 Verification of Behavior-aware Privacy Requirements in Web Services Composition

Jiajun Lu
College of Computer Science and Technology, Nanjing University
of Aeronautics and Astronautics, Nanjing, 210016
Email: lujiajun.ck@163.com

Zhiqiu Huang and Changbo Ke
College of Computer Science and Technology, Nanjing University
of Aeronautics and Astronautics, Nanjing, 210016
Email: zqhuang@nuaa.edu.cn

Abstract—Privacy has been acknowledged to be a critical concern for many collaborative business environments. Recently, verifying whether Web services composition satisfies privacy requirement is a hot spot for privacy protection. However, little research focuses on behavioral privacy requirement. This paper proposes an approach based on model checking to verify the satisfiability of behavior-aware privacy requirements in services composition. Firstly, we extract LTL specification from the behavior constrains of privacy requirements. On the other side, the behavior of BPEL process is modeled by extended interface automata, which supports privacy semantics. Then it is transformed to Promela description, the input language of the model checker SPIN. Finally, we illustrate the verification of privacy requirements with SPIN.

Index Terms—Web services composition, behavior-aware privacy requirements, model checking, SPIN

I. INTRODUCTION

Recently, Web services are applied widely in the academic and industry field as a new distributed computing model [1]. Users have to submit some personal information, which is privacy sensitive, to service providers to finish the necessary business. Because of the fact that the technologies of Web services spring up and users’ requirements increase rapidly, multiple web services are composed to fulfill more business requirements [2] as the single Web service is not competent. In the process of composing, service providers may expose some of user’s sensitive information to other collaborators. Owing to none of protocols between services and users is designed to specify the behavior, it is hardly to guarantee that user’s personal data is exposed and applied according to users’ intension, especially in cloud computing environment [3,4,5]. As a result, it is becoming a critical problem to ensure that services composition concurrently fits users’ privacy demands while users’ business requirements are meet.

In the information system and software engineering domain, privacy protection represents the capability that the individual control the collection, exposition and maintenance of information about themselves [6]. As Web service privacy has became a research hot spot of service computing, many organizations have proposed a series of software industry standards and technology implementation frameworks supporting privacy protection. The Platform for Privacy Preferences (P3P) [7] presented by W3C and corresponding privacy preferences description language APPEL (A P3P Preference Exchange Language) [8] are capable to define privacy policy of service providers and privacy preferences of users. OASIS provides eXtensible Access Control Markup Language (XACML) [9] to manage access control and extend support for privacy policy via privacy policy profile. Some researchers add privacy property into the role-based access control model RBAC. As a result, a privacy-aware role-based access control model is put forward to describe privacy-related access control policy [10].

In order to fulfill the functional requirement, web service provider must choose a group of Web services to achieve business target while ensure that the disclosure of users’ privacy data meets their privacy requirements. As a result, it is necessary to analyze privacy requirements of service composition in the design phrase, that is, verify whether Web service composition utilizes privacy data according to privacy policies. At present, some research working on this aspect has been done in the domestic and overseas. Yin Hua Li and Boualem Benatallah correspond Web service business process execution language (WS-BPEL) with P3P policy description and verify the consistency [11]. Linyuan Liu and Zhiqiu Huang transform access authority on privacy data into the privacy policy, model services composition and verify the privacy requirement on the model at last [12]. Adam Barth et al. express privacy properties of service users with linear temporal logic (LTL) formulas and verify privacy requirement based on the composition model [13]. The authors in paper [14] model service assembly with hypergraph and provide the method to transform, furthermore propose an algorithm which can achieve the minimal privacy disclosure service assembly. However,
the privacy requirements above which researchers consider just restrict the single Web service’s access on privacy data. It is hardly sufficient to protect privacy in cross-organizational services composition. The interaction between services must to be taken into account in privacy protection. To supplement current research, this paper proposes a model checking approach based on temporal properties verification to check the satisfiability of behavior-aware privacy requirement in Web services composition. The main ideas of our work can be depicted as Fig.1.

![Privacy Requirement Diagram](Image)

**Figure 1. The framework of privacy requirement verification**

The remainder of this paper is organized as follows. Section 2 analyzes behavior-aware privacy requirement combining particular application scenario. Section 3 presents the method to extract the LTL specification of privacy requirement. Section 4 models the behaviors of BPEL services composition by extending interface automata to support privacy semantics and transforms form interface automata of BPEL to Promela description. Section 5 uses SPIN model checker to verify whether the behaviors of services composition satisfy with the privacy requirement through a case study. Finally Section 6 concludes the paper.

