasible, then m is not reachable from m_0. The inverse does not hold in general: there are markings satisfying (1) which are not reachable. Those markings are said to be *spurious* [17]. Figure 2(a)-(c) presents an example of a net with spurious markings: the Parikh vector $\widehat{\sigma} = (2,1,0,0,1,0)$ and the marking $m = (0,0,1,1,0)$ are a solution to the marking equation, as shown in Fig. 2(c). However, m is not reachable by any feasible sequence. Figure 2(b) depicts the graph containing the reachable markings and the spurious markings (shadowed). The numbers inside the states represent the tokens at each place $(p_1, \ldots, p_5)$. This graph is called the *potential reachability graph*. The initial marking is represented by the state $(1,0,0,0,0)$. The marking $(0,0,1,1,0)$ is only reachable from the initial state by visiting a negative marking through the sequence $t_1 t_2 t_5 t_1$, as shown in Fig. 2(b). Therefore, equation (1) provides only a sufficient condition for reachability of a marking.

For well-structured Petri nets, e.g. when the net is *free-choice* [15], *live*, *bounded* and *reversible*, equation (1) together with a collection of sets of places (called *traps invariants*) of the system completely characterizes reachability [11]. For the rest of cases, the problem of the spurious solutions can be palliated by the

use of trap invariants [12], or by the addition of some special places named *cutting implicit places* [17] to the original Petri net that remove spurious solutions from the original marking equation.

3 Tackling Obstructed Workflows

To tackle an obstructed state in a workflow, we envisage a hybrid approach, depending on the existence of historical information. In case no historical information is provided, in Section 3.1 we propose a model-based exploration approach that suggests the minimal amount of resources to be added into the model to escape from an obstructed state. When historical information is provided in form of an event log, the method presented in Section 3.2 could be used, which simply detects the most similar historical successful trace.

3.1 Model-based Obstruction Solving

If only the model of a workflow with its authorizations and constraints are given, we intend to solve an obstructed state not by changing the model (cf. [3]), but by finding the best path with minimal violation. By flattening the workflow with its authorizations and users into a Petri net and encoding the obstruction with a corresponding marking, the marking equation shall be tweaked to provide a minimized Parikh vector to reach a completed marking, possibly violating given firing rules. The minimal Parikh vector shall be computed by solving the marking equation in the domain of Natural numbers, using Integer Linear Programming (ILP) techniques [7].

Flattening of Authorization Data into WF-net Given the example workflow in Fig. 1(a), we flatten the authorization and constraints into a WF-net step by step. This encoding may only show our intention and needs to be developed further in terms of WF-net soundness. The following steps shall give the intention of this flattening of authorization data into a WF-net and will be refined in more detail in future.

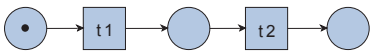
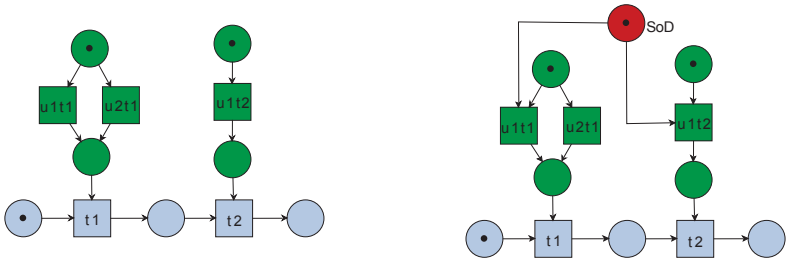


Fig. 3: Simplified Payment Workflow as WF-net with initial marking

First, because of the absence of ambiguous gateways in the model in Fig. 1(a), we can easily transform the workflow into the Petri net in Fig 3. To model access control, we assume the simple access control model without roles from Fig 1(b)). The user-task allocation is noted by the corresponding transitions (e.g. *u1t1* in Fig. 4(a)). Firing *u1t1* for instance represents the decision of who shall execute



(a) WF-net with access control (b) Fig. 4a with SoD

Fig. 4: Flattening access control and SoD into Simplified Payment Workflow

a specific task. In a further step, we model the SoD constraint in Fig. 4(b) by introducing a choice place for all users authorized for both tasks. In this way, we are able to model SoD constraints for sequential as well as concurrent tasks. A generalized way to model SoD constraints in WF-nets shall be provided in future.

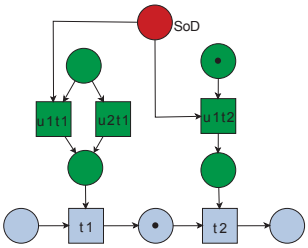


Fig. 5: Obstructed marking in flattened WF-net

The initial marking of the net is represented in Fig. 4(b). After $t1$ has been executed by $u1$, we are running in an obstructed state. This obstructed marking is represented in Fig. 5.

