Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Using information flow analysis to detect implicit information leaks for web service composition*

Jia-xin JIANG‡[1], Zhi-qiu HUANG[1,2], Wei-wei MA[1], Yan CAO[1]

[1]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China*

[2]*Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210000, China*

E-mail: jiangjiaxin@nuaa.edu.cn; zqhuang@nuaa.edu.cn; maweiwei@nuaa.edu.cn; caoyan926@nuaa.edu.cn

Received June 24, 2016; Revision accepted Jan. 11, 2017; Crosschecked Apr. 4, 2018

Abstract: Information leak, which can undermine the compliance of web-service-composition business processes for some policies, is one of the major concerns in web service composition. We present an automated and effective approach for the detection of implicit information leaks in business process execution language (BPEL) based on information flow analysis. We introduce an adequate meta-model for BPEL representation based on a Petri net for transformation and analysis. Building on the concept of Petri net place-based noninterference, the core contribution of this paper is the application of a Petri net reachability graph to estimate Petri net interference and thereby to detect implicit information leaks in web service composition. In addition, a case study illustrates the application of the approach on a concrete workflow in BPEL notation.

Key words: Information flow analysis; Business process execution language; Petri net; Interference
https://doi.org/10.1631/FITEE.1601371 **CLC number:** TP311

1 Introduction

A web service (WS) is a modular and self-described application that uses standard web technologies to interact with other services (Benatallah et al., 2002). With the rapid development of web service technologies, the number of web services published on the Internet is growing. Yet, a single web service can have difficulty in meeting a user's multiple function requests (Singh, 2001). So, it is necessary to combine individual web services into a set to obtain a more complex service, namely, a web service composition (Papazoglou, 2012).

Users are able to enjoy the convenience of web service composition; however, they usually have to submit some personal information, which is sensitive information, to service providers to carry out necessary business processes. In the process of composing, service providers may expose some of the users' sensitive information to other collaborators. On the one hand, information leak problems can be due to service providers misusing the sensitive information. In the combining process, the member services achieve their business functions by sending and receiving messages. The message interactions frequently need to use sensitive information and the more frequently services use the information, the more likely they are to misuse it (Massacci et al., 2006). On the other hand, information leak problems can be due to some dishonest services. They may use sensitive data to analyze a user's purchase preferences or promote their products to customers via telephone or e-mail. An even more serious problem is where some service

‡ Corresponding author

* Project supported by the National High-Tech R&D Program (863) of China (No. 2015AA015303) and the National Natural Science Foundation of China (No. 61272083)

ORCID: Jia-xin JIANG, http://orcid.org/0000-0001-6185-4438

providers sell sensitive information to the third parties without the user's permission (Hui et al., 2006).

In the face of these problems with information leaks, users have become increasingly worried about their individual, sensitive information security (Kagal et al., 2004), and want to be able to control the release of the information (Yee, 2007). In a nutshell, information security is the right of an entity to determine when, how, and to what extent it will release some sensitive information (Tbahriti et al., 2014). Because the nature of the service-oriented computing (SOC) environment is open, autonomous, and dynamic, when sensitive data is released, the user actually has a very difficult time controlling how service providers use the information (Carminati et al., 2005). Users hope not only to complete their business processes via web service compositions, but also to avoid suffering from the hazards of information leaks or at least to minimize the hazards.

Information leaks include explicit information leaks and implicit information leaks. An explicit information leak (i.e., a data leak) is the direct but illegal access to a data object. An implicit information leak concerns that unauthorized subjects can infer secret information. Although certification standards (e.g., ISO/IEC 27001) demand the identification of both kinds of leaks, the current state-of-the-art mechanisms detect mainly explicit information leaks (Atluri et al., 2001; Barkaoui et al., 2008; Armando and Ranise, 2011).

Access control mechanisms ensure that all objects that have direct access to the system are permitted. By controlling the reading, writing, updating, and deleting of information, they can avoid accidental or intentional damage, to realize protections that are safe and effective. Yet, once sensitive information is allowed to be accessed, an access control mechanism cannot control how the information is used; i.e., access control mechanisms cannot control the spread of information. Furthermore, the technical issue here is that access control mechanisms only monitor a particular transmission channel between a classified subject and a public subject, whereas so-called covert channels are neglected, i.e., channels that are not meant to transfer information but are misused to do so (Lampson, 1973).

Information flow (IF) control is based on a semantic specification of security requirements (Myers and Liskov, 1997). Compared with traditional access control policies, IF policies are stronger because they capture information propagation throughout the system (end-to-end) rather than mere data access. Thus, they can capture security violations that lie beyond the scope of an access control mechanism, such as the information leaks that take place over covert channels.

2 Related work

Business process execution language (BPEL) security focuses on the enforcement of confidentiality, integrity, and availability requirements in web service composition business processes (Atluri, 2001). The information flow analysis focuses on confidentiality, duality, and integrity (Sabelfeld and Myers, 2003). The vast majority of the proposed techniques in BPEL security are based on Petri nets. This is due to the extensive use of Petri net models to reason through BPEL and the availability of mature tool support. Lohmann et al. (2006) proposed the BPEL2oWFN tool to realize the transformation from BPEL to a Petri net. Tan et al. (2009) specified the data mapping from BPEL to a Petri net.