### II. PRIVACY REQUIREMENT ANALYSIS

The interaction of services in composition may bring about unpredictable privacy concerns. As the privacy requirements of users are becoming more and more distinct, some requirement constraints focus on the interaction behavior of services, namely behavior-aware privacy requirement. The constraints comprise three types, data-activity dependency, data dependency and data mutex.

Take a shopping online scenario as example. There are several collaborative services: OnlineShopping, Creditcard, Debitcard, Shipper, E-mail and Message. At the beginning, the buyer sends an order request (OrderReq) to the OnlineShopping. Then the service Creditcard and Debitcard can be chosen as two kinds of online payment methods. If payment is successful, OnlineShopping informs the service Shipper to deliver the goods. After the OnlineShopping receives arrival message of the goods, it informs the buyer to pick up the goods by invoking the service E-mail or Message. Fig.2 presents the web service invocation of shopping online process.

![Service Invocation Diagram](Image)

**Figure 2. The service invocation of OnlineShopping process**

The OnlineShopping process needs to collect user’s privacy data name, creditcard_no, debitcard_no, address, e-mail and mobile_no, and then discloses some of them to the collaborative services as required. The behavior-aware privacy requirements involved in this services composition can be divided into three types, just as follows:

1. Date-activity dependency. These requirements limit that the access of individual privacy data must take the occurrence of certain activity as a condition. For example, data mobile_no can be accessed only after activity Shipper has finished.

2. Data dependency. The access sequence of multiple privacy data is restricted in these requirements. For example, data address can be accessed after creditcard_no was used.

3. Data mutex. The use of some privacy data must be mutually exclusive in a service composition. For example, mobile_no and creditcard_no can’t be both possessed by the OnlineShopping process.

All the three types of constraints can be described as the temporal relations between data and activities or between data and data, and further expressed by LTL formulas.

### III. LTL SPECIFICATION OF PRIVACY REQUIREMENT

Linear Temporal Logic (LTL) is based upon the propositional calculus and used to describe assertions that changes over time through introducing temporal operators [15]. LTL formulas are constructed from atomic propositions, logical operators “∧”, “∨”, “¬”, and temporal operators. Some LTL temporal operators indicating the future are X (meaning “next”), G (“globally”), F (“eventually”), and U (“until”). The corresponding past time operators in LTL are Y (meaning “yesterday”), H (“historically”), O (“once”) and S (“since”). The semantic of LTL temporal operators can be easily defined on finite length service composition privacy behaviors. Given a services composition model M and an LTL property Φ, we say that M |= Φ, iff for any path w, w |= Φ. The three types of behavior-aware privacy requirements discussed in Section II can totally be
described as temporal properties in a computation path (for short “path”) and further expressed with LTL formulas. Especially, the mutex constrains of privacy data in data mutex privacy requirement can also be transformed to temporal relation between them. The correspondence between privacy requirement types and LTL formulas are shown in Table 1.

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>LTL formula</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-activity dependency</td>
<td>Privacy data name can be used after activity Login has finished.</td>
<td>( G(name \rightarrow G(Login)) )</td>
<td>If data name is going to be accessed, it is required that activity Login has occurred.</td>
</tr>
<tr>
<td>Data dependency</td>
<td>Privacy data address can be accessed after name was used.</td>
<td>( G(address \rightarrow O(name)) \leftrightarrow \neg(\neg name \cup address) )</td>
<td>If data address is going to be accessed, it is required that data name has been used. In other words, it is not allowed that address is accessed while name has not been used.</td>
</tr>
<tr>
<td>Data mutex</td>
<td>Privacy data creditcard_no and mobile_no couldn’t be used by a service composition.</td>
<td>( G([creditcard_no \rightarrow G-mobile_no] \lor [mobile_no \rightarrow G-creditcard_no]) )</td>
<td>If data creditcard_no has been accessed, data mobile_no will not be used any more, vice versa.</td>
</tr>
</tbody>
</table>

Based on the correspondence, we can transform different types of privacy requirements to LTL specification.

IV. MODELING THE PRIVACY BEHAVIORS OF SERVICES COMPOSITION

In this paper, we use SPIN model checker \([16,17]\) to verify whether the behaviors of BPEL services composition satisfy privacy requirements. The input language of SPIN is called Promela, a modeling language for finite-state concurrent processes. Promela specifications allow us to model BPEL workflow using a set of concurrent processes that can communicate with each other. We implement the modeling of privacy behavior in two phases: (1) model the interface behaviors of BPEL process by extending interface automata to support privacy semantics, (2) transform from the interface automata model to Promela description. The interface automata as an intermediate model achieved the extraction of control information and privacy data of BPEL.

