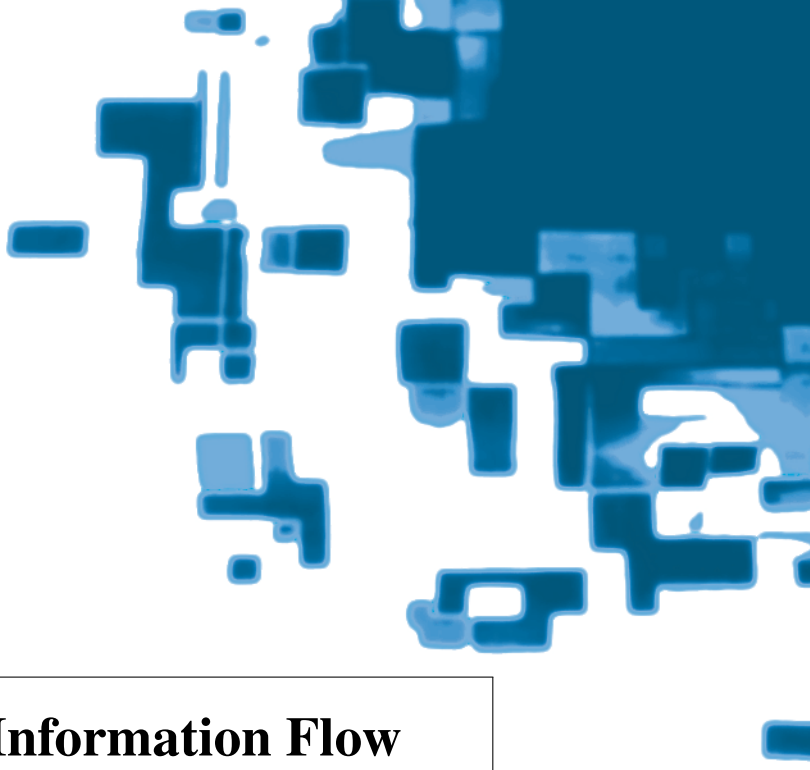

Model-driven Information Flow Security for Component-Based Systems

Najah Ben Said , Takoua Abdellatif, Saddek Bensalem, Marius Bozga

Verimag Research Report n^o TR-2013-7

June 2013

Reports are downloadable at the following address
http://www-verimag.imag.fr

Unité Mixte de Recherche 5104 CNRS - Grenoble INP - UJF
Centre Equation
2, avenue de VIGNATE
F-38610 GIERES
tel : +33 456 52 03 40
fax : +33 456 52 03 50
http://www-verimag.imag.fr

Model-driven Information Flow Security for Component-Based Systems

Najah Ben Said , Takoua Abdellatif, Saddek Bensalem, Marius Bozga

June 2013

Abstract

This paper proposes a framework for information flow security in component-based systems which follows the model-driven security approach. The security policy is defined and verified from the early steps of the system design. Two kinds of non-interference properties are formally introduced and for both of them, sufficient conditions that ensures and simplifies the automated verification are proposed. The verification is compositional, first locally, by checking the behavior of every atomic component and then globally, by checking the inter-components communication and coordination. The benefit of the approach is illustrated through an application to secure heterogeneous distributed systems.

Keywords: component-based systems, information flow security, non-interference, unwinding conditions, verification.

Reviewers:

Notes: The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 318772 - Distributed MILS for Dependable Information and Communication Infrastructures (D-MILS).

How to cite this report:

```
@techreport {TR-2013-7,
  title = {Model-driven Information Flow Security for Component-Based Systems},
  author = {Najah Ben Said , Takoua Abdellatif, Saddek Bensalem, Marius Bozga},
  institution = {{Verimag} Research Report},
  number = {TR-2013-7},
  year = {}
}
```

1 Introduction

The amount and complexity of nowadays conceived systems and software knows a continuous increase. Information protection and secure information flow between these systems is paramount and represent a great design challenge. Model driven security (MDS) [7] is an innovative approach that tends to solve system-level security issues by providing an advanced modeling process representing security requirements at a high level of abstraction. Indeed, MDS guarantees separation of concerns between functional and security requirements, from early phases of the system development till final implementation.

Information flow security can be ensured using various mechanisms. Amongst the first approaches considered, ones find access control mechanisms [26, 19], that allow protecting data confidentiality by limiting access to data to be read or modified only by authorized users. Unfortunately, these mechanisms have been proven incomplete and limited since only by preventing the direct access to data, indirect (implicit) information flows are still possible given rise to the so called covert channels [28]. As an alternative, non-interference has been studied as a global property to characterize and to develop techniques ensuring information flow security. Initially defined by Goguen and Meseguer [17], non-interference ensures that the system's secret information does not affect its public behavior.

In this work, we adapt the MDS approach to develop a component-based framework, named *secBIP* that guarantees automated verification and implementation of secure information flow systems with respect to specific definition of non-interference. In general, component-based frameworks allow the construction of complex systems by composition of atomic components with communication and coordination operators. That is, systems are obtained from unitary atomic components that can be independently deployed and composed with other components. Component-based frameworks are usually well adopted for managing key issues for functional design including heterogeneity of components, distribution aspects, performance issues, etc. Nonetheless, the use of component-based frameworks is also beneficial for establishing information flow security. Particularly, the explicit system architecture allows tracking easily intra and inter-components information flow.

The *secBIP* framework is built as an extension of the *BIP* [9, 8] framework encompassing information flow security. *SecBIP* allows creating systems that are secure by construction if certain local conditions hold for composed components. The *secBIP* extension includes specific annotations for classification of both data and events. Thanks to the explicit use of composition operators in *BIP*, the information flow is easily tracked within models and security requirements can be established in a compositional manner, first locally, by checking the behavior of atomic components and then globally, by checking the communication and coordination inter-components.

Information flow security has been traditionally studied separately for language-based models [25, 29] (see also the survey [24]) and trace-based models [21, 22, 30, 20]. While the former mostly focus on verification of data-flow security properties in programming languages, the latter is treating security in event-based systems. In *secBIP*, we achieve a useful combination between both aspects, data-flow and event-flow security, in a single semantics model. We introduce and distinguish two types of non-interference, respectively *event non-interference* and *data non-interference*. For events, non-interference states that the observation of public events should not allow deducing any information about the occurrence of secret events. For data, it states that there is no leakage of secret data into public ones.

The paper is structured as follows. Section 2 recalls the main concepts of the component-based framework adopted in this work. In section 3, we formally introduce the security extension and we provide the two associated definitions of non-interference, respectively for data flows and event flows. Next, in section 4 we formally establish non-interference based on unwinding relations and we provide sufficient conditions that facilitate its automatic verification. In section 5, we provide a use-case as illustrative example. Section 6 discusses the related work and section 7 concludes and presents some lines for future work. Finally, due to space limitations, all the proofs of technical results are given in an appendix. A full self-contained version including the proofs can be found as technical report [**?**].

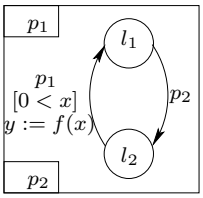

Figure 1: Atomic Component.