The state-of-the-art information flow analysis method for BPEL contains two aspects: (1) the detection of explicit information flows, such as data flows, in terms of both discretionary access control (DAC) and mandatory access control (MAC) based on multi-level security; (2) the detection of implicit flows over covert channels, such as information flow interferences.

1. DAC

Discretionary access control policy allows the owner of information to specify access to the information in accordance with the owner's wishes. This means the owner of the information can share their own resources at their discretion with other users who have a different authority in the system. Shafiq et al. (2005) presented a colored Petri net framework to verify the consistency of role-based access control policies. Sun et al. (2008) used this framework to manage service-based business process authorization. Lu et al. (2009) used colored Petri nets to model and analyze a workflow with separation of duty constraints in a collaborative business process. With the context of the DAC model, the aforementioned analysis technologies merely cover point-to-point security guarantees.

2. MAC

In a mandatory access control scheme, each subject or object in the system is endowed with its corresponding security classes, which can be changed only by management (such as the security administrator) according to strict rules. Existing approaches for MAC-based data flow analysis are based on a multilevel security (MLS) model (Bell, 1983), in particular Denning (1976)'s model and Bell and LaPadula (1973)'s model. Juszczyszyn (2003) used the colored Petri net (CPN) for the multi-level security (MLS) specification of policies and a CPN-tool to realize the verification as a reachability problem. Röhrig and Knorr (2004) defined task-based access control as a dynamic business process and then specified it with a Petri net. While these technologies employ MLS to capture information flows, they do not detect interference.

3. Information flow analysis

The information flow analysis of web service business processes focusing on interference is a novel research field (Accorsi and Wonnemann, 2009). This approach is inspired by the structural noninterference with Petri nets put forward by Busi and Gorrieri (2009). Accorsi et al. (2015) presented an approach to reform the labeled Petri nets to judge whether interference occurs in a business process model and to add additional places and transitions to detect small probabilities of information leakage in the system. Our approach is to use a Petri reachability graph to estimate the place-based noninterference of a Petri net, thereby detecting the small probability implicit information leakage.

3 Preliminaries of information flow analysis

This section introduces the preliminaries of information flow analysis including the principle of information flow based on a lattice model and noninterference, which is one of the information flow properties.

3.1 Lattice model

Information flow analysis is a formal approach to guarantee the absence of information leaks between designated subjects in a system (Accorsi and Wonnemann, 2010). For this reason, the subjects that interact with the system are given security labels to indicate their level of trust accorded by the system. Denning (1976) proposed a lattice model to describe the channel and information flow strategy.

Definition 1 (Lattice) If a tuple $(L, \leq)$ is a partially ordered set, and each couple of elements (like x and y) in L has the minimum upper bound and the maximum lower bound, then the tuple $(L, \leq)$ is a lattice.

In the real world, the number of subjects and objects is limited, so the actual system considers only a finite lattice.

Definition 2 (Finite lattice) A lattice $(L, \leq)$ with the number of elements in L being limited is called a finite lattice.

Information flow strategy is a series of rules about the flow of information that can be allowed within the system, and the constraint relations of the information flow that can be described by the lattice. We divide the level of information security according to the sensitivity of the information, and a security class (SC) consists of all information for each level. The information flow strategy can be described by a finite lattice (SC, $\leq$). The flow of information must follow certain strategies, so as to ensure the legitimacy and security of the information flow. Fig. 1 shows two security classes: sensitive (H) and insensitive (L). According to the security strategy, the legal information flows in the lattice are H to H, L to L, and L to H, but the flow H to L is an illegal information flow.

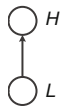

Fig. 1 A simple information flow lattice model

3.2 Noninterference

In IF analysis, the criterion for the security of a system is based on the notion of noninterference. Goguen and Meseguer (1982) introduced noninterference to formalize when a sensitive input to a system with multiple users was protected from the untrusted users of that system.

Noninterference implies that a high-level input has no effect on a low-level output; i.e., if the inputs differ only in high-level inputs, then the system will provide the same low-level outputs. Correspondingly, interference means that a high-level input has an

effect on a low-level output. Tschantz et al. (2015) gave us the formal proof of a connection between IF analysis and causality, and the proof that interference was an effect from a high-level input to a low-level output.

Fig. 2 denotes the information flow interference between a high domain and a low domain. A direct information leak happens if a high subject S_{h} writes classified information into a low object O_{l} (called a 'write down') or if a low subject S_{l} reads classified information from a high object O_{h} (called a 'read up'). Furthermore, an indirect information leak happens if a low subject S_{l} can observe the behavior of a high subject S_{h} and infer information.

