Chapter 8

FORENSIC LEAK DETECTION FOR BUSINESS PROCESS MODELS

Rafael Accorsi and Claus Wonnemann

Abstract This paper presents a formal forensic technique based on information flow analysis to detect data and information leaks in business process models. The approach can be uniformly applied to the analysis of process specifications and the log files generated during process execution. The Petri net dialect IFnet is used to provide a common basis for the formalization of isolation properties, the representation of business process specifications and their analysis. The utility of the approach is illustrated using an eHealth case study.

Keywords: Business process forensics, leak detection, information flow analysis

1. Introduction

Up to 70% of business processes, including customer relationships and supply chains, operate in a fully automated manner. The widespread adoption of business processes for automating enterprise operations has created a substantial need to obtain business layer evidence.

Forensic tools for enterprise environments primarily focus on the application layer (servers and browsers) and the technical layer (virtual machines and operating systems). However, tools for business process forensics are practically non-existent. Thus, forensic examiners must manually gather evidence about whether or not a process exhibits data and information leaks [9]. Specifically, they must demonstrate the presence or absence of harmful data flows across different enterprise domains or information flows via "covert channels."

Leaks occur as a result of data flows or information flows. Data flows are direct accesses of data over legitimate channels that violate access control policies. Information flows are indirect accesses of information

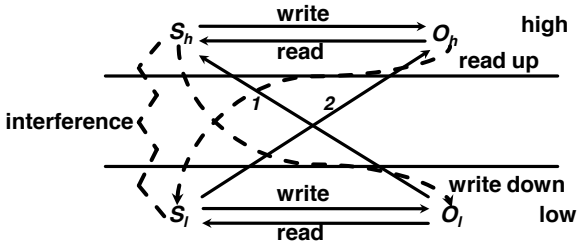


Figure 1. Security levels and leaks.

over covert channels that allow attackers to derive confidential information.

A forensic examiner typically seeks to detect leaks that arise from the interaction of different subjects with a business process. The abstract system model considers subjects (s) separated into two security classes with regard to a particular object (o): high subjects (s_h) that are able to access o and low subjects (s_l) that cannot access o. Subjects in both classes may interact with the business process. An information leak occurs when a low subject obtains information that is intended to be visible only to high subjects.

Figure 1 illustrates the model. It integrates mandatory access control [18] with information flow control [10]. The solid arrows denote legitimate data flows. Subjects may read and write to objects within a security level. Additionally, high subjects (s_h) may read lower level objects (o_l) and low subjects (s_l) may write high objects (o_h). Forbidden data flows, which are represented using dashed arrows, occur when high subjects write low objects (write down) or low subjects read high objects (read up). An interference occurs when low subjects may, through observations and knowledge about the operation of the business process, combine information to derive high information.

This paper presents a formal forensic technique that serves as a uniform basis for leak detection in business process specifications and log files. Figure 2 outlines the overall technique. The technique employs a Petri net based meta-model (IFnet) as a formal model for leak detection analysis in business processes. Business process specifications can be automatically translated to IFnet models [3]. Alternatively, process reconstruction techniques [22] may be applied to mine IFnet models from business process logs. Because business processes are represented as IFnet models and isolation properties as Petri net patterns, information flow analysis and evidence generation can be based on an approach developed for Petri nets [7].

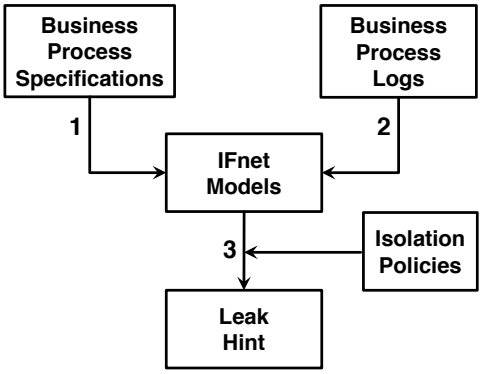


Figure 2. Forensic analysis of business processes.