A. BPEL Modeling with Interface Automata

Interface Automata (IA) is a formal model to describe the temporal aspect of software component interface. Specifically, it’s designed to capture effectively both input assumptions and output guarantees about the order of the interactions between a component and its environment. For more details, please refer to \([18]\). Besides, the description of interfaces and behaviors for a component is either supported by IA. As a result, IA is employed to express interface behaviors of service. The operations in the service interface correspond to the actions in IA. Since the privacy data will be accessed when the service is invoked, we add the corresponding privacy data requirement to every transition, which is called Privacy Interface Automata (PIA).

**Definition 1: PIA.** A privacy interface automata can be defined as \( P := <V_P, V_P^{init}, A_P, D_P, \Gamma_P> \), where:

- \( V_P \) is a finite set of states, each state \( v \in V_P \).
- \( V_P^{init} \) is the set of initial state, \( V_P^{init} \subseteq V_P \). We require that \( V_P^{init} \) contains at most one state. If \( V_P^{init} = \emptyset \), then \( P \) is called empty.
- \( A_P \) is a finite set of actions, including Input, Output and Internal actions: \( A_P^I, A_P^O \) and \( A_P^I \), they are mutually disjoint. We denote \( A_P = A_P^I \cup A_P^O \cup A_P^I \).
- \( D_P \) is a finite set of privacy data access arrays, for short privacy array. For each privacy array \( d \in D_P \), it is consisted of \(|O|\) elements, where \( O \) denotes a finite set of privacy data objects. As the elements of array \( d[i] \in \{0,1\} \) \( 0 \leq i \leq |O| \), \( 1 \) denotes to access privacy data \( i \) and \( 0 \) denotes not.
- \( \Gamma_P \) is a finite set of transitions, \( \Gamma_P \subseteq V_P \times A_P \times D_P \times V_P \).

We use state transition sequence with privacy array to express the behavior of a PIA: 

\[
\begin{align*}
  v_0 & \xrightarrow{a_0,d_0} v_1 \xrightarrow{a_1,d_1} \cdots \xrightarrow{a_{n-1},d_{n-1}} v_{n-1} \xrightarrow{a_n,d_n} v_n
\end{align*}
\]

, where \( n \) is a non-negative integer. The system starts from state \( v_0 \) transits to \( v_1 \) via \( a_0 \) and requests access \( d_0 \), and then transits to \( v_2 \) via \( a_1 \) and requests access \( d_1 \), the rest may be deduced by analogy.

Based on the PIA definition and WS-BPEL specification \([19]\), the transformation from BPEL process
to PIA is presented as Fig.3. Some typical primitive activities and structured activities are listed as examples.

<table>
<thead>
<tr>
<th>BPEL</th>
<th>Sample Code</th>
<th>Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>receive</td>
<td><code>&lt;receive operation = “op” variable = “opmsg”/&gt;</code></td>
<td>opmsg?, d</td>
</tr>
<tr>
<td>reply</td>
<td><code>&lt;reply operation = “op” variable = “opmsg”/&gt;</code></td>
<td>opmsg!, d</td>
</tr>
<tr>
<td>invoke</td>
<td><code>&lt;invoke operation = “op” inputVariable = “inputmsg”/&gt;</code></td>
<td>inputmsg!, d</td>
</tr>
<tr>
<td>sequence</td>
<td><code>&lt;sequence&gt;&lt;act1…&gt;&lt;act2…&gt;&lt;/sequence&gt;</code></td>
<td>act1 act2</td>
</tr>
<tr>
<td>switch</td>
<td><code>&lt;switch&gt;&lt;case condition = “”&gt;act1&lt;/case&gt;&lt;case condition = “”&gt;act2&lt;/case&gt;&lt;/switch&gt;</code></td>
<td>act1 act2</td>
</tr>
</tbody>
</table>

Figure 3. Translation from BPEL to PIA

As shown in the figure, the input message of receive statement corresponds to an input action in the transition of PIA, while the output message of reply and invoke statement corresponds to an output action. Data variables in the message can be obtained from the WSDL of services. The privacy array is constructed by all privacy data in the message variables, and then added to the transition of PIA. Especially, when modeling invoke statement, we only consider unidirectional invoke activity, which only sends call requests to collaborative services in the composition and doesn’t need any response. The reason is that generally privacy data only appears in request message rather than response message of a Web service.