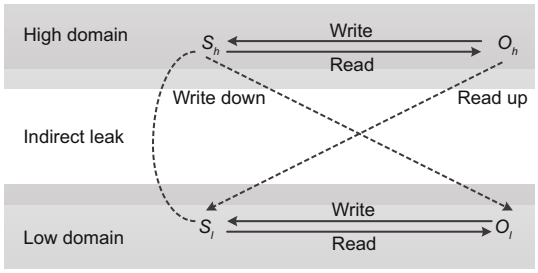

Fig. 2 Information flow interference between a high domain and a low domain

Interference can happen over various channels: besides explicit data transmission through message sending or shared storage (dataflow), there are a variety of covert channels. These include control flow, timing, and termination behavior, or the probability distribution of processed data (Sabelfeld and Myers, 2003). To formally prove that a system does not allow IF, the relevant aspects of its behavior are modeled and checked for an adequate noninterference property.

4 An approach for information flow analysis in BPEL

This section outlines the proposed approach for information flow analysis in BPEL. Given a BPEL model, the first step is to translate BPEL into a Petri net representation using an existing tool. The target meta-model is a labeled Petri net enriched with annotations to express the security classes for information flow analysis. According to the labeled Petri net, we build the Petri net reachability graph to analyze the Petri net interference and thereby detect the information leaks for web service composition.

4.1 Petri net

A Petri net is a well-founded process-modeling technique that has formal semantics. Petri nets have been used to model and analyze several types of processes including protocols, manufacturing systems, and business processes. A Petri net is a directed bipartite graph, in which the nodes represent places (i.e., conditions) or transitions (i.e., events that may occur). The directed arcs describe which places are preconditions and/or post-conditions for which transitions. Tokens occupy places. When there is at least one token in every place connected to a transition, we say that the transition is enabled. Any enabled transition may fire by removing one token from every input place, and depositing one token in each output place.

Definition 3 (Petri net) A Petri net is defined as a tuple $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F})$, where

1. $\mathcal{P}$ is a finite set of places; $\mathcal{T}$ is a finite set of transitions.

2. $\mathcal{P} \cap \mathcal{T} = \varnothing$; $\mathcal{P} \cup \mathcal{T} \neq \varnothing$.

3. $\mathcal{F} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is a set of directed arcs, called the 'flow relation'.

At any time, a place contains zero or more tokens. The marking (or state) is the distribution of tokens over places. A marking is a bag over the set of places $\mathcal{P}$, i.e., a function from $\mathcal{P}$ to the natural numbers: $\mathcal{M} \in \mathcal{P} \longrightarrow \mathbb{N}$. A partial ordering is defined to compare markings. For any two states $\mathcal{M}_1$ and $\mathcal{M}_2$, $\mathcal{M}_1 \leq \mathcal{M}_2$ if and only if $p \in \{\mathcal{P} : \mathcal{M}_1(p) \leq \mathcal{M}_2(p)\}$. The sum of two bags $(\mathcal{M}_1 + \mathcal{M}_2)$ is defined in a straightforward way.

A marked Petri net is a pair $(\mathcal{N}, \mathcal{M})$, where $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F})$ is a Petri net and $\mathcal{M}$ is a bag over $\mathcal{P}$ denoting the marking of the net. Elements of $\mathcal{P} \cup \mathcal{T}$ are called nodes. Node x is an input node of another node y if and only if there is a directed arc from x to y (i.e., $x\mathcal{F}y$). Node x is an output node of y if and only if $y\mathcal{F}x$. For any $x \in \mathcal{P} \cup \mathcal{T}$, $\cdot x = \{y | y\mathcal{F}x\}$ and $x\cdot = \{y | x\mathcal{F}y\}$.

$M(p_i)$ refers to the number of tokens on place p_i. The number of tokens may change during the execution of the net. Transitions are the active components in a Petri net. They change the state of the net according to the following firing rules:

1. Transition t is enabled if and only if each input place p of t contains at lease one token, i.e., $\forall\, p \in \cdot t,\ M(p) \geq 1$.

2. An enabled transition may fire. If transition t fires, then t consumes one token from each input place p of t and produces one token for each output place p of t. When transition t fires, the Petri net will change as follows:

$$M'(p)=\begin{cases}M(p)-1, & p\in{\cdot t}-t{\cdot},\\ M(p)+1, & p\in t{\cdot}-{\cdot t},\\ M(p), & \text{otherwise.}\end{cases}\tag{1}$$

Given a Petri net $\mathcal{N}=(\mathcal{P},\mathcal{T},\mathcal{F})$ and a state $\mathcal{M}_1$, the following notations are defined:

1. $\mathcal{M}_1\xrightarrow{t}\mathcal{M}_2$: transition t is enabled in $\mathcal{M}_1$ and firing t in M_1 results in state $\mathcal{M}_2$.

2. $\mathcal{M}_1\longrightarrow\mathcal{M}_2$: there is a transition t such that $\mathcal{M}_1\xrightarrow{t}\mathcal{M}_2$.