2 Component-Based Design

The *secBIP* framework is built as an extensions of the *BIP* [9, 8]. BIP is a formal component-based framework for modeling, design and implementation of heterogeneous real-time systems, fully implemented and available on [1]. *BIP* stands for *behavior*, *interaction* and *priority*, that is, the three layers used for the definition of components and their composition in this framework. *BIP* allows the construction of complex, hierarchically structured models from atomic components characterized by their behavior and their interfaces. Such components are transition systems enriched with data. Transitions are used to move from a source to a destination location. Each time a transition is taken, component data (variables) may be assigned new values, computed by user-denied functions (in C). Atomic components are composed by layered application of interactions and priorities. Interactions express synchronization constraints and do the transfer of data between the interacting components. Priorities are used to filter amongst possible interactions and to steer system evolution so as to meet performance requirements e.g., to express scheduling policies.
In this section, we formally introduce the key concepts of *BIP* which are further relevant for dealing with information flow security. In particular, we give a formal definition of atomic components and their composition through multiparty interactions. Priorities are not considered in this work.

2.1 Atomic Components

Definition 1 (atomic component) *An atomic component B is a tuple (L, X, P, T) where L is a set of locations, X is a set of variables, P is a set of ports and $T \subseteq L \times P \times L$ is a set of port labelled transitions. For every port $p \in P$, we denote by X_p the subset of variables exported and available for interaction through p. For every transition $\tau \in T$, we denote by g_τ its guard, that is, a boolean expression defined on X and by f_τ its update function, that is, a parallel assignment $\{x := e^x_\tau\}_{x \in X}$ to variables of X.*

Example 1 *Figure 1 provides an example of an atomic component. It contains two control locations l_1 and l_2 and two ports p_1 and p_2. The transition labeled with p_1 can take place only if the guard $(0 < x)$ is true. When the transition takes place, the variable y is recalculated as some function of x.*

Let $\mathcal{D}$ be the data domain of variables. Given a set of variables Y, we call valuation on Y any function $\mathbf{y} : Y \to \mathcal{D}$ mapping variables to data. We denote by $\mathbf{Y}$ the set of all valuations defined on Y.

Definition 2 (atomic component semantics) *The semantics of an atomic component $B = (L, X, P, T)$ is defined as the labelled transition system $\text{LTS}(B) = (Q_B, \Sigma_B, \underset{B}{\longrightarrow})$ where the set of states $Q_B = L \times \mathbf{X}$, the set of labels is $\Sigma_B = P \times \mathbf{X}$ and the set of labelled transitions $\underset{B}{\longrightarrow}$ is defined by the rule:*

$$\textsc{Atom}\ \frac{\tau = \ell \xrightarrow{p} \ell' \in T \quad \mathbf{x}''_p \in \mathbf{X}_p \qquad g_\tau(\mathbf{x}) \quad \mathbf{x}' = f_\tau(\mathbf{x}[X_p \leftarrow \mathbf{x}''_p])}{(\ell, \mathbf{x}) \underset{B}{\xrightarrow{p(\mathbf{x}''_p)}} (\ell', \mathbf{x}')}$$

That is, $(\ell', \mathbf{x}')$ is a successor of $(\ell, \mathbf{x})$ labelled by $p(\mathbf{x}''_p)$ iff (1) $\tau = \ell \xrightarrow{p} \ell'$ is a transition of T, (2) the guard g_τ holds on the current valuation $\mathbf{x}$, (3) $\mathbf{x}''_p$ is a valuation of exported variables X_p and (4) $\mathbf{x}' = f_\tau(\mathbf{x}[X_p \leftarrow \mathbf{x}''_p])$ meaning that, the new valuation $\mathbf{x}'$ is obtained by applying f_τ on $\mathbf{x}$ previously modified according to $\mathbf{x}''_p$. Whenever a p-labelled successor exist in a state, we say that p is *enabled* in that state.

2.2 Composite Components

Composite components are obtained by composing an existing set of atomic components $\{B_i = (L_i, X_i, P_i, T_i)\}_{i=1,n}$ trough specific composition operators. We consider that atomic components have pairwise disjoint sets of states, ports, and variables i.e., for any two $i \neq j$ from $\{1..n\}$, we have $L_i \cap L_j = \emptyset$, $P_i \cap P_j = \emptyset$, and $X_i \cap X_j = \emptyset$. We denote $P = \bigcup_{i=1}^n P_i$ the set of all the ports, $L = \bigcup_{i=1}^n L_i$ the set of all locations, and $X = \bigcup_{i=1}^n X_i$ the set of all variables.

Definition 3 (interaction) *An* interaction *a between atomic components is a triple (P_a, G_a, F_a), where $P_a \subseteq P$ is a set of ports, G_a is a guard, and F_a is an update function. By definition, P_a uses at most one port of every component, that is, $|P_i \cap P_a| \leq 1$ for all $i \in \{1..n\}$. Therefore, we simply denote $P_a = \{p_i\}_{i \in I}$, where $I \subseteq \{1..n\}$ contains the indices of the components involved in a and for all $i \in I, p_i \in P_i$. G_a and F_a are both defined on the variables exported by the ports in P_a (i.e., $\bigcup_{p \in P_a} X_p$).*

Definition 4 (composite component) *A composite component $C = \gamma(B_1, \ldots, B_n)$ is obtained by applying a set of interactions γ to a set of atomic components $B_1, \ldots B_n$.*

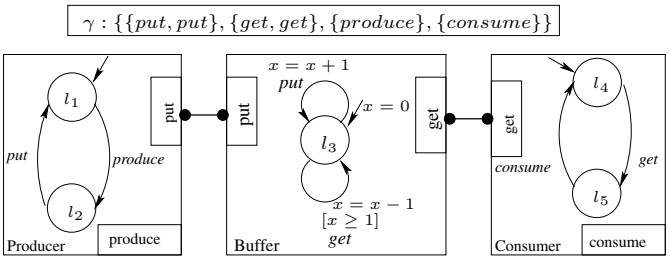

Figure 2: Producer-Buffer-Consumer example

Example 2 *Figure 2 presents a classical producer-buffer-consumer example modeled in* secBIP. *It consists of three atomic components, namely Producer, Buffer and Consumer. The Buffer is a shared memory placeholder, which is accessible by both the Producer and the Consumer. It holds into the local variable x the number of items available. The Buffer interacts with the Producer (res. Consumer) on the put (resp. get) interaction. On the put interaction, an item is added to the Buffer and x is incremented. On the get interaction, the Consumer removes an item from the Buffer, if at least one exists (the guard $[x \geq 1]$), and x is decremented. Finally, the transitions labeled produce and consume do not require synchronization - they are executed alone (on singleton port interactions) by the respective components.*