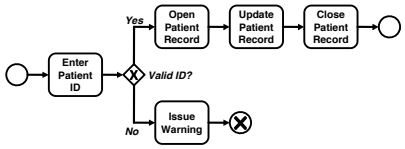
The Petri net based meta-model offers three advantages. First, it provides a uniform modeling formalism for a plethora of business process specification languages (e.g., BPMN, BPEL and EPC). Second, it allows the well-founded formalization of structural isolation properties as Petri net patterns [7]. Third, it provides a sound basis for efficient isolation analysis, which reduces to determining whether or not a Petri net encompasses a leak pattern [11]. Moreover, the graphical notation and similarity with business process specifications renders the approach practical for enterprise forensics.

2. Related Work

The enterprise meta-model can be used to classify previous work on business process forensics. The model has three layers. The business layer contains business process specifications and business objects. The application layer provides for data objects and services. The technical layer contains software and hardware for service operations.

Forensic techniques mainly focus on the application and technical layers. Application layer techniques fall in the domain of network forensics [17], primarily focusing on the choreography and use of distributed web services [15]. Gunestas, *et al.* [12] have introduced "forensic web services" that can securely maintain transaction records between web services. Chandrasekaran, *et al.* [8] have developed techniques for inferring sources of data leaks in document management systems.

Forensic techniques for database systems attempt to identify leaks in query answering systems [6], detect tampering attempts [16] and measure retention [19]. Meanwhile, new issues are arising due to increased virtualization [5] and the need for live evidence acquisition [13].




```
<sequence name="main">
  <empty name="Enter_Patient_ID"/>
  <switch name="Switch_1">
    <case>
      <sequence>
        <empty name="Open_Patient_Record"/>
        <empty name="Update_Patient_Record"/>
        <empty name="Close_Patient_Record"/>
      </sequence>
    </case>
    <otherwise>
      <empty name="Issue_Warning"/>
    </otherwise>
  </switch>
</sequence>
```

Figure 3. Update Patient Record process model and BPEL specification.

With regard to the business layer, techniques have been developed to analyze the structural properties of business processes [1, 23], but little, if any, work has focused on obtaining evidence on isolation properties. Accorsi and Wonnemann [2] have described a forensic approach for log analysis based on propagation graphs that model how data items spread in a system. Sun, *et al.* [20] have developed a technique for reasoning about data flows and business processes, but they do not relate it to security. Trcka, *et al.* [21] have specified anti-patterns (using Petri nets) that express data flow flaws, but they are neither related to isolation nor security. Atluri, *et al.* [4] have provided a model for analyzing Chinese wall policies, but it does not address leak detection.

Covert channels are not intended to transfer information, but are often misused to this end [14]. However, while covert channels are very relevant to isolation properties, they do not fall in the scope of forensic analysis of business processes. The approach proposed in this paper stands out in that it provides a powerful and automated means for analyzing both data and information flows in business processes.

3. Business Process Leak Detection

This section describes the forensic leak detection approach and illustrates its application using an eHealth database example.

Figure 3 presents the Update Patient Record business process for an eHealth company along with the corresponding BPEL code. Although it is very simple, updating a patient record is a central fragment that recurs in several hospital information systems (e.g., accounting, medical treatment and billing). The business process involves five activities (boxes) and an exclusive choice (x-diamond), which denotes an if-statement. The goal of forensic analysis is to detect whether or not the business process leaks information and, if it does, to determine the specific channels over which this occurs.

3.1 IFnet and Translations

IFnet is an extension of colored Petri nets tailored to the specification and analysis of isolation properties in business processes. Colored Petri nets generalize standard Petri nets by supporting distinguishable tokens. Following the standard terminology, tokens are distinguished by their color, which is an identifier from the universe $\mathcal{C}$ of token colors.

A colored Petri net is a tuple $N = (P, T, F, C, I, O)$ where P is a finite set of places, T is a finite set of transitions such that $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs called the flow relation. Given $x, y \in (P \cup T)$, xFy denotes that there is an arc from x to y. The functions C, I and O define the capacity of places and the input and output of transitions, respectively:

- The "capacity function" $C : P \to \mathbb{N}$ is the number of tokens a place can hold at a time.
- The "input function" $I : T \times P \times \mathcal{C} \to \mathbb{N}$ is the number of tokens expected for each transition t, each place i with iFt, and each token color c.
- The "output function" $O : T \times P \times \mathcal{C} \to \mathbb{N}$ is the number of produced tokens for each transition t, each place o with tFo, and each token color c.