**Example 1:** Fig.4 describes the BPEL process of **OnlineShopping** discussed in Section II.

![Figure 4. The BPEL process of OnlineShopping](image)

The process **OnlineShopping** sends or receives message from collaborative services through operations described in the interfaces. Each message variable would likely contain some privacy data so that the privacy data may be accessed when the message is delivered. The privacy data involved in process **OnlineShopping** is listed in Table 2.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Operation</th>
<th>Message</th>
<th>Privacy Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>receive</td>
<td>OrderReq?</td>
<td>orderReq</td>
<td>name, creditcard_no, debitcard_no, address, e-mail, mobile_no</td>
</tr>
</tbody>
</table>

Table 2.

Privacy Data in Process **OnlineShopping**
invoke Creditcard! creditcardReq name, creditcard_no
invoke Debitcard! debitcardReq name, debitcard_no
invoke Shipper! shipReq name, address
invoke E-mail! e-mailReq name,e-mail
invoke Message! messageReq name, mobile_no
reply End! endReq N/A

Based on the definition of PIA and the transformation rule from BPEL to PIA, the PIA model of OnlineShopping is illustrated in Fig.5.

We label a privacy data access array for each transition of PIA. The corresponding actions and privacy arrays of the PIA model of OnlineShopping are shown in Table 3, where elements of each array denote the privacy data name, creditcard_no, debitcard_no, address, e-mail and mobile_no respectively. The elements of the array indicate whether OnlineShopping requests to access a privacy data, where 1 indicates that it requests and 0 indicates not.

<table>
<thead>
<tr>
<th></th>
<th>OrderReq?, d0</th>
<th>Creditcard!, d1</th>
<th>Debitcard!, d2</th>
<th>Shipper!, d3</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-mail!, d4</td>
<td>1, 0, 0, 1, 0</td>
<td>1, 0, 0, 0, 0</td>
<td>1, 0, 0, 1, 0</td>
<td>0, 0, 0, 0, 0</td>
</tr>
</tbody>
</table>

In Fig.5, after OnlineShopping receives the input action OrderReq? in the initial state v0, the state transits to v1. In this transition process, OnlineShopping collects the privacy data name, creditcard_no, debitcard_no, address, e-mail and mobile_no respectively. When the output action Creditcard! occurs, the state transits from v1 to v2. In this transition process, OnlineShopping discloses privacy data name and creditcard_no to collaborative service Creditcard. All of the subsequent behaviors of OnlineShopping can be deduced.

B. Transformation from PIA to Promela

A BPEL Web services composition specified using PIA will be transformed into Promela specification which consists of a set of concurrent processes. Each process represents the action in the transition τ of PIA and is connected by a set of communication channels. The processing steps of the transformation algorithm are as follows:

1. Traverse the transition τ of PIA model and get the action a and privacy array d to build new activities in array Activities. Then create transfers between activities in array Transfers (distinguish from transition in PIA). If τ has more than one next transition τ' in PIA, set condition to each transfer.

2. Traverse the array Transfers to generate the declaration of array mtype and channel message.

3. Traverse the activity of Activities to generate the variable declaration of the activity and involved privacy data. Create a new proctype process for each activity. Search the Transfers in each activity and add transfer’s condition to the related process unless it is empty. If the current activity is the start activity of the transfer, then generate an output message. If it is the end activity, then generate an input message.
4. Label the finish of the activity and the access of corresponding privacy data at the end of each process.

**Algorithm 1**: Transformation from PIA to Promela  
**Input**: The PIA model of BPEL process, PIA  
**Output**: The Promela description of BPEL process, Promela

```plaintext
for all τ ∈ PIA.Γ_p  
Acti vities.addNewActivity (getAction (τ), getData (τ));  
for all τ' ∈ PIA.Γ_p  
    //the arrival state of τ equals to the start state of τ'  
    if (getArrivalState (τ) = getStartState (τ')) then  
        Transfers.addNewTransfer(getAction(τ),  
                          getAction(τ'), condition); //create transfer  
for all activity ∈ Activities  
Promela.NewVariable (activity); //variable declaration  
// proctype process declaration  
Promela.NewProctype (activity);  
for all transfer ∈ Transfers  
    Promela.proctype.AddCondition (getCondition(transfer));  
if (getStartActivity(transfer) = activity) then  
    //generate the output message  
    Promela.proctype.SendMsg(getNum(transfer),  
                            transfer); //getNum() is to get the position of  
                            //transfer in array mtype  
if (getEndActivity(transfer) = activity) then  
    Promela.proctype.ReceiveMsg(getNum(transfer),  
                            transfer); //generate the input message  
Promela.SetTrue (getVariable(activity));  
// label the finish of the activity  
return Promela;
```

V. PRIVACY REQUIREMENT VERIFICATION

Take the shopping online scenario in Section II as example to illustrate the verification of behavior-aware privacy requirements with model checker SPIN. The BPEL process and PIA model have been presented in Section IV.  