3. $\mathcal{M}_1\xrightarrow{\sigma}\mathcal{M}_n$: the firing sequence $\sigma=t_1,t_2,\ldots,t_{n-1}$ from state $\mathcal{M}_1$ leads to state $\mathcal{M}_n$ via a (possibly empty) set of intermediate states $\mathcal{M}_2,\mathcal{M}_3,\ldots,\mathcal{M}_{n-1}$, i.e., $\mathcal{M}_1\xrightarrow{t_1}\mathcal{M}_2\xrightarrow{t_2}\cdots\xrightarrow{t_{n-1}}\mathcal{M}_n$.

4. $\mathcal{M}_1\xrightarrow{*}\mathcal{M}_n$: there is a firing sequence σ so that $\mathcal{M}_1\xrightarrow{\sigma}\mathcal{M}_n$.

5. $[M_0>$: the set of reachable states of M_0, $[M_0>=\{M|M_0\xrightarrow{*}M\}$.

6. $\mathcal{M}_1\xrightarrow{t_1}\mathcal{M}_2\xrightarrow{t_2}\cdots\xrightarrow{t_{n-1}}\mathcal{M}_n$ is a path of the Petri net.

Definition 4 (Reachability graph) The reachability graph (RS) of a Petri net PN $=((\mathcal{P},\mathcal{T},\mathcal{F}),\mathcal{M})$ is a triple (S,s_0,R), where $S=[M_0>$ (M_0 is the initial state of PN), $s_0=M_0$ is the initial state of the reachability graph, and $R:S\times\mathcal{T}\longrightarrow S$ is defined as $s'=R(s,t)$ if and only if $s\xrightarrow{t}s'$.

4.2 Transforming BPEL to a Petri net

The transformation of the initial BPEL process employs the BPEL2oWFN tool, which implements a feature-complete mapping of BPEL 2.0 to the so-called open workflow net (OWFN), which is a variant of Petri nets tailored to workflow models. BPEL2oWFN (Fig. 3) can be used to translate BPEL into a standard Petri net, which can be analyzed for deadlocks or other classical Petri net properties, soundness, and temporal logical formulas by using a common model checker (Lohmann et al., 2009). In addition, BPEL2oWFN transforms a BPEL process into an OWFN, to analyze the interaction between BPEL processes and Fiona, which is a tool that decides controllability and computes the operating guideline (Lohmann et al., 2006).

4.3 Transforming a Petri net to a labeled Petri net

This study is concerned mainly about the information flow from high-level users to low-level users. Each user has a specific operating authority and the ability to observe the system. Some operations on the user are prohibited, and some of the output is not visible to the user. According to the operation and observation capabilities of different security-class users, input events can be divided into H user input events and L user input events. $\mathcal{T}_H$ and $\mathcal{T}_L$ are used to represent the set of these events, satisfying $\mathcal{T}_H\cap\mathcal{T}_L=\varnothing$. The events in $\mathcal{T}_L$ are visible to L users, but the events in $\mathcal{T}_H$ are not visible. $\mathcal{P}_L$ represents the set of places that L users can observe. For the change in system status caused by the events in $\mathcal{T}$, L users can observe only through the change in distribution of the token in $\mathcal{P}_L$.

A labeled Petri net is based on a Petri net, as these feature formal semantics and various notations of noninterference for Petri net-based systems were available in Busi and Gorrieri (2003). This study defines a labeled Petri net as PN $=((\mathcal{P},\mathcal{T}_H,\mathcal{T}_L,\mathcal{F}),\mathcal{M})$, where $\mathcal{T}_H\cap\mathcal{T}_L=\varnothing$ and $\mathcal{T}_H\cup\mathcal{T}_L=\mathcal{T}$. Additionally, the set of places that L users can observe is $\mathcal{P}_L=\cup_{t\in\mathcal{T}_L}({\cdot t}\cup t{\cdot})$.

In the PN operation, L users can observe only the events in $\mathcal{T}_L$ and the distribution of tokens in $\mathcal{P}_L$; that is, L users actually observe the partial execution of PN. We consider it to be a projection of L users, denoted by L_{RS}.

Definition 5 (L reachability graph) Given a labeled Petri net PN $=((\mathcal{P},\mathcal{T}_H,\mathcal{T}_L,\mathcal{F}),\mathcal{M})$, we define its reachability graph $\mathrm{RS}(S,s_0,R)$ projected on L users as $L_{\mathrm{RS}}(S_L,s_{0,L},R_L)$:

1. $S_L=\{s\downarrow_{P_L}|s\in S\}$, where $s\downarrow_{P_L}$ is defined as $\forall\,p\in P_L,\ s\downarrow_{P_L}(p)=s(p)$.

2. $s_{0,L}=s_0\downarrow_{P_L}$.

3. $R_L:S_L\times(T_L\cup\{\epsilon\})\longrightarrow S_L$ is the state transition function. $R_L(s\downarrow_{P_L},t)=s'\downarrow_{P_L}$ if and only if one of the following conditions holds: if $t\in T_L$, then $R(s,t)=s'$; if $t=\epsilon$, then $\exists\,t'\in T_H$, $R(s,t')=s'$.