A place contains zero or more tokens. The marking (or state) is the distribution of tokens over places. A marking M is a bag over the Cartesian product of the set of places and the set of token colors. M is a function from $P \times \mathcal{C}$ to the natural numbers, i.e., $M : P \times \mathcal{C} \to \mathbb{N}$. A partial ordering is defined to compare states with regard to the number of tokens in places. For any two states M_1 and M_2, $M_1 \leq M_2$ if for all $p \in P$ and for all $c \in \mathcal{C}$: $M_1(p, c) \leq M_2(p, c)$. The sum of two bags $(M_1 + M_2)$, the difference $(M_1 - M_2)$ and the presence of an element in a bag $(a \in M_1)$ are defined in a straightforward way. A marked colored Petri net is a pair (N, M) where $N = (P, T, F, C, I, O)$ is a colored Petri net and M is a bag over $P \times \mathcal{C}$ that denotes the marking of the net.

Elements of $P \cup T$ are nodes. A node x is an input node of another node y if there is a directed arc from x to y (i.e., xFy). Node x is an output node of y if yFx. For any $x \in P \cup T$, $\overset{N}{\bullet} x = \{y \mid yFx\}$ and $x \overset{N}{\bullet} = \{y \mid xFy\}$. Note that the superscript N is omitted when it is clear from the context.

The number of tokens may change during the execution of a net. Transitions are the active components in a colored Petri net. They change the state of the net according to the following firing rule:

- A transition $t \in T$ is enabled in state M_1 if each input place contains sufficiently many tokens of each color and each output place has sufficient capacity to contain the output tokens:

$$\forall i \in {\bullet t}, \forall c \in \mathcal{C} : I(t,i,c) \leq M_1(i,c) \tag{1}$$

$$\forall o \in t{\bullet} : \sum_{c \in \mathcal{C}} O(t,o,c) + \sum_{c \in \mathcal{C}} M_1(o,c) \leq C(o) \tag{2}$$

- Once enabled, a transition t may fire and consume the designated number of tokens from each of its input places and produce the designated number of tokens for each of its output places. Firing of transition t in state M_1 results in a state M_2 defined as:

$$\forall i \in {\bullet t}, \forall c \in \mathcal{C} : M_2(i,c) = M_1(i,c) - I(t,i,c) \tag{3}$$

$$\forall o \in t{\bullet}, \forall c \in \mathcal{C} : M_2(o,c) = M_1(o,c) + O(t,o,c) \tag{4}$$

$$\forall p \in P \setminus ({\bullet t} + t{\bullet}), \forall c \in \mathcal{C} : M_2(p,c) = M_1(p,c) \tag{5}$$

Given a colored Petri net N and a state M_1, we define:

- $M_1 \xrightarrow{t} M_2$: Transition t is enabled in M_1 and firing t in M_1 results in state M_2.
- $M_1 \longrightarrow M_2$: Transition t exists such that $M_1 \xrightarrow{t} M_2$.
- $M_1 \xrightarrow{\sigma} M_n$: Firing sequence $\sigma = t_1 t_2 t_3 ... t_{n-1}$ from state M_1 leads to state M_n via a (possibly empty) set of intermediate states $M_2, ..., M_{n-1}$, i.e. $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} ... \xrightarrow{t_{n-1}} M_n$.

M_n is reachable from M_1 (i.e., $M_1 \xrightarrow{*} M_n$) if a firing sequence σ exists such that $M_1 \xrightarrow{\sigma} M_n$. The set of states reachable from state M_1 is denoted by $[M_1]$.

3.2 IFnet Extension

IFnet extends the colored Petri net formalism by adding constructs required for business process modeling and information flow analysis. An IFnet models business process activities through transitions and data items (including documents, messages and variables) through tokens. Tokens with color `black` have a special status: they do not stand for data items, but indicate the triggering and termination of activities. The set of colored tokens that are not `black` is denoted by $\mathcal{C}_{\mathtt{c}}$.

