Electronic Notes in Theoretical Computer Science 54 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume54.html 11 pages

Information Flow Security in Mobile Ambients [1]

Agostino Cortesi Riccardo Focardi

Dipartimento di Informatica,
Università Ca' Foscari di Venezia,
Via Torino 155, 30173 Venezia – Mestre (Italy)
{cortesi,focardi}@dsi.unive.it

Abstract

A multilevel security policy is considered in the scenario of mobile systems, and modeled within "pure" Mobile Ambients calculus, in which no communication channels are present and the only possible actions are represented by the moves performed by mobile processes. The information flow property of interest is defined in terms of the possibility for a confidential ambient/data to move outside a security boundary. Then, a very simple syntactic property is given that is sufficient to imply the absence of unwanted information flows.
Keywords: Mobile Ambients, Security, Static Analysis.

1 Introduction

The problem. When an user is identified and allowed to access some computer resources, an access control policy is imposed that guarantees that no information leak is possible. In particular, the system should detect "Trojan horses", i.e. (aware or unaware) malicious programs that hide their dangerous contents behind a trustworthy façade.

In this paper, we focus on *Multilevel Security*, a particular *Mandatory Access Control* security policy: every entity is bound to a security level (for simplicity, we consider only two levels: high and low), and information may just flow from the low level to the high one. Typically, two access rules are imposed: (*i*) *No Read Up*, a low level entity cannot access information of a high level entity; (*ii*) *No Write Down*, a high level entity cannot leak information to a low level entity. Sometimes, these two access controls are not enough as information may be indirectly leaked, through, e.g., some system side-effect: a typical example is represented by a resource shared among the security levels

[1] Partially supported by MURST Projects "Interpretazione Astratta, Type Systems e Analisi Control-Flow", and "Certificazione automatica di programmi mediante interpretazione astratta".

©2001 Published by Elsevier Science B. V.

which may be alternatively overloaded by some Trojan horse (causing, e.g., longer response time at all security levels) in order to transmit information to a malicious low level entity. These indirect ways of transmitting information are called *covert channels*. Figure1 summarizes this policy.

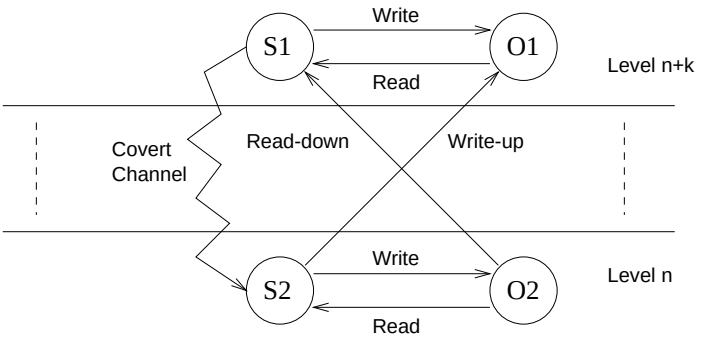

Fig. 1. Multilevel Security Policy.

In order to detect both direct and indirect information leakages, a typical approach (see, e.g., [2,5–7,9,10]) consists in directly defining what is an information flow from one level to another one. Then, it sufficient to verify that, in any system execution, no information flow is possible from level high to level low. This is the approach we follow in this paper.

The scenario. We will consider information flow security in the scenario of mobile systems. This particular setting, where code may migrate from one security level to another one, complicates even further the problem of capturing all the possible information leakages. As an example, confidential data may be read by an authorized agent which, moving around, could expose them to unexpected attacks. Moreover, the code itself could be confidential, and so not allowed to be read/executed by lower levels.

In order to study this problem as much abstractly as possible, we consider the "pure" Mobile Ambients calculus [4], in which no communication channels are present and the only possible actions are represented by the moves performed by mobile processes. This allows to study a very general notion of information flow which should be applicable also to more "concrete" versions of the calculus.

Verification. The information flow property of interest is defined in terms of the possibility for a confidential ambient/data to move outside a security boundary. We then give a very simple syntactic property and we prove, by exploiting the control flow analysis proposed in [8], that it is sufficient to imply the absence of unwanted information flows.