Definition 5 (composite component semantics) *Let $C = \gamma(B_1, \ldots, B_n)$ be a composite component. Let $B_i = (L_i, X_i, P_i, T_i)$ and $\text{LTS}(B_i) = (Q_i, \Sigma_i, \xrightarrow[B_i]{})$ their semantics, for all $i = 1, n$. The semantics of C is the labelled transition system $\text{LTS}(C) = (Q_C, \Sigma_C, \xrightarrow[C]{})$ where the set of states $Q_C = \otimes_{i=1}^n Q_i$, the set of labels $\Sigma_C = \gamma$ and the set of labelled transitions $\xrightarrow[C]{}$ is defined by the rule:*

$$\textsc{Comp}\ \frac{\begin{array}{c} a = (\{p_i\}_{i \in I}, G_a, F_a) \in \gamma \qquad G_a(\{\mathbf{x}_{p_i}\}_{i \in I}) \quad \{\mathbf{x}''_{p_i}\}_{i \in I} = F_a(\{\mathbf{x}_{p_i}\}_{i \in I}) \\ \forall i \in I.\ (\ell_i, \mathbf{x}_i) \xrightarrow[B_i]{p_i(\mathbf{x}''_{p_i})} (\ell'_i, \mathbf{x}'_i) \qquad \forall i \notin I.\ (\ell_i, \mathbf{x}_i) = (\ell'_i, \mathbf{x}'_i) \end{array}}{((\ell_1, \mathbf{x}_1), \ldots, (\ell_n, \mathbf{x}_n)) \xrightarrow[C]{a} ((\ell'_1, \mathbf{x}'_1), \ldots, (\ell'_n, \mathbf{x}'_n))}$$

For each $i \in I$, $\mathbf{x}_{p_i}$ above denotes the valuation $\mathbf{x}_i$ restricted to variables of X_{p_i}.

The rule expresses that a composite component $C = \gamma(B_1, \ldots, B_n)$ can execute an interaction $a \in \gamma$ *enabled* in state $((\ell_1, \mathbf{x}_1), \ldots, (\ell_n, \mathbf{x}_n))$, iff (1) for each $p_i \in P_a$, the corresponding atomic component B_i can execute a transition labelled by p_i, and (2) the guard G_a of the interaction holds on the current valuation of variables exported on ports participating in a. Execution of interaction a triggers first the update function F_a which modifies variables exported by ports $p_i \in P_a$. The new values obtained, encoded in the valuation $\mathbf{x}''_{p_i}$, are then used by the components' transitions. The states of components that do not participate in the interaction remain unchanged.
Any finite sequences of interactions $w = a_1 ... a_k \in \gamma^*$ executable by the composite component starting at some given initial state q_0 is named a trace. The set of all traces w from state q_0 is denoted by $\text{TRACES}(C, q_0)$.

3 Information Flow Security

We explore information flow policies [11, 10, 17] with focus on the non-interference property. In order to track information we adopt the classification technique and we define a classification policy where we annotate the information by assigning security levels to different parts of *secBIP* model (data variables, ports and interactions). The policy describes how information can flow from one classification with respect to the other.
As an example, we can classify public information as a Low (L) security level and secret (confidential) information as High (H) security level. Intuitively High security level is more restrictive than Low security level and we denote it by $L \subseteq H$. In general, security levels are elements of a security domain, defined as follows:

Definition 6 (security domain) *A security domain is a lattice of the form $\langle S, \subseteq, \cup, \cap \rangle$ where:*

- *S is a finite set of security levels.*
- *$\subseteq$ is a partial order "can flow to" on S that indicates that information can flow from one security level to an equal or a more restrictive one.*
- *$\cup$ is a "join" operator for any two levels in S and that represents the upper bound of them.*
- *$\cap$ is a "meet" operator for any two levels in S and that represents the lower bound of them.*

As an example we consider the set $S = \{L, M_1, M_2, H\}$ of security levels that are governed by the "can flow to" relation $L \subseteq M_1$, $L \subseteq M_2$, $M_1 \subseteq H$ and $M_2 \subseteq H$. M_1 and M_2 are incomparable and we note $M_1 \nsubseteq M_2$ and $M_1 \nsupseteq M_2$. This security domain is graphically illustrated in figure 3.

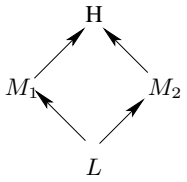

Figure 3: Security domain

Let $C = \gamma(B_1, \ldots B_n)$ be a composite component, fixed. Let X (resp. P) be the set of all variables (resp. ports) defined in all atomic components $(B_i)_{i=1,n}$.
Let $\langle S, \subseteq, \cup, \cap \rangle$ be a security domain, fixed.

Definition 7 (security assignment) *A security assignment for component C is a mapping $\sigma : X \cup P \cup \gamma \rightarrow S$ that associates security levels to variables, ports and interactions such that, moreover, the security levels of ports matches the security levels of interactions, that is, for all $a \in \gamma$ and for all $p \in P$ it holds $\sigma(p) = \sigma(a)$.*

In atomic components, the security levels considered for ports and variables allow to track intra-component information flows and control the intermediate computation steps. Moreover, inter-components communication, that is, interactions with data exchange, are tracked by the security levels assigned to interactions.
Let σ be a security assignment for C, fixed.
For a security level $s \in S$, we define $\gamma\downarrow_s^\sigma$ the restriction of γ to interactions with security level at most s that is formally, $\gamma\downarrow_s^\sigma = \{a \in \gamma \mid \sigma(a) \subseteq s\}$.
For a security level $s \in S$, we define $w|_s^\sigma$ the projection of a trace $w \in \gamma^*$ to interactions with security level lower or equal to s. Formally, the projection is recursively defined on traces as $\epsilon|_s^\sigma = \epsilon$, $(aw)|_s^\sigma = a(w|_s^\sigma)$ if $\sigma(a) \subseteq s$ and $(aw)|_s^\sigma = w|_s^\sigma$ if $\sigma(a) \not\subseteq s$. The projection operator $|_s^\sigma$ is naturally lifted to sets of traces W by taking $W|_s^\sigma = \{w|_s^\sigma \mid w \in W\}$.
For a security level $s \in S$, we define the equivalence $\approx_s^\sigma$ on states of C. Two states q_1, q_2 are equivalent, denoted by $q_1 \approx_s^\sigma q_2$ iff (1) they coincide on variables having security levels at most s and (2) they coincide on control locations having outgoing transitions labeled with ports with security level at most s.
We are now ready to define the two notions of non-interference.

Definition 8 (event non-interference) *The security assignment σ ensures event non-interference of $\gamma(B_1, \ldots, B_n)$ at security level s iff,*

$$\forall q_0 \in Q_C^0 : \quad \text{TRACES}(\gamma(B_1, \ldots, B_n), q_0)|_s^\sigma = \text{TRACES}((\gamma\downarrow_s^\sigma)(B_1, \ldots, B_n), q_0)$$

Event non-interference ensures isolation/security at interaction level. The definition excludes the possibility to gain any relevant information about the occurrences of interactions (events) with strictly greater (or incomparable) levels than s, from the exclusive observation of occurrences of interactions with levels lower or equal to s. That is, an external observer is not able to distinguish between the case where such higher interactions are not observable on execution traces and the case these interactions have been actually statically removed from the composition. This definition is very close to Rushby's [23] definition for transitive non-interference. But, let us remark that event non-interference is not concerned about the protection of data.