Formally, an IFnet is a tuple $N = ((P, T, F, C, I, O), S_{\mathcal{U}}, A, G, L_{\mathcal{SC}})$ where (P, T, F, C, I, O) is a colored Petri net and:

- Function $S_{\mathcal{U}} : T \rightarrow \mathcal{U}$ assigns transitions to subjects from a set $\mathcal{U}$. A subject is the acting entity on whose behalf a corresponding business process activity is performed.
- Function $A : T \times \mathcal{C}_{\mathsf{c}} \rightarrow \{read, write\}$ defines if a transition t reads or writes an input datum $i \in \bullet t$.
- Function $G : T \rightarrow \mathsf{P}_{\mathcal{C}}$ assigns predicates (guards) to transitions where $\mathsf{P}_{\mathcal{C}}$ denotes the set of predicates over colored tokens. A predicate evaluates to either *true* or *false* and is denoted by, e.g., $\mathtt{p}(\mathtt{red}, \mathtt{green})$ where $\mathtt{p}$ is the name of the predicate and the identifiers in parentheses indicate the tokens needed for its evaluation. For an enabled transition to fire, its guard must evaluate to *true*.
- Function $L_{\mathcal{SC}} : T \cup \mathcal{C}_{\mathsf{c}} \rightarrow \mathcal{SC}$ assigns security labels to transitions and colored tokens. $\mathcal{SC}$ is a finite set of security labels that forms a lattice under the relation $\prec$. Every set $\mathcal{SC}$ contains an additional element `unlabeled`, which denotes that a transition or token does not hold a label.

An IFnet must meet five structural conditions [3]. The first two conditions ensure that a business process has defined start and end points. The third condition prevents "dangling" transitions or activities that do not contribute to the business process. The fourth condition requires transitions to signal their triggering and termination via `black` tokens. The fifth condition ensures that data items are passed through the business process according to the transitions.

The translation from a BPMN or BPEL business process specification to an IFnet model occurs automatically. It involves two steps. In the first step, the structure of the process is translated to an IFnet model. In the second step, the net is labeled for analysis: activities, places and resources are annotated with security labels (high and low). The first step runs in a fully automated manner. The second uses an unfolding strategy to automatically derive labels.

The labeling strategy involves the unfolding of the IFnet model to investigate the interaction between two subjects in the business process. Formally, for a marked Petri net $(N, M) = ((P, T, F), M)$ and a resource relation $\mathcal{D}$, the corresponding IFnet is a tuple $(P_L \cup P_H \cup P_{\mathcal{D}}, T_L \cup T_H, F_L \cup F_H \cup F_{\mathcal{D}}, M_L + M_H + M_{\mathcal{D}})$ where:

- $((P_L, T_L, F_L), M_L)$ corresponds to the net $((P, T, F), M)$.
- $((P_H, T_H, F_H), M_H)$ is an equivalent net to $((P_L, T_L, F_L), M_L)$ with its elements renamed for distinction. The function $\Upsilon_{T_L \rightarrow T_H} : T_L \longrightarrow T_H$ maps transitions from T_L to their counterparts in T_H.

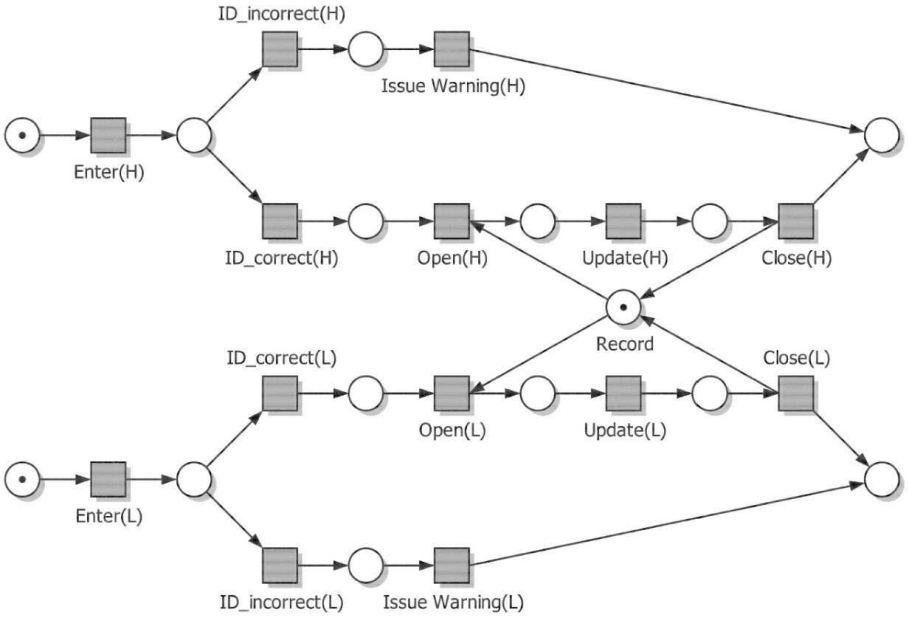


Figure 4. IFnet of the Update Patient Record business process.