The Petri net transformed from BPEL models the behavior of a single workflow instance. A web service composition may contain multiple instances,

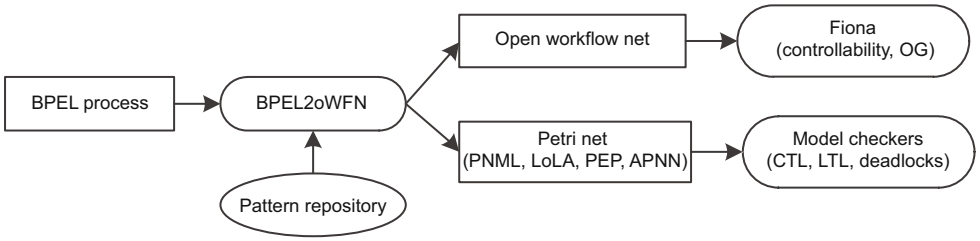

Fig. 3 BPEL2oWFN tool chain to analyze business process execution language (BPEL) processes

and when these instances attempt to access a specific resource but only one instance can hold the resource at one point in time, implicit information leaks may happen. Ahmad et al. (2010) gave us a method to model two or more instances of a Petri net and the interaction.

4.4 Place-based noninterference

Place-based noninterference (PBNI) is an approach for reasoning through the structural noninterference in Petri nets. Specifically, PBNI tackles the characterization and detection of places that correspond to a violation of noninterference.

Definition 6 (Conflict places) Given PN $= ((\mathcal{P}, \mathcal{T}_H, \mathcal{T}_L, \mathcal{F}), \mathcal{M})$, letting $p \in \mathcal{P}$ be a place of PN, p is a conflict place if and only if $p \cdot \cap \mathcal{T}_L \neq \varnothing$ and $p \cdot \cap \mathcal{T}_H \neq \varnothing$.

Definition 7 (Causal places) Given PN $= ((\mathcal{P}, \mathcal{T}_H, \mathcal{T}_L, \mathcal{F}), \mathcal{M})$, letting $p \in \mathcal{P}$ be a place of PN, p is a causal place if and only if $p \cdot \cap \mathcal{T}_L \neq \varnothing$ and $\cdot p \cap \mathcal{T}_H \neq \varnothing$.

Fig. 4 shows our deep understanding of conflict places and causal places. The essence of conflict places is that the occurrence of transition h hinders the occurrence of transition l, whereas the essence of causal places is that the occurrence of transition h leads to the occurrence of transition l. Thus, H user can transfer information to L user through both conflict places and causal places.

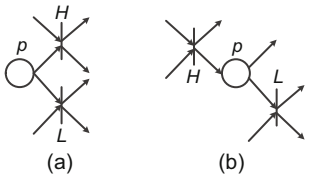

Fig. 4 Conflict (a) and causal (b) places

Definition 8 If PN $= ((\mathcal{P}, \mathcal{T}_H, \mathcal{T}_L, \mathcal{F}), \mathcal{M})$ has neither conflict places nor causal places, then PN has a noninterference property.

The key which causes the covert channel information flow is that there exists $s \xrightarrow{h} s'$ satisfying the condition of $s \downarrow p_L \neq s' \downarrow p_L$.

Zhou and Ju (2012) gave us the existence judgment of Petri net place-based noninterference, which is illustrated as follows:

Definition 9 PN $= ((\mathcal{P}, \mathcal{T}_H, \mathcal{T}_L, \mathcal{F}), \mathcal{M})$ has the interference property if and only if the reachability graph $L_{\mathrm{RS}}(S_L, s_{0,L}, R_L)$ satisfies $\exists\, s_1 \in S_L,\ s_2 \in S_L(s_1 \xrightarrow{h} s_2 \wedge s_1 \neq s_2)$.

According to Definition 9, Algorithm 1 determines whether the Petri net has an interference property by observing the L reachability graph.

In Algorithm 1, enabled(s) means that in state s, transition t is enabled. $s' = \mathrm{fire}(s, t)$ means that state s' is the next state when state s achieves transition t, and the calculation is shown in Definition 4. If the Petri net has interference, then we have reason to believe that there is information leakage.

5 Case study

To illustrate the idea behind the approach, we present a case study for the verification of the BPEL process. This case study demonstrates the verification of such an isolation requirement for concurrently running instances of a BPEL workflow. The BPEL workflow is transformed into a preliminary variant of a labeled Petri net. The transformation includes two steps: (1) the BPEL model is mapped into an equivalent Petri net model using an existing transformation tool—BPEL2oWFN; (2) the resultant Petri net is unfolded into a labeled Petri net, to which the security class is added, to model concurrently running instances and their access to shared resources.

The case study uses the 'update patient record' workflow depicted in Fig. 5. It consists of five activities (depicted as boxes) and an exclusive

Algorithm 1 Deciding whether the Petri net has interference

Input: Petri net PN = $((\mathcal{P}, \mathcal{T}_L, \mathcal{T}_H, \mathcal{F}), \mathcal{M}_0)$.