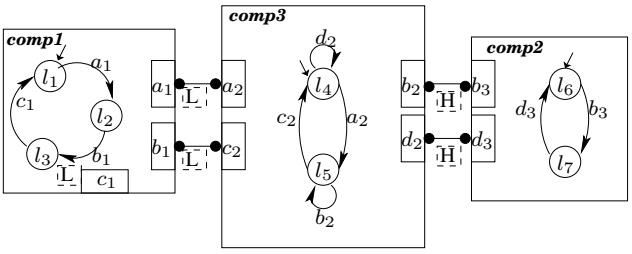

Figure 4: Example for event non-interference.

Example 3 *Figure 4 presents a simple illustrative example for event non-interference. The model consists of three atomic components $comp_{i,i=1,2,3}$. Different security levels have been assigned to ports and interactions: $comp_1$ is a low security component, $comp_2$ is a high security component, and $comp_3$ is mixed security component. The security levels are represented by dashed squares related to interactions, internal ports and variables. As a convention, we apply high (H) level for secret data and interactions and low(L) level for public ones. The set of traces is represented by the automaton in figure 5 (a). The set of projected execution traces at security level L is represented by the automaton depicted in figure 5 (b). This set is equal to the set of traces obtained by restricted composition, that is, using interaction with security level at most L and depicted in figure 5 (c). Therefore, this example satisfies the event non-interference condition at level L.*

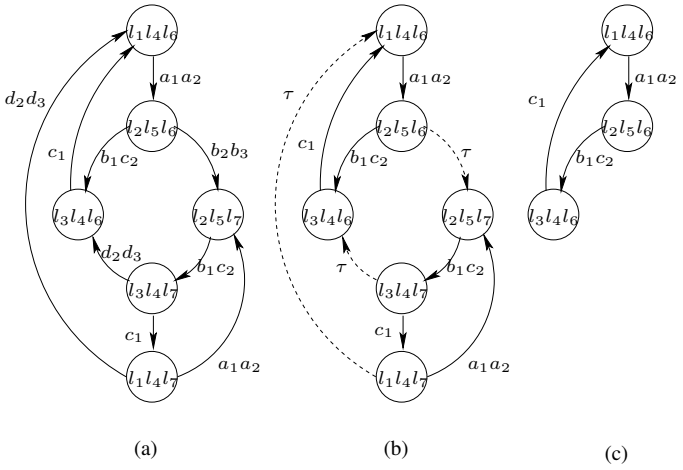

Figure 5: Sets of traces represented as automata.

Definition 9 (data non-interference) *The security assignment σ ensures data non-interference of $C = \gamma(B_1, \ldots, B_n)$ at security level s iff,*

$$\forall q_1, q_2 \in Q_C^0 : \; q_1 \approx_s^\sigma q_2 \Rightarrow$$
$$\forall w_1 \in \text{TRACES}(C, q_1), w_2 \in \text{TRACES}(C, q_2) : \; w_1|_s^\sigma = w_2|_s^\sigma \Rightarrow$$
$$\forall q_1', q_2' \in Q_C : \; q_1 \xrightarrow[C]{w_1} q_1' \wedge q_2 \xrightarrow[C]{w_2} q_2' \Rightarrow q_1' \approx_s^\sigma q_2'$$

Data non-interference provides isolation/security at data level. The definition ensures that, all states reached from initially indistinguishable states at security level s, by execution of arbitrary but identical traces whenever projected at level s, are also indistinguishable at level s. That means that observation of all variables and interactions with level s or lower excludes any gain of relevant information about variables at higher (or incomparable) level than s. Compared to event non-interference, data non-interference is a stronger property that considers the system's global states (local states and valuation of variables) and focus on their equivalence along identical execution traces (at some security level).

Example 4 *Figure 6 presents an extension with data variables of the previous example from figure 4. We consider the following two traces $w_1 = \langle a_1a_2, b_2b_3, c_2b_1, d_2d_3, c_1, a_2a_1\rangle$ and $w_2 = \langle a_1a_2, b_2b_3, c_2b_1, c_1, a_2a_1\rangle$ that start from the initial state $((l_1, u = 0, v = 0), (l_4, x = 0, y = 0), (l_6, z = 0, w = 0))$. Although the projected traces at level L are equal, that is, $w_1|_L^\sigma = w_2|_L^\sigma = \langle a_1a_2, c_2b_1, c_1, a_1a_2\rangle$, the reached states by w_1 and w_2 are different, respectively $((l_2, u = 4, v = 2), (l_5, x = 3, y = 2), (l_6, z = 1, w = 1))$ and $((l_2, u = 4, v = 2), (l_5, x = 2, y = 2), (l_7, z = 1, w = 0))$ and moreover non-equivalent at low level L. Hence, this example is not data non-interferent at level L.*

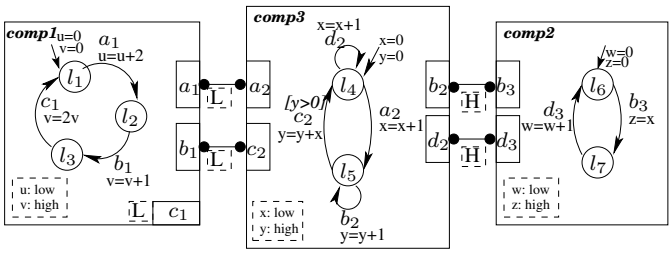

Figure 6: Example for data non-interference.

Definition 10 (secure component) *A security assignment σ is secure for a component $\gamma(B_1, \ldots, B_n)$ iff it ensures both event and data non-interference, at all security levels $s \in S$.*

4 Verification

The verification technique of non-interference proposed for *secBIP* models is using the so-called unwinding conditions. This technique was first introduced by Goguen and Meseguer for the verification of transitive non-interference for deterministic systems [17]. The unwinding approach reduces the verification of information flow security to the existence of certain unwinding relation. This relation is usually an equivalence relation that respects some additional properties on atomic execution steps, which are shown sufficient to imply non-interference. In the case of *secBIP*, the additional properties are formulated in terms of individual interactions/events and therefore easier to handle.
Let $C = \gamma(B_1, \ldots, B_n)$ be a composite component and let σ be a security assignment for C.

Definition 11 (unwinding relation) *An equivalence $\sim_s$ on states of C is called an unwinding relation for σ at security level s iff the two following conditions hold:*

1. *local consistency*
   $\forall q, q' \in Q_C : \forall a \in \gamma : q \xrightarrow[C]{a} q' \Rightarrow \sigma(a) \subseteq s \vee q \sim_s q'$

2. *output and step consistency*
   $\forall q_1, q_2, q_1' \in Q_C : \forall a \in \gamma :$
   $q_1 \sim_s q_2 \wedge q_1 \xrightarrow[C]{a} q_1' \wedge \sigma(a) \subseteq s \Rightarrow$
   $\exists q_2' \in Q_C : q_2 \xrightarrow[C]{a} q_2' \wedge$
   $\forall q_2' \in Q_C : q_2 \xrightarrow[C]{a} q_2' \Rightarrow q_1' \sim_s q_2'$

The existence of unwinding relations is tightly related to non-interference. The following two theorems formalize this relation for the two types of non-interference defined. Let C be a composite component and σ a security assignment.