- $P_{\mathcal{D}}$ is a set of places that model the blocking of resources. There exists exactly one $p \in P_{\mathcal{D}}$ for each pair $(t_0, t_1) \in \mathcal{D}$. The function $\Upsilon_{P_{\mathcal{D}} \to \mathcal{D}} : P_{\mathcal{D}} \longrightarrow \mathcal{D}$ maps places from $P_{\mathcal{D}}$ to the corresponding pairs of transitions in T (and thus in T_L).

- $M_{\mathcal{D}}$ denotes the initial marking of places $P_{\mathcal{D}}$. $M_{\mathcal{D}}$ marks each $p \in P_{\mathcal{D}}$ with exactly one token.

- $F_{\mathcal{D}}$ denotes the arcs that connect places in P to blocking and releasing transitions in T_L and T_H. For each $p \in P_{\mathcal{D}}$ with $\Upsilon_{P_{\mathcal{D}} \to \mathcal{D}}(p) = (t_0, t_1)$, $F_{\mathcal{D}}$ contains the following arcs:

  - (p, t_0) denotes the blocking of resource p through transition t_0.
  - (t_1, p) denotes the release of resource p by transition t_1.
  - $(p, \Upsilon_{T_L \to T_H}(t_0))$ denotes the blocking of resource p through a transition in T_H that corresponds to t_0.
  - $(\Upsilon_{T_L \to T_H}(t_1), p)$ denotes the corresponding release of resource p.

This strategy, and others that obtain labels from access control lists and role-based access control policies, have been automated. Figure 4

presents the IFnet for the business process in Figure 3. The record is a resource shared by the high subject (upper part of the net) and the low subject (lower part). The resulting IFnet is the subject of analysis.

3.3 Isolation Policies

Isolation policies formalize confidentiality properties. They are safety properties that denote leaks that should not occur between high and low subjects. Our approach captures these properties using IFnet patterns. In the following, we demonstrate patterns that capture data flow and information flow violations. These patterns stand for extensional policies, i.e., policies that capture leaks independently of the actual business process at hand. This allows the compilation of a library of patterns from which patterns could be selected depending on the purpose of the investigation. Intensional policies capture properties specific to a business process and are not suitable for thorough isolation analysis.

Data flow patterns capture the direct data leaks that occur in two situations. The first is when a high subject writes to a low object. The second is when a low subject reads a high object. For example, the IFnet pattern in Figure 5(a) formalizes the "write down" rule: resource a (grey token) is written by high and then read by low. A leak occurs when this is reachable in an IFnet.

The patterns capture access control policies over a lattice-based information flow model. This approach can express a number of requirements, including the Bell-LaPadula and Chinese wall models, as well as binding and separation of duties.

Information flow patterns capture interferences between the activities of high and low subjects. Each interference allows low subjects to derive information about high objects. Formally, patterns capture well-founded bisimulation-based information flow properties specified in terms of a process algebra.

Busi and Gorrieri [7] have demonstrated the correspondence of patterns. For example, the patterns in Figures 5(b) and 5(c) capture the bisimulation-based property of non-deducibility, which prohibits a low subject from deriving an aspect of a high object. The place s in Figure 5(b) is a "causal place" – whenever high fires an activity, low is able to observe it. The place s in Figure 5(c) is a "conflict place" – both high and low compete for the control flow token in s. If high obtains the token, then low can derive that high has performed the corresponding activity. Other patterns capture additional bisimulation-based properties [3].