Output: 'true' if the Petri net has interference; 'false' if the Petri net has noninterference.

function main()
Initialize set stack s to empty
search($\mathcal{M}_0$)
end function
function search(s)
enabled(s) = $\varnothing$
while enabled(s) $\neq$ enable(s) **do**
  Select transition t from set enable(s)–enabled(s)
  $s' = \text{fire}(s, t)$
  if $t \in T_H$ and $s \downarrow p_L$ **then**
    return false
  end if
  enabled(s) = enabled(s) $\cup$ t
  Push s in the stack
  search(s')
end while
if stack is empty **then**
  return true
end if
Pull out the top element s''
search(s'')
end function

choice (depicted as a rhombus), which represents an if-statement. The corresponding BPEL fragment of the workflow is shown in Fig. 6.

The concurrent access to the same patient record causes information leaks. For instance, a doctor (H user) opens a specific patient's record and meanwhile a nurse (L user) attempts to access the same record. The nurse obtains information on the doctor's privacy behavior: she/he knows that the doctor uses the special record and might deduce from the patient that the doctor is currently treating, and the time at which the doctor needs to update the record. The value in this approach is in dealing with these privacy information leaks.

According to the BPEL fragment mentioned above (Fig. 6), we transfer it to a Petri net model using the BPEL2oWFN tool. The transformed Petri net model of the 'update patient record' workflow is depicted in Fig. 7.

The activities of the workflow are modeled by transitions, which are depicted as boxes. Places, depicted as circles, represent conditions. The state of the workflow, which is marked by a single token

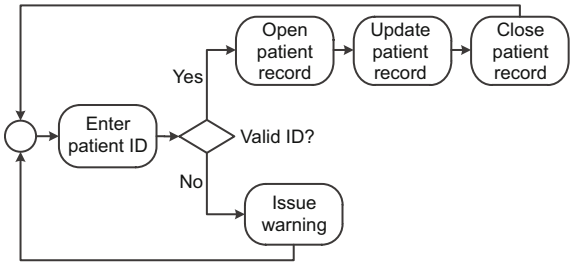

Fig. 5 'Update patient record' workflow

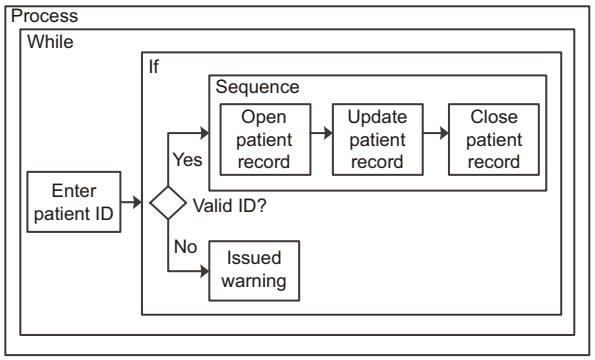

Fig. 6 Business process execution language (BPEL) fragment of the 'update patient record' workflow

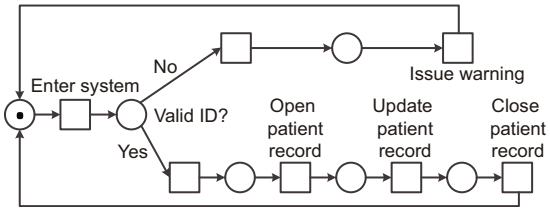

Fig. 7 Petri net of the workflow

and depicted as a black dot, initially resides in the start place. In addition, the arcs are depicted as arrows.

The Petri net models only the behavior of a single workflow instance. To check the information leak between high-level users and low-level users, we need to extend the Petri net to model two instances (including H and L users) of the workflow and their explicit interaction. The interaction between instances occurs through access to the patient record, which is a shared and limited resource. Multiple instances may attempt to access a specific patient record, but only one instance can hold the record at the same time. Thus, we change the Petri net to Fig. 8.

To simplify the analysis process, we analyze only the partial Petri net, which is marked by a dotted-line ellipse in Fig. 8. The partial Petri net to be analyzed is depicted in Fig. 9.

A Petri net can be transformed into a reachability graph to analyze the information flow interfer-

ence. According to Definition 4, we transform the Petri net (Fig. 9) to a reachability graph (Fig. 10) to describe the running states of the Petri net.