4.1 Event Non-Interference

In this section we give a formal theorem proof of event non-interference based on the unwinding technique.

Theorem 1 (event non-interference) *If an unwinding relation $\sim_s$ exists for the security assignment σ at security level s, then σ ensures event non-interference of C at level s.*

Proof: We shall prove $\text{TRACES}(\gamma(B_1, \ldots, B_n), q_0)|_s^\sigma = \text{TRACES}((\gamma \downarrow_s^\sigma)(B_1, \ldots, B_n), q_0)$ by double inclusion. "$\supseteq$" inclusion: Independently of the unwinding relation, by using elementary set properties it holds that $\text{TRACES}((\gamma \downarrow_s^\sigma)(B_1, \ldots, B_n), q_0) = \text{TRACES}((\gamma \downarrow_s^\sigma)(B_1, \ldots, B_n), q_0)|_s^\sigma \subseteq \text{TRACES}(\gamma(B_1, \ldots, B_n), q_0)|_s^\sigma$. "$\subseteq$" inclusion: This direction is an immediate consequence of Lemma 1 hereafter. It states that for every trace w in $\text{TRACES}(\gamma(B_1, \ldots, B_n), q_0)$ its projection $w|_s^\sigma$ is also a valid trace in $\text{TRACES}(\gamma(B_1, \ldots, B_n), q_0)$. But, this also means that $w|_s^\sigma$ is a valid trace in $\text{TRACES}((\gamma \downarrow_s^\sigma)(B_1, \ldots, B_n), q_0)$ which proves the result. □

Lemma 1 *In the conditions of theorem 1, for every trace w in $\text{TRACES}(\gamma(B_1, \ldots, B_n), q_0)$, for every state q such that $q_0 \xrightarrow[C]{w} q$, the projected trace $w|_s^\sigma$ is also a valid trace in $\text{TRACES}(\gamma(B_1, \ldots, B_n), q_0)$ and moreover, for every state q' such that $q_0 \xrightarrow[C]{w|_s^\sigma} q'$ it holds $q \sim_s q'$.*

Proof: The lemma is proved by induction on the length of the trace w. For the empty trace $w = \epsilon$ verification is trivial: $\sim_s$ holds for the initial state $q_0 \sim_s q_0$ and $\epsilon = \epsilon|_s^\sigma$. By induction hypothesis, let assume the property holds for traces of length n. We shall prove the property for traces of length $n+1$. Let $w' = wa$ be an arbitrary trace of length $n+1$, let w be its prefix (trace) of length n and let a be the last interaction. Consider states q, q_1 such that $q_0 \xrightarrow[C]{w} q \xrightarrow[C]{a} q_1$. By the induction hypothesis we know that $w|_s^\sigma$ is a valid trace and for all states q' such that $q_0 \xrightarrow[C]{w|_s^\sigma} q'$ it holds $q \sim_s q'$. We distinguish two cases, depending on the security level of a:

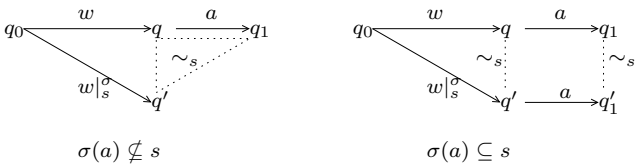

Figure 7: Proof illustration for lemma 1

- $\sigma(a) \not\subseteq s$: In this case, $w'|_s^\sigma = w|_s^\sigma$ hence, $w'|_s^\sigma$ is a valid trace as well, reaching the same states q'. Moreover, since a is invisible for s, the unwinding condition (1) ensures that $q \sim_s q_1$. By transitivity, this implies that $q_1 \sim_s q'$, which proves the result.
- $\sigma(a) \subseteq s$: In this case, $w'|_s^\sigma = w|_s^\sigma a$. From the unwinding condition (2), since $q \sim_s q'$ and a is visible and enabled in q then, a must also be enabled in q'. Therefore, $w|_s^\sigma$ can be extended with a from state q' to some q_1' hence, $w'|_s^\sigma$ is indeed a valid trace. Moreover, since $q \sim_s q'$ the unwinding condition (2) ensures also that $q_1 \sim_s q_1'$, which proves the result.

□

4.2 Data Non-Interference

Based on the unwinding technique, we formally give a verification of data non-interference.

Let $C = \gamma(B_1, \ldots, B_n)$ be a composite component and let σ be a security assignment for C.

Theorem 2 (data non-interference) *If the equivalence relation $\approx_s^\sigma$ is also an unwinding relation for the security assignment σ at security level s, then σ ensures data non-interference of C at level s.*

Proof: Let us consider two equivalent states $q_1 \approx_s^\sigma q_2$. The *first condition* for data non-interference requires that, for any trace w_1 from q_1 there exists a trace w_2 from q_2 having the same projection at level s, that is, $w_1|_s^\sigma = w_2|_s^\sigma$.
We shall prove a slightly stronger property, namely, the trace w_2 can be chosen such that, the successors q_1' and q_2' of respectively q_1 by w_1 and q_2 by w_2 are moreover equivalent, that is, $q_1' \approx_s^\sigma q_2'$. The proof is by induction on the length of the trace w_1. *The base case*: for the empty trace $w_1 = \epsilon$ we take equally $w_2 = \epsilon$ we immediately have $q_1' = q_1 \approx_s^\sigma q_2 = q_2'$. *The induction step:* we assume, by induction hypothesis that the property holds for all traces w_1 such that $|w_1| \leq n$ and we shall prove it for all traces w_1' such that $|w_1'| = n+1$. Let a be the last interaction executed in w_1', that is, $w_1' = w_1 a$ with $|w_1| = n$. Let q_1'' be the state reached from q_1 by w_1. From the induction hypothesis, there exists a trace w_2 that leads q_2 into q_2'' such that $w_1|_s^\sigma = w_2|_s^\sigma$ and moreover $q_1'' \approx_s^\sigma q_2''$. We distinguish two cases, depending on the security level of a:

- $\sigma(a) \not\subseteq s$: since $\approx_s^\sigma$ is unwinding and $q_1'' \xrightarrow[C]{a} q_1'$ it follows that $q_1'' \approx_s^\sigma q_1'$. In this case, we take $w_2' = w_2$ and $q_2' = q_2''$ which ensures both $w_1'|_s^\sigma = w_1|_s^\sigma = w_2|_s^\sigma = w_2'|_s^\sigma$ and $q_1' \approx_s^\sigma q_1'' \approx q_2'' = q_2'$.