It is important to note that the non-interference properties capture possibilistic information leaks, which are business process vulnerabilities

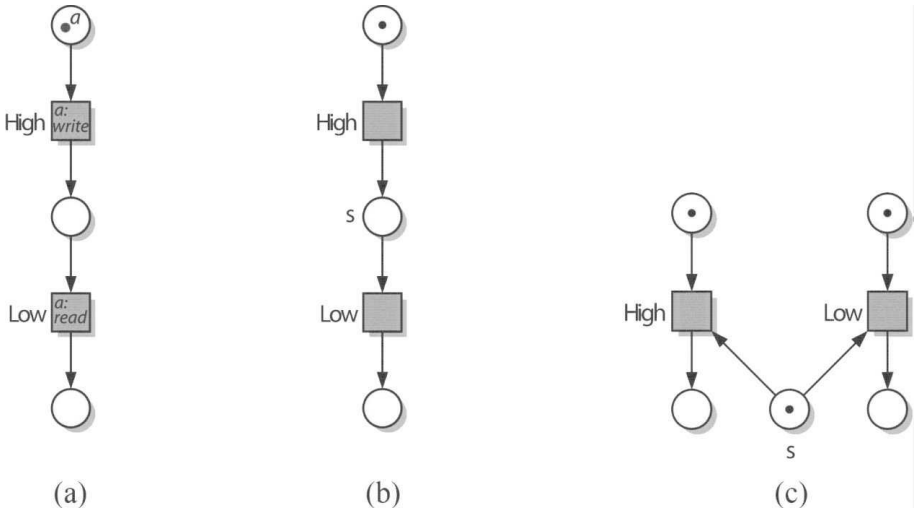


Figure 5. Isolation properties as IFnet patterns.

that allow for the derivation of information. Hence, they are weaker than data leaks, which indicate concrete illegal data flows.

3.4 Leak Detection Analysis

Based on the business process and policies formalized in the IFnet meta-model, the leak detection algorithm checks whether or not the policy patterns are reachable. This section describes the verification procedure and demonstrates that the eHealth example exhibits information and data leaks.

The verification procedure involves two steps. The first step checks if the business process model exhibits harmful causal and/or conflict places. If this is true, the second step determines if the detected places are reachable during an execution of the net. The first step involves a static check of the net, but the second step is dynamic and requires the analysis of the entire marking graph generated by the business process model.

A decision procedure is employed to perform the dynamic check when an IFnet exhibits a causal place. First, the marking graph is generated. For each marking in the list, a reachable marking is computed for every enabled transition and the corresponding pair is added to the current marking. When a new marking is found, it is added to the list for examination.

Given the marking graph, the procedure for detecting a casual place traverses the marking graph attempting to find markings in which the

potential causal place s reached by a high transition subsequently leads to a low transition. If these conditions are met by s, then the place is an active causal place. The procedure for detecting active conflict places operates in a similar manner.

3.5 Example

The IFnet in Figure 4 exhibits both data and information leaks. A data leak that violates the policy in Figure 5(a) occurs when a high subject updates a patient record (first execution of the business process) and a low subject opens the record. In this case, high has "written down" and leaked data to low. The information leak occurs at the place labeled Record.

Upon the firing of transition Open(H), the patient record is removed from the storage place and transition Open(L) is blocked until the token is returned. In this case, the high part of the net influences the low part because it prevents the transition from firing. There is an information flow (through a resource exhaustion channel) that allows the low part to deduce that high currently holds the patient record.

Upon the firing of transition Close(H), the token representing the patient record is returned to its storage place and might be consumed by transition Open(L). Hence, opening the patient record on the low side requires its preceding return on the high side. This causality reveals to low the fact that high has returned the record.

Other derivations with regard to the time and duration of update are possible, but their semantics depend on the purpose of evidence generation.

4. Conclusions

The proposed approach for the forensic analysis of business processes is based on IFnet, a meta-model tailored to the analysis of data and information leaks. The eHealth case study demonstrates the utility of the approach. While the focus has been on generating evidence for existing business process specifications, the approach is applicable to the analysis of log files generated by business process executions. For this purpose, we are investigating process reconstruction algorithms for mining IFnet models. In particular, we are extending the algorithms to produce a series of different models (as opposed to one model) of a process, which could help cope with "multitenancy" in cloud and grid environments. Also, by considering runtime information, it may be possible to analyze other isolation properties based on execution dynamics.