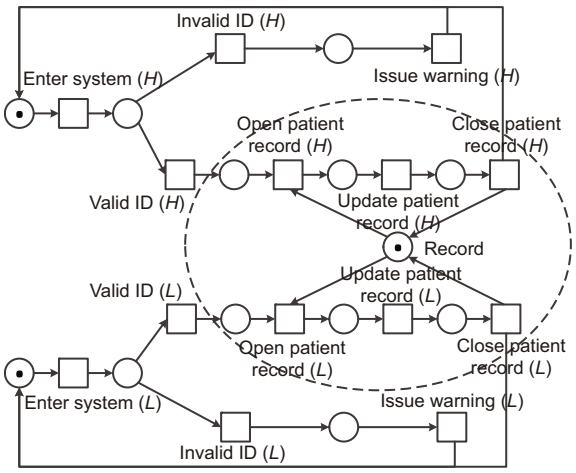

Fig. 8 Labeled Petri net of the workflow

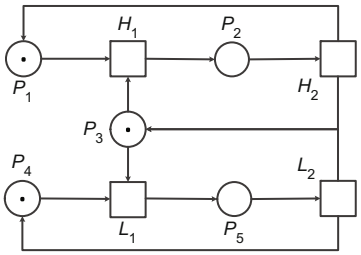

Fig. 9 Partial Petri net extracted from the workflow

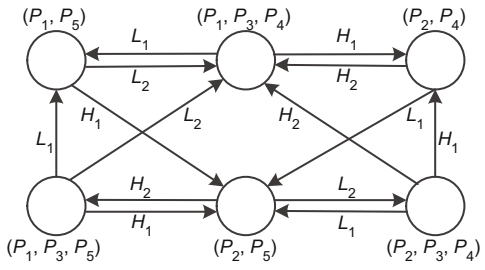

Fig. 10 Reachability graph of the partial workflow

Then we extract the L user's projection from the reachability graph to produce the L-reachability graph according to Definition 5. The L-reachability graph is depicted in Fig. 11.

In Fig. 11, we discover that state(P_3, P_4) can convert to state(P_4) by H_1, and state$(P_3, P_4) \neq$ state(P_4), which means that the Petri net has an interference property according to Definition 9. In other words, some state transitions of low-level users

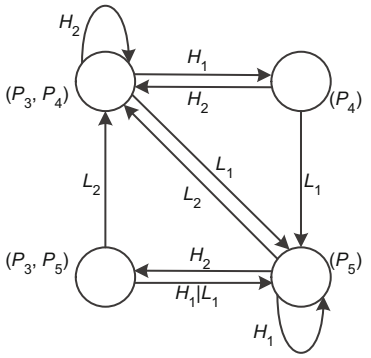

Fig. 11 *L*-reachability graph of the partial workflow

are determined by the behavior of the high-level users. Thus, there are some information leaks in this web service business process.

6 Conclusions and future work

We have presented an approach for information flow analysis of BPEL specifications to detect implicit information leaks for web service composition and demonstrated its applicability in a case study. Overall, the static detection of information flows for BPEL is a novel promising research direction to ensure formally founded security guarantees for web service composition.

Future work aims at three directions: (1) obtaining a reasonable distinction between leaks and necessary flows—because the enhanced expressive power provided by information flow analysis has a limitation, the interferences detected during the analysis may not necessarily denote an information leak, although they theoretically do so; (2) augmenting the set of information flow properties to be analyzed, such as resource conflicts and parameter confidentiality; (3) spelling out the consequences for more complex security models beyond the high and low classes. This allows one to capture fine-grained properties, which are especially useful for data-flow reasoning.

References

Accorsi R, Wonnemann C, 2009. Detective information flow analysis for business processes. Int Conf on Business Process, Services Computing, and Intelligent Service Management, p.223-224.

Accorsi R, Wonnemann C, 2010. Static information flow analysis of workflow models. Int Conf on Business Process and Service Science, p.194-205.

Accorsi R, Lehmann A, Lohmann N, 2015. Information leak detection in business process models: theory, applica-

tion, and tool support. *Inform Syst*, 47:244-257. https://doi.org/10.1016/j.is.2013.12.006

Ahmad F, Huang H, Wang X, 2010. Petri net modeling and deadlock analysis of parallel manufacturing processes with shared-resources. *J Syst Softw*, 83(4):675-688. https://doi.org/10.1016/j.jss.2009.11.705

Armando A, Ranise S, 2011. Automated analysis of infinite state workflows with access control policies. 7th Int Workshop on Security and Trust Management, p.157-174. https://doi.org/10.1007/978-3-642-29963-6_12

Atluri V, 2001. Security for workflow systems. *Inform Secur Technol Rep*, 6(2):59-68. https://doi.org/10.1016/S1363-4127(01)00207-2

Atluri V, Chun S, Mazzoleni P, 2001. A Chinese wall security model for decentralized workflow systems. 8th ACM Conf on Computer and Communications Security, p.48-57. https://doi.org/10.1145/501983.501991

Barkaoui K, Ayed R, Boucheneb H, et al., 2008. Verification of workflow processes under multilevel security considerations. 3rd Int Conf on Risks and Security of Internet and Systems, p.77-84. https://doi.org/10.1109/CRISIS.2008.4757466

Bell D, 1983. Secure computer systems: a retrospective. IEEE Symp on Security and Privacy, p.161-162. https://doi.org/10.1109/SP.1983.10001

Bell D, LaPadula L, 1973. Secure Computer Systems: Mathematical Foundations and Model. DTIC Document.

Benatallah B, Dumas M, Maamar Z, 2002. Definition and execution of composite web services: the self-serv project. *IEEE Data Eng Bull*, 25(4):47-52. http://sites.computer.org/debull/A02DEC-CD.pdf

Busi N, Gorrieri R, 2003. A survey on noninterference with Petri nets. ACPN: Lectures on Concurrency and Petri Nets, p.328-344.