- $\sigma(a) \subseteq s$: since $\approx_s^\sigma$ is unwinding and $q_1'' \approx_s^\sigma q_2''$ and $q_1'' \xrightarrow[C]{a} q_1'$ there must exists q_2' such that $q_2'' \xrightarrow[C]{a} q_2'$ and moreover, for any such choice $q_1' \approx_s^\sigma q_2'$. Hence, in this case, the trace $w_2' = w_2 a$ executed from q_2 and leading to q_2' satisfies our property, namely $w_1'|_s^\sigma = w_1|_s^\sigma a = w_2|_s^\sigma a = w_2'|_s^\sigma$ and $q_1' \approx_s^\sigma q_2'$.

The *second condition* for data non-interference requires that, for any traces w_1 and w_2 with equal projection on security level s, that is $w_1|_s^\sigma = w_2|_s^\sigma$, any successor states q_1' and q_2' of respectively q_1 by w_1 and q_2 by w_2 are also equivalent at level s. This property is proved also by induction on $|w_1| + |w_2|$, that is, on the sum of the lengths of traces w_1, w_2. *The base case*: for empty traces $w_1 = w_2 = \epsilon$ we have that $q_1' = q_1$ and $q_2' = q_2$ and hence trivially $q_1' \approx_s^\sigma q_2'$. *The induction step*: we assume, by induction hypothesis that the property holds for any traces w_1, w_2 such that $|w_1| + |w_2| \leq n$ and we shall prove it for all traces w_1', w_2' such that $|w_1'| + |w_2'| = n + 1$. We distinguish two cases, depending on the security levels of the last interactions occurring in w_1' and w_2'.

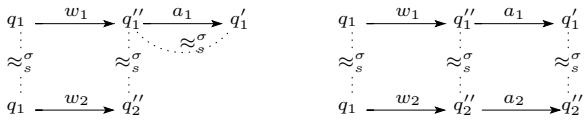

Figure 8: Proof illustration for theorem 2

- at least one of the last interactions in w_1' or w_2' has a security level not lower or equal to s. W.l.o.g, consider that indeed $w_1' = w_1 a_1$ and $\sigma(a_1) \not\subseteq s$. This situation is depicted in figure 8, (left).

  Let q_1'' be the state reached from q_1 after w_1. Since $w_1'|_s^\sigma = w_2'|_s^\sigma$ and $\sigma(a_1) \not\subseteq s$ it follows that $w_1|_s^\sigma = w_1'|_s^\sigma = w_2'|_s^\sigma$. The induction hypothesis holds then for w_1 and w_2' because $|w_1| + |w_2'| = n - 1$ and hence we have that $q_1'' \approx_s^\sigma q_2'$. Moreover, q_1' is a successor of q_1'' by interaction a_1. Since the security level of a_1 is not lower or equal to s, and $\approx_s^\sigma$ is an unwinding relation at level s, it follows from the local consistency condition that $q_1'' \approx_s^\sigma q_1$. Then, by transitivity of $\approx_s^\sigma$ we obtain that $q_1' \approx_s^\sigma q_2'$.

- the last interactions of both traces w_1' and w_2' have security level lower or equal to s. That is, consider $w_1' = w_1 a_1$ and $w_2' = w_2 a_2$ with $\sigma(a_1) \subseteq s, \sigma(a_2) \subseteq s$. This situation is depicted in figure 8, (right).

  Let q_1'' and q_2'' be the states reached respectively from q_1 by w_1 and from q_2 by w_2. Since $\sigma(a_1) \subseteq s$,$\sigma(a_2) \subseteq s$ we have $w_1'|_s^\sigma = w_1|_s^\sigma a_1$, $w_2'|_s^\sigma = w_2|_s^\sigma a_2$. From the hypothesis, $w_1'|_s^\sigma = w_2'|_s^\sigma$, it follows that both $a_1 = a_2$ and $w_1|_s^\sigma = w_2|_s^\sigma$. Therefore, the induction hypothesis can be applied for traces w_1, w_2 because $|w_1| + |w_2| = n - 2$ and hence, we obtain $q_1'' \approx_s^\sigma q_2''$. But now, q_1' and q_2' are immediate successors of two equivalent states q_1'' and q_2'' by executing some interaction $a = a_1 = a_2$, having security level lower or equal to s. Since, $\approx_s^\sigma$ is an unwinding relation at level s, it follows from the step consistency condition that successors states q_1' and q_2' are also equivalent at level s, hence, $q_1' \approx_s^\sigma q_2'$.

□

4.3 Sufficient Conditions

The two theorems above can be used to derive a practical verification method of non-interference using unwinding. We provide hereafter sufficient syntactic conditions ensuring that indeed the unwinding relations $\sim_s$ and $\approx_s$ exist on the system states. These conditions aim to efficiently simplify the verification and reduce it to local constrains check on both transitions (inter-component verification) and interactions (intra-component verification). Especially, they give an easy way to automate the verification.

Definition 12 (security conditions) *Let $C = \gamma(B_1, \ldots, B_n)$ be a composite component and let σ be a security assignment. We say that C satisfies the security conditions for security assignment σ iff:*

(i) the security assignment of ports, in every atomic component B_i is locally consistent, that is:

- *for every pair of causal transitions:*

$$\forall \tau_1, \tau_2 \in T_i : \ \tau_1 = \ell_1 \xrightarrow{p_1} \ell_2, \ \tau_2 = \ell_2 \xrightarrow{p_2} \ell_3 \ \Rightarrow \\ \ell_1 \neq \ell_2 \ \Rightarrow \ \sigma(p_1) \subseteq \sigma(p_2)$$

- *for every pair of conflicting transitions:*

$$\forall \tau_1, \tau_2 \in T_i : \ \tau_1 = \ell_1 \xrightarrow{p_1} \ell_2, \ \tau_2 = \ell_1 \xrightarrow{p_2} \ell_3 \ \Rightarrow \\ \ell_1 \neq \ell_2 \ \Rightarrow \ \sigma(p_1) \subseteq \sigma(p_2)$$

(ii) all assignments $x := e$ occurring in transitions within atomic components and interactions are sequential consistent, in the classical sense:

$$\forall y \in use(e) : \ \sigma(y) \subseteq \sigma(x)$$

(iii) variables are consistently used and assigned in transitions and interactions, that is,

$$\forall \tau \in \cup_{i=1}^{n} T_i \ \ \forall x, y \in X \ : x \in def(f_\tau), y \in use(g_\tau) \ \Rightarrow \\ \sigma(y) \subseteq \sigma(p_\tau) \subseteq \sigma(x) \\ \forall a \in \gamma \ \ \forall x, y \in X \ : x \in def(F_a), y \in use(G_a) \ \Rightarrow \\ \sigma(y) \subseteq \sigma(a) \subseteq \sigma(x)$$

(iv) all atomic components B_i are port deterministic:

$$\forall \tau_1, \tau_2 \in T_i : \ \tau_1 = \ell_1 \xrightarrow{p} \ell_2, \tau_2 = \ell_1 \xrightarrow{p} \ell_3 \ \Rightarrow \\ (g_{\tau_1} \wedge g_{\tau_2}) \textit{ is unsatisfiable}$$

The first family of conditions (i) is similar to Accorsi's conditions [4] for excluding causal and conflicting places for Petri net transitions having different security levels. Similar conditions have been considered in [13, 15] and lead to more specific definitions of non-interferences and bisimulations on annotated Petri nets. The second condition (*ii*) represents the classical condition needed to avoid information leakage in sequential assignments. The third condition (*iii*) tackles covert channels issues. Indeed, (*iii*) enforces the security levels of the data flows which have to be consistent with security levels of the ports or interactions (e.g., no low level data has to be updated on a high level port or interaction). Such that, observations of public data would not reveal any secret information. Finally, conditions (*iv*) enforces deterministic behavior on atomic components.
The relation between the syntactic security conditions and the unwinding relations is precisely captured by the following theorem.