Tweaking the marking equation Given an obstruction marking m_{obs} and a final marking m_{end}, the ILP model below sketches our intended approach for using the marking equation:

ILP model for Completing an Obstruction State m_{obs}

$$\begin{array}{l}\text{Min cost}(X, \Delta) \text{ subject to:} \\ \quad m_{live} = m_{obs} + \Delta \\ \quad m_{end} = m_{live} + \mathbf{N} \cdot X \\ \quad X, \Delta \geq \mathbf{0} \quad X \in \mathbb{N}^{|T|} \quad \Delta \in \mathbb{N}^{|P|}\end{array}$$

After an obstructed marking m_{obs} has been reached, the idea would be to add the necessary tokens to the deadlocked model in order to take the current obstructed marking to a final state. The ILP model above has two sets of variables[3]: Δ is the addition of tokens to m_{obs} that takes to an unobstructed marking m_{live}, and X is the Parikh vector that will take from m_{live} to m_{end}. A solution to the ILP model will then jointly decide the necessary amount of tokens and the consequent firings to be made to reach m_{end}. Remarkably, the cost function is a minimization that considers both the length of the trace completing the workflow (through the Parikh vector X) and the amount of tokens needed to escape from the obstruction marking (the variables Δ), thus globally optimizing these two decisions. We consider the cost as a user-defined function, since perhaps different costs can be assigned depending on the context, e.g., if a shortest path is preferred independently of the violations performed then one can set cost 0 (or significantly less than X variables) to Δ variables. On the other hand, if the amount of violations should be reduced, the opposite cost can be set. Also, the cost for variables in the X vector may differ, e.g., if the firing of certain activities should be incentivated/avoided. The same holds for the Δ variables.

For instance, for the Petri net of Fig. 5, the ILP model above (assigning unitary costs to both X and Δ) will find the solution $\Delta = (0, 0, 0, 1, 0, 0, 0, 0)$, i.e., putting a token in the SoD place, and $X = (0, 1, 0, 0, 1)$, with X according to the order $t1$, $t2$, $u1t1$, $u2t1$, $u1t2$.

Clearly, the assignment on Δ and X variables defines the violations to make in order to complete the workflow. Assessing the impact and meaning of these violations for the authorization, constraints and users is a further challenge here, representing a next step in our research dealing with security in business processes.

3.2 Log-based Obstruction Work-Around

If there is historical information of the process, i.e. logs, our intention is to exploit this information and divide the cases of the log into successful and obstructed ones, based on the analysis of obstructions with the given model. Here, existing approaches analysing satisfiability and obstructions in workflows could be adapted to the extended WF-net incorporating authorization and SoD constraints.Given such partition of the logs, we intend to take an obstructed trace and find its nearest match to the successfully executed traces. The nearest match would then propose the partial sequence for the rest of execution to reach a completed state.

[3] m_{live} can be computed from m_{obs} and Δ.

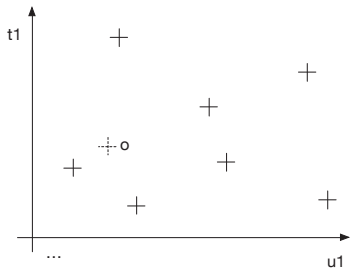


Fig. 6: Sketch of n-dimensional space indicating obstructed trace o

Fig. 6 sketches an n-dimensional space representing the traces of successful executions. Point o indicates the obstructed trace. k-nearest neighbour (kNN) algorithms could then find the closest successful path to reach a completed state. Here, also the implications from choosing the closest trace regarding security violations need to be investigated further.

4 Practical Applications

The presented approach provides a range of applications. It could be used to recommend who shall perform which tasks, for example in a Break-Glass situation, or as an assisted delegation, showing potential best delegates (with least violation) to the delegator. The core idea behind the approach however, is to enable automated delegation. Moreover, obstruction analysis techniques could also help policy-designer to improve their policies. We envisage to conduct a case study to further investigate and underline the practical applications of the approach.

5 Conclusion and Future

Our work is located in the tension between security controls on the one hand and maintaining flexibility in terms of process availability on the other. The intention here is to take a certain degree of violation into account to still succeed the workflow. As a next step, we aim to define more general solutions on how to flatten SoD and BoD constraints into WF-nets. Regarding the presented model-based approach there is the limitation that the Parikh vector does not propose a certain order of executing the transitions. In this regard, results need to be investigated further. Moreover, to better assess the violations taken into account, future work will also aim to use profiling techniques based on logs. Moreover, we want to implement the developed methods into the Security Workflow Analysis Toolkit (SWAT) [16] to provide more specific evidence on how a reliable solution in an organizational software system should be constituted.