References

[1] R. Accorsi and L. Lowis, ComCert: Automated certification of cloud-based business processes, *ERCIM News*, vol. 83, pp. 50–51, 2010.

[2] R. Accorsi and C. Wonnemann, Auditing workflow executions against dataflow policies, *Proceedings of the Thirteenth International Conference on Business Information Systems*, pp. 207–217, 2010.

[3] R. Accorsi and C. Wonnemann, InDico: Information flow analysis of business processes for confidentiality requirements, *Proceedings of the Sixth ERCIM Workshop on Security and Trust Management*, 2010.

[4] V. Atluri, S. Chun and P. Mazzoleni, A Chinese wall security model for decentralized workflow systems, *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 48–57, 2001.

[5] D. Bem, Virtual machine for computer forensics – The open source perspective, in *Open Source Software for Digital Forensics*, E. Huebner and S. Zanero (Eds.), Springer, New York, pp. 25–42, 2010.

[6] S. Bottcher and R. Steinmetz, Finding the leak: A privacy audit system for sensitive XML databases, *Proceedings of the Twenty-Second International Conference on Data Engineering Workshops*, pp. 100–110, 2006.

[7] N. Busi and R. Gorrieri, Structural non-interference in elementary and trace nets, *Mathematical Structures in Computer Science*, vol 19(6), pp. 1065–1090, 2009.

[8] M. Chandrasekaran, V. Sankaranarayanan and S. Upadhyaya, Inferring sources of leaks in document management systems, in *Advances in Digital Forensics IV*, I. Ray and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 291–306, 2008.

[9] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka and J. Molina, Controlling data in the cloud: Outsourcing computation without outsourcing control, *Proceedings of the ACM Workshop on Cloud Computing Security*, pp. 85–90, 2009.

[10] D. Denning, A lattice model of secure information flow, *Communications of the ACM*, vol. 19(5), pp. 236–243, 1976.

[11] S. Frau, R. Gorrieri and C. Ferigato, Petri net security checker: Structural non-interference at work, *Proceedings of the Fifth International Workshop on Formal Aspects in Security and Trust*, Springer-Verlag, Berlin, Germany, pp. 210–225, 2008.

[12] M. Gunestas, D. Wijesekera and A. Singhal, Forensic web services, in *Advances in Digital Forensics IV*, I. Ray and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 163–176, 2008.

[13] B. Hay, M. Bishop and K. Nance, Live analysis: Progress and challenges, *IEEE Security and Privacy*, vol. 7(2), pp. 30–37, 2009.

[14] B. Lampson, A note on the confinement problem, *Communications of the ACM*, vol 16(10), pp. 613–615, 1973.

[15] L. Lowis and R. Accorsi, Vulnerability analysis in SOA-based business processes, to appear in *IEEE Transactions on Service Computing*, 2011.

[16] K. Pavlou and R. Snodgrass, Forensic analysis of database tampering, *ACM Transactions on Database Systems*, vol. 33(4), pp. 30:1–30:47, 2008.

[17] M. Ponec, P. Giura, H. Bronnimann and J. Wein, Highly efficient techniques for network forensics, *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 150–160, 2007.

[18] R. Sandhu and P. Samarati, Authentication, access control and audit, *ACM Computing Surveys*, vol. 28(1), pp. 241–243, 1996.

[19] P. Stahlberg, G. Miklau and B. Levine, Threats to privacy in the forensic analysis of database systems, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 91–102, 2007.

[20] S. Sun, J. Zhao, J. Nunamaker and O. Sheng, Formulating the data-flow perspective for business process management, *Information Systems Research*, vol. 17(4), pp. 374–391, 2006.

[21] N. Trcka, W. van der Aalst and N. Sidorova, Data-flow anti-patterns: Discovering data-flow errors in workflows, *Proceedings of the Twenty-First International Conference on Advanced Information Systems Engineering*, pp. 425–439, 2009.

[22] W. van der Aalst, B. van Dongen, J. Herbst, L. Maruster, G. Schimm and A. Weijters, Workflow mining: A survey of issues and approaches, *Data and Knowledge Engineering*, vol. 47(2), pp. 237–267, 2003.

[23] 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*, vol. 43(3), pp. 90–93, 2010.