Busi N, Gorrieri R, 2009. Structural noninterference in elementary and trace nets. *Math Struct Comput Sci*, 19(6):1065-1090. https://doi.org/10.1017/S0960129509990120

Carminati B, Ferrari E, Hung P, 2005. Exploring privacy issues in web services discovery agencies. *IEEE Secur Priv*, 3(5):14-21. https://doi.org/10.1109/MSP.2005.121

Denning D, 1976. A lattice model of secure information flow. *Commun ACM*, 19(5):236-243. https://doi.org/10.1145/360051.360056

Goguen J, Meseguer J, 1982. Security policies and security models. IEEE Symp on Security and Privacy, p.11-20. https://doi.org/10.1109/SP.1982.10014

Hui K, Tan B, Goh C, 2006. Online information disclosure: motivators and measurements. *ACM Trans Intern Technol*, 6(4):415-441. https://doi.org/10.1145/1183463.1183467

Juszczyszyn K, 2003. Verifying enterprise's mandatory access control policies with colored Petri nets. 12th IEEE Int Workshops on Enabling Technologies, p.184-189. https://doi.org/10.1109/ENABL.2003.1231405

Kagal L, Paolucci M, Srinivasan N, et al., 2004. Authorization and privacy for semantic web services. *IEEE Intell Syst*, 19(4):50-56. https://doi.org/10.1109/MIS.2004.23

Lampson B, 1973. A note on the confinement problem. *Commun ACM*, 16(10):613-615. https://doi.org/10.1145/362375.362389

Lohmann N, Massuthe P, Stahl C, et al., 2006. Analyzing interacting BPEL processes. 4th Int Conf on Business Process Management, p.17-32. https://doi.org/10.1007/11841760_3

Lohmann N, Verbeek E, Dijkman R, 2009. Petri net transformations for business processes—a survey. *Trans Petri Nets Other Models Concurr*, 2:46-63. https://doi.org/10.1007/978-3-642-00899-3_3

Lu Y, Zhang L, Sun J, 2009. Using colored Petri nets to model and analyze workflow with separation of duty constraints. *Int J Adv Manuf Technol*, 40(1-2):179-192. https://doi.org/10.1007/s00170-007-1316-1

Massacci F, Mylopoulos J, Zannone N, 2006. Hierarchical hippocratic databases with minimal disclosure for virtual organizations. *VLDB J*, 15(4):370-387. https://doi.org/10.1007/s00778-006-0009-y

Myers A, Liskov B, 1997. A decentralized model for information flow control. 16th ACM Symp on Operating System Principles, p.129-142. https://doi.org/10.1145/268998.266669

Papazoglou M, 2012. Cloud blueprint: a model-driven approach to configuring federated clouds. 2nd Int Conf on Model and Data Engineering, p.1. https://doi.org/10.1007/978-3-642-33609-6_1

Röhrig S, Knorr K, 2004. Security analysis of electronic business processes. *Electron Commerce Res*, 4(1-2):59-81. https://doi.org/10.1023/B:ELEC.0000009282.06809.c5

Sabelfeld A, Myers A, 2003. Language-based information-flow security. *IEEE J Sel Areas Commun*, 21(1):5-19. https://doi.org/10.1109/JSAC.2002.806121

Shafiq B, Masood A, Joshi J, et al., 2005. A role-based access control policy verification framework for real-time systems. 10th IEEE Int Workshop on Object-Oriented Real-Time Dependable Systems, p.13-20. https://doi.org/10.1109/WORDS.2005.11

Singh M, 2001. Being interactive: physics of service composition. *IEEE Intern Comput*, 5(3):6-7.

Sun H, Wang X, Yang J, et al., 2008. Authorization policy based business collaboration reliability verification. 6th Int Conf on Service-Oriented Computing, p.579-584. https://doi.org/10.1007/978-3-540-89652-4_49

Tan W, Fan Y, Zhou M, 2009. A Petri net-based method for compatibility analysis and composition of web services in business process execution language. *IEEE Trans Autom Sci Eng*, 6(1):94-106. https://doi.org/10.1109/TASE.2008.916747

Tbahriti S, Ghedira C, Medjahed B, et al., 2014. Privacy-enhanced web service composition. *IEEE Trans Serv Comput*, 7(2):210-222. https://doi.org/10.1109/TSC.2013.18

Tschantz M, Datta A, Datta A, et al., 2015. A methodology for information flow experiments. 28th IEEE Symp on Computer Security Foundations, p.554-568. https://doi.org/10.1109/CSF.2015.40

Yee G, 2007. A privacy controller approach for privacy protection in web services. 4th ACM Workshop on Secure Web Services, p.44-51. https://doi.org/10.1145/1314418.1314426

Zhou C, Ju S, 2012. A Petri net based approach to covert information flow analysis. *Chin J Comput*, 35(8):1688-1699. https://doi.org/10.3724/sp.j.1016.2012.01688