According to Algorithm 1, firstly, we should transform from the PIA model to Promela description, which consists of type declaration (mtype), channel declaration (chan msg), variable declaration, process declaration (proctype) and so on, to indicate the communication among processes. Some code fragment is listed in Fig.6.

```plaintext
mtype = { creditcard_req, debitcard_req,  
          creditcard_shipper, debitcard_shipper, email_req,  
          message_req, email_end, message_end };  
chan msg1 = [2]of{mtype};  
......  
// variable declaration of condition  
bool creditcard_select = true;  
bool email_select = true;  
......  
// variable declaration of the finish of an activity  
bool end_done = false;  
......  
// variable declaration of privacy data  
bool email_no = false;  
bool mobile_no = false;  
......

active proctype orderReq(){  
    if ::(creditcard_select == true) -> msg1!creditcard_req  
        ::(creditcard_select == false) -> msg2!debitcard_req  
    fi;
}
active proctype creditcard(){  
    msg1?creditcard_req;  
    creditcard_no = true;  creditcard_done = true;  
    msg3!creditcard_shipper;
}
......
active proctype end(){  
    if ::(email_select == true) -> msg7?email_end  
        ::(email_select == false) -> msg8?message_end  
    fi;  
    end_done = true;
}
```

Figure 6. Promela description of the PIA model of OnlineShopping

The behavior-aware privacy requirements in shopping online scenario are as follows:

1) After service Shipper has finished, privacy data mobile_no is allowed to be accessed.
2) Only after privacy data debitcard_no has been used, e-mail could be accessed.
3) Privacy data creditcard_no and mobile_no could not be used by a service composition.

The requirements can be categorized as data-activity dependency, data dependency and data mutex. According to the correspondence between privacy requirement types and LTL formulas listed in Table 1, they can be expressed with LTL as follows:

- \( G(mobile_no \rightarrow G \text{shipper\_done}) \)
- \( \neg (\neg debitcard_no \ U \text{email\_no}) \)
- \( G[ (creditcard_no \rightarrow G\neg mobile_no) \vee (mobile_no \rightarrow G\neg creditcard_no)] \)

Then, translate the formulas above into the form that SPIN model checker can recognize:
Finally, input the Promela description of process OnlineShopping and LTL formulas into SPIN and verify them. As Fig.7, when verifying formula 1, “errors: 0” occurs. After exhausting state space, the fact that all paths meet LTL formula 1 implies that Promela model satisfies the specification.

VI. CONCLUSIONS

The result for verifying formula 2 is below. Program breaks off at the depth 30 and a counter-example “assertion violated !(email_no)” is achieved. It means that LTL formula 2 is not satisfied by an existing path starting from the initial state. As a result, the model doesn’t satisfy this specification.

The counter-example path: queue1 (msg1) -> queue2 (msg3) -> queue3 (msg5) in the file .trail records that process OnlineShopping invokes service Creditcard, Shipper and E-mail in sequence and privacy data creditcard_no, address and mobile_no are totally accessed. The third requirement is violated.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Grant No. 61272083, No. 61262002 and No. 61170043) and China Postdoctoral Science Foundation of China (Grant No. 20110491411).

REFERENCES


Jiajun Lu was born in 1988 and received the B.S. degree in Computer Science and Technology. Now he is a master candidate at College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Jiangsu, China. His research interests include service-oriental computing and privacy.

Zhiqiu Huang was born in 1965. He received his Ph.D. degree in Computer Science from Nanjing University of Aeronautics and Astronautics in 1999. Now he is a professor and Ph.D. supervisor at College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. His research interests include software engineering, formal methods, cloud computing and privacy.

E-mail: zqhuang@nuaa.edu.cn

Changbo Ke was born in 1984. He is a Ph.D candidate of Nanjing University of Aeronautics and Astronautics. His research interests include security and privacy of information system and ontology-based software engineering.