Theorem 3 (unwinding theorem) *Whenever the security conditions hold, the equivalence relation $\approx_s^\sigma$ is an unwinding relation for the security assignment σ, at all security level s.*

The following corollary is the immediate consequence of theorems 1, 2 and 3.

Corollary 1 *Whenever the security conditions hold, the security assignment σ is secure for the component C.*

5 Application

We illustrate the *secBIP* framework to handle information flow security issues for a typical example, the web service reservation system introduced [16]. A businessman, living in France, plans to go to Berlin for a private and secret mission. To organize his travel, he uses an intelligent web service who contacts two travel agencies: The first agency, *AgencyA*, arranges flights in Europe and the second agency, *AgencyB*, arranges

flights exclusively to Germany. The reservation service obtains in return specific flight information and their corresponding prices and chooses the flight that is more convenient for him.

In this example, there are two types of interference that can occur, (1) data-interference since learning the flight price may reveal the flight destination and (2) event interference, since observing the interaction with *AgencyB* can reveal the destination as well. Thus, to keep the mission private, the flight prices and interactions with *AgencyB* have to be kept confidential.

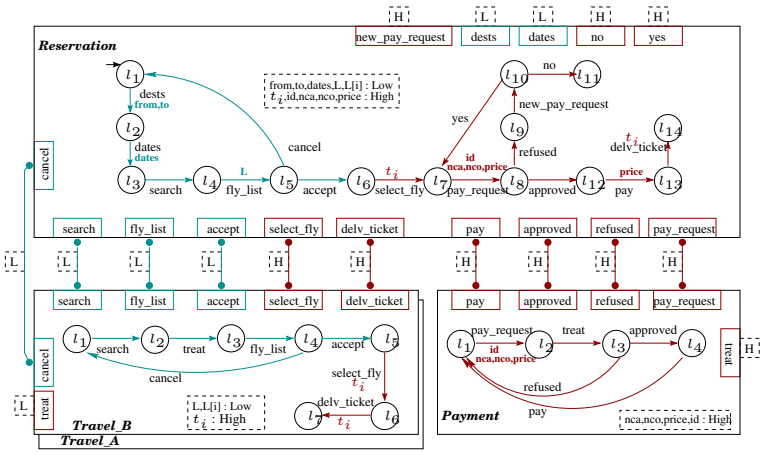

Figure 9: Reservation Web Service composition

The modeling of the system using *secBIP* involves two main distinct steps: first, functional requirements modeling reflecting the system behavior, and second, security annotations enforcing the desired security policy. The model of the system has four components denoted: *Travel_A* and *Travel_B* who are instances from the same component and correspond respectively to AgencyA and AgencyB, and components *Reservation* and *Payment*. To avoid figure 9 cluttering, we did not represent the interactions with *Travel_A* component. Search parameters are supplied by a user through the *Reservation* component ports *dests* and *dates* to which we associate respectively variables ($from$, to) and $dates$. Next, through search interaction, *Reservation* component contacts *Travel_B* component to search for available flights and obtains in return a list L of specific flights with their corresponding prices. Thereafter, *Reservation* component selects a ticket t_i from the list L and requests the *Payment* component to perform the payment.

All the search parameters $from$, to, $dates$, as well as the flights list L are set to low since users are not identified while sending these queries. Other sensitive data like the selected flight t_i, the price variable p and the payment parameters (identity id, credit card variable cna and code number cno) are set to high. Internal ports *dests* and *dates* as well as *search*, *fly_list*, *accept* interactions are set to low since these interactions (events) do not reveal any information about the client private trip. However, the $select\_fly$ interaction must be set to high since the observation of the selection event from *AgencyB* allow to deduce deduce the client destination. In the case of a selected flight from *AgencyA*, the $select\_fly$ interaction could be set to low since, in this case, the destination could not be deduced just from the event occurrence.

We recall that any system can be proven non-interferent iff it satisfies the syntactic security conditions from definition 12. Indeed, these conditions hold for the system model depicted in Figure 9. In particular, it can be easily checked that all assignments occurring in transitions within atomic component as well as within interactions are sequential consistent. For example, at the $select\_fly$ interaction we assign a low level security item from the flight list L to a high security level variable ti, formally $t_i = L[i]$. Besides, the security levels assignments to ports exclude inconsistencies due to causal and conflicting transitions, in all atomic components.

6 Discussion and Related work

Non-interference properties have been already studied using different model-based approaches. Recently, [27] adapted an MDS method for handling information flow security using UML sequence diagrams. Additionally, Petri-nets have been extensively used for system modeling and information flow security verifications tools such as InDico [5] have been developed. A component-based model has been proposed in [3] and used to study implementation issues of secure information flows. Our presented work on *secBIP* is different in several respects. First, *secBIP* is a formal framework. Unlike UML, system's runtime behavior is always meaningfully defined and can be formally analyzed. Moreover, *secBIP* provides a system construction methodology for complex systems. Indeed, big systems are functionally decomposed into multiple sub-components communicating through well-defined interactions. Such a structural decomposition of the system is usually not available on Petri-nets models.
Furthermore, *secBIP* handles both event and data-flow non-interference, in a single semantic model. To the best of our knowledge, these properties have never been jointly considered for component-based models. Nevertheless, the need to consider together event and data flow non-interference has been already tackled in the existing literature. The bottom line is that preserving the safety of data flow in a system does not necessarily preserve safe observability on system's public behavior (i.e., secret/private executions may have an observable impact on system public events). The issue has been recently considered in [4], for data leaks and information leaks in business processes based on system's data-flows and work-flows. Also, [6] showed that formal verification of the system's event behavior is not sufficient to guarantee specific data properties. Also, [14] attempted to fill the gap between respectively language-based and process calculus-based information security and make an explicit distinction between preventing the data leakage through the execution of programs and preventing secret events from being revealed in inter-process communications.
Compared to Security-typed programming languages [2, 32] and operating systems [18, 33, 12] enforcing information flow control, *secBIP* is a component-based modeling approach where non-interference is verified at a more abstract level. Thus, *secBIP* can be written in different languages independently from a specific platform.
Finally, it is worth mentioning that a lot of classical approaches fall short to handle information flow security [31] for real systems. For *secBIP* we privilege a very pragmatic approach and provide simple (syntactic) sufficient conditions allowing to automate the verification of non-interference. These conditions allow to eliminate a significant amount of security leakages, especially covert channels, independently from system language or the execution platform. However, these conditions can be very restrictive in some cases. The system designer may be interested to relax the non-interference properties, however, these extensions are out of the scope of the current paper.