References

1. Rafael Accorsi. Sicherheit im Prozessmanagement. *digma Zeitschrift für Datenrecht und Informationssicherheit*, 2013.
2. David A. Basin, Samuel J. Burri, and Günter Karjoth. Obstruction-free authorization enforcement: Aligning security with business objectives. In *CSF*, pages 99–113. IEEE Computer Society, 2011.
3. David A. Basin, Samuel J. Burri, and Günter Karjoth. Optimal workflow-aware authorizations. In Vijay Atluri, Jaideep Vaidya, Axel Kern, and Murat Kantarcioglu, editors, *SACMAT*, pages 93–102. ACM, 2012.
4. Matt Bishop. *Introduction to Computer Security*. Addison-Wesley Professional, 2004.
5. Reinhardt Botha and Jan Eloff. Separation of duties for access control enforcement in workflow environments. *IBM Systems Journal*, 40(3):666–682, March 2001.
6. Samuel J. Burri. *Modeling and enforcing workflow authorizations*. PhD thesis, ETH, Zürich, 2012.
7. Josep Carmona and Jordi Cortadella. Encoding large asynchronous controllers with ILP techniques. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(1):20–33, 2008.
8. Nicola Clark and David Jolly. Societe generale loses $7 billion in trading fraud, 2008.
9. Jason Crampton and Gregory Gutin. Constraint expressions and workflow satisfiability. In Mauro Conti, Jaideep Vaidya, and Andreas Schaad, editors, *SACMAT*, pages 73–84. ACM, 2013.
10. Jason Crampton and Charles Morisset. An auto-delegation mechanism for access control systems. In Jorge Cuéllar, Javier Lopez, Gilles Barthe, and Alexander Pretschner, editors, *STM*, volume 6710 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2010.
11. J. Desel and J. Esparza. Reachability in cyclic extended free-choice systems. *TCS 114, Elsevier Science Publishers B.V.*, 1993.
12. J. Esparza and S. Melzer. Verification of safety properties using integer programming: Beyond the state equation. *Formal Methods in System Design*, (16):159–189, 2000.
13. Julius Holderer, Rafael Accorsi, and Günter Müller. When four-eyes become too much: a survey on the interplay of authorization constraints and workflow resilience. In Roger L. Wainwright, Juan Manuel Corchado, Alessio Bechini, and Jiman Hong, editors, *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, pages 1245–1248. ACM, 2015.
14. Maria Leitner and Stefanie Rinderle-Ma. A systematic review on security in process-aware information systems - constitution, challenges, and future directions. *Information & Software Technology*, 56(3):273–293, 2014.
15. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–574, April 1989.
16. Rafael Accorsi, Julius Holderer, Thomas Stocker, and Richard M. Zahoransky. Security workflow analysis toolkit. In Stefan Katzenbeisser, Volkmar Lotz, and Edgar R. Weippl, editors, *Sicherheit 2014: Sicherheit, Schutz und Zuverlässigkeit, Beiträge der 7. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI), 19.-21. März 2014, Wien, Österreich*, volume 228 of *LNI*, pages 433–442. GI, 2014.

17. M. Silva, E. Teruel, and J. M. Colom. Linear algebraic and linear programming techniques for the analysis of place/transition net systems. In Reisig, W. and Rozenberg, G., editors, *Lecture Notes in Computer Science: Lectures on Petri Nets I: Basic Models*, volume 1491, pages 309–373. Springer-Verlag, 1998.
18. R. L. Trope and E. K. Ressler. Mettle fatigue: Vw's single-point-of-failure ethics. *IEEE Security Privacy*, 14(1):12–30, Jan 2016.
19. Wil M. P. van der Aalst. The application of Petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.
20. Qihua Wang and Ninghui Li. Satisfiability and resiliency in workflow authorization systems. *ACM Trans. Inf. Syst. Secur.*, 13(4):40:1–40:35, December 2010.
21. Qihua Wang, Ninghui Li, and Hong Chen. On the security of delegation in access control systems. In Sushil Jajodia and Javier López, editors, *ESORICS*, volume 5283 of *Lecture Notes in Computer Science*, pages 317–332. Springer, 2008.
22. Christian Wolter, Michael Menzel, and Christoph Meinel. Modelling security goals in business processes. In *Modellierung*, volume 127, pages 201–216, 2008.