7 Conclusion and future work

We present a MDS framework to secure component-based systems. We formally define two types of non-interference, respectively event and data non-interference. We provide a set of sufficient syntactic conditions formulated to simplify non-interference verification. These conditions are extensions of security typed language rules applied to our model. The use of our framework has been demonstrated to secure a web service application.
This work is currently being extended in two directions. First, we are investigating additional security conditions allowing to relax the non-interference property and control when and where downgrading can occur. Second, we are working towards the implementation of a complete design flow for secure systems based on *secBIP*. As a first step, we shall implement the verification method presented for annotated *secBIP* models. Then, use these models for generation of secure implementations, that is, executable code where the security properties are enforced by construction, at the generation time.

References

[1] http://www-verimag.imag.fr/BIP-Tools,93.html?lang=en/. 2

[2] http://www.cs.cornell.edu/jif/. 6

[3] T. Abdellatif, L. Sfaxi, R. Robbana, and Y. Lakhnech. Automating information flow control in component-based distributed systems. In *CBSE*, pages 73–82, 2011. 6

[4] R. Accorsi and A. Lehmann. Automatic information flow analysis of business process models. In *Proceedings of the 10th international conference on Business Process Management*, BPM'12, pages 172–187. Springer-Verlag, 2012. 4.3, 6

[5] R. Accorsi, C. Wonnemann, and S. Dochow. Swat: A security workflow analysis toolkit for reliably secure process-aware information systems. In *ARES*, pages 692–697, 2011. 6

[6] C. Bartolini, A. Bertolino, E. Marchetti, and I. Parissis. Architecting dependable systems v. chapter Data Flow-Based Validation of Web Services Compositions: Perspectives and Examples, pages 298–325. Springer-Verlag, 2008. 6

[7] D. Basin, J. Doser, and T. Lodderstedt. Model driven security: from uml models to access control infrastructures. *ACM Transactions on Software Engineering and Methodology*, 15, 2006. 1

[8] A. Basu, S. Bensalem, M. Bozga, J. Combaz, M. Jaber, T.-H. Nguyen, and J. Sifakis. Rigorous component-based design using the BIP framework. *IEEE Software, Special Edition – Software Components beyond Programming – from Routines to Services*, 28(3):41–48, 2011. 1, 2

[9] A. Basu, M. Bozga, and J. Sifakis. Modeling Heterogeneous Real-time Systems in BIP. In *Proceedings of SEFM'06*, pages 3–12. IEEE Computer Society Press, 2006. 1, 2

[10] E. D. Bell and J. L. La Padula. Secure computer system: Unified exposition and multics interpretation, 1976. 3

[11] D. E. Denning and P. J. Denning. Certification of programs for secure information flow. *Commun. ACM*, pages 504–513, 1977. 3

[12] P. Efstathopoulos, M. Krohn, S. VanDeBogart, C. Frey, D. Ziegler, E. Kohler, D. Mazières, F. Kaashoek, and R. Morris. Labels and event processes in the asbestos operating system. *SIGOPS Oper. Syst. Rev.*, 39(5):17–30, Oct. 2005. 6

[13] R. Focardi and R. Gorrieri. Classification of security properties (part i: Information flow). In *Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, FOSAD '00, pages 331–396, London, UK, UK, 2001. 4.3

[14] R. Focardi, S. Rossi, and A. Sabelfeld. Bridging language-based and process calculi security. In *In Proc. of Foundations of Software Science and Computation Structures (FOSSACS'05), volume 3441 of LNCS*, pages 299–315. Springer-Verlag, 2005. 6

[15] S. Frau, R. Gorrieri, and C. Ferigato. Formal aspects in security and trust. chapter Petri Net Security Checker: Structural Non-interference at Work, pages 210–225. Berlin, Heidelberg, 2009. 4.3

[16] D. Hutter and M. Volkamer. Information flow control to secure dynamic web service composition. In *In International Conference on Security in Pervasive Computing*, pages 196–210. Springer, LNCS, 2006. 5

[17] G. Joseph Amadee and M. José. Security policy and security models. *In Proceedings of 1982 Symposium on Security and Privecy. IEEE Computer Society Press*, pages 11–20, 1982. 1, 3, 4

[18] M. Krohn, A. Yip, M. Brodsky, N. Cliffer, M. F. Kaashoek, E. Kohler, and R. Morris. Information flow control for standard os abstractions. *SIGOPS Oper. Syst. Rev.*, 41(6):321–334, 2007. 6

[19] D. R. Kuhn. Role based access control on mls systems without kernel changes. In *In Proceedings of the ACM Workshop on Role Based Access Control*, pages 25–32, 1998. 1

[20] H. Mantel. Possibilistic definitions of security - an assembly kit. In *Proceedings of the 13th IEEE workshop on Computer Security Foundations*, CSFW '00, pages 185–. IEEE Computer Society, 2000. 1

[21] D. McCullough. Noninterference and the composability of security properties. In *Proceedings of the 1988 IEEE conference on Security and privacy*, SP'88, pages 177–186. IEEE Computer Society, 1988. 1

[22] J. McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proceedings of the 1994 IEEE Symposium on Security and Privacy*, SP '94, pages 79–. IEEE Computer Society, 1994. 1

[23] J. Rushby. Noninterference, transitivity, and channel-control security policies. Technical report, dec 1992. 3

[24] A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE Journal on selected areas in communications*, 21(1), 2003. 1

[25] A. Sabelfeld and D. Sands. A per model of secure information flow in sequential programs. *Higher Order Symbol. Comput.*, 14(1):59–91, Mar. 2001. 1

[26] R. Sandhu, R. S, and Q. Munawer. How to do discretionary access control using roles, 1998. 1

[27] F. Seehusen, K. Stølen, and S. IKT. *A Method for Model-driven Information Flow Security*. SINTEF rapport. SINTEF ICT, 2009. 6

[28] J. Shen, S. Qing, Q. Shen, and L. Li. Covert channel identification founded on information flow analysis. In *Proceedings of the 2005 international conference on Computational Intelligence and Security - Volume Part II*, CIS'05, pages 381–387. Springer-Verlag, 2005. 1

[29] G. Smith and D. Volpano. Secure information flow in a multi-threaded imperative language. In *Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '98, pages 355–364. ACM, 1998. 1

[30] A. Zakinthinos and E. S. Lee. A general theory of security properties. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, SP '97, pages 94–. IEEE Computer Society, 1997. 1

[31] S. Zdancewic. Challenges for information-flow security. In *In Proc. Programming Language Interference and Dependence (PLID*, 2004. 6

[32] S. Zdancewic, L. Zheng, N. Nystrom, and A. C. Myers. Secure program partitioning. *ACM Trans. Comput. Syst.*, 20(3):283–328, Aug. 2002. 6

[33] N. Zeldovich, S. Boyd-Wickizer, and D. Mazières. Securing distributed systems with information flow control. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 293–308. USENIX Association, 2008. 6