The rest of the paper is organized as follows. In Section 2 we introduce the basic terminology on ambient calculus and we briefly report the control flow analysis of [8]. Then, in Section 3, we present the model of multilevel

security for mobile agents and we show how to guarantee absence of unwanted information flows. Section 4 concludes the paper.

2 Background

In this section we introduce the basic terminology on ambient calculus and we briefly report the control flow analysis of [8].

2.1 Mobile Ambients

The Mobile Ambients calculus has been introduced in [4] with the main purpose of explicitly modelling mobility. Indeed, ambients are arbitrarily nested boundaries which can moves around through suitable capabilities. The syntax of processes is given as follows, where n denotes an ambient name.

P, Q	::=	$(\nu n)P$	restriction
	|	$\mathbf{0}$	inactivity
	|	$P \mid Q$	composition
	|	$!P$	replication
	|	$n^{\ell^a}[\,P\,]$	ambient
	|	$in^{\ell^t} n.P$	capability to enter n
	|	$out^{\ell^t} n.P$	capability to exit n
	|	$open^{\ell^t} n.P$	capability to open n

The labels $\ell^a \in \mathbf{Lab}^a$ on ambients and labels $\ell^t \in \mathbf{Lab}^t$ on transitions, have been introduced in the control flow analysis proposed in [8]. This is just a way of indicating "program points" and will be useful in the next section when developing the analysis.

Intuitively, the restriction $(\nu n)P$ introduces the new name n and limits its scope to P; $\mathbf{0}$ does nothing; $P \mid Q$ is P and Q running in parallel; replication provides recursion and iteration as $!P$ represents any number of copies of P in parallel. By $n^{\ell^a}[\,P\,]$ we denote the ambient named n with the process P running inside it. The capabilities $in^{\ell^t} n$ and $out^{\ell^t} n$ move their enclosing ambients in and out ambient n, respectively; the capability $open^{\ell^t} n$ is used to dissolve the boundary of a sibling ambient. For more details see [4,8].

2.2 Control Flow Analysis

The control flow analysis described in [8] aims at modelling which processes can be inside what other processes. It works on pairs $(\hat{I}, \hat{H})$, where:

- The first component $\hat{I}$ is an element of $\wp(\mathbf{Lab}^a \times (\mathbf{Lab}^a \cup \mathbf{Lab}^t))$. If a process contains an ambient labelled ℓ^a having inside either a capability or an ambient labelled ℓ, then (ℓ^a, ℓ) is expected to belong to $\hat{I}$.

$$
\begin{array}{lll}
(res) & \beta^{\mathrm{CF}}_{\ell}((\nu n)P) & = \beta^{\mathrm{CF}}_{\ell}(P) \\
(zero) & \beta^{\mathrm{CF}}_{\ell}(\mathbf{0}) & = (\emptyset, \emptyset) \\
(par) & \beta^{\mathrm{CF}}_{\ell}(P \mid Q) & = \beta^{\mathrm{CF}}_{\ell}(P) \sqcup \beta^{\mathrm{CF}}_{\ell}(Q) \\
(repl) & \beta^{\mathrm{CF}}_{\ell}(!P) & = \beta^{\mathrm{CF}}_{\ell}(P) \\
(amb) & \beta^{\mathrm{CF}}_{\ell}(n^{\ell^a}[\,P\,]) & = \beta^{\mathrm{CF}}_{\ell^a}(P) \sqcup (\{(\ell, \ell^a)\}, \{(\ell^a, n)\}) \\
(in) & \beta^{\mathrm{CF}}_{\ell}(in^{\ell^t} n.P) & = \beta^{\mathrm{CF}}_{\ell}(P) \sqcup (\{(\ell, \ell^t)\}, \emptyset) \\
(out) & \beta^{\mathrm{CF}}_{\ell}(out^{\ell^t} n.P) & = \beta^{\mathrm{CF}}_{\ell}(P) \sqcup (\{(\ell, \ell^t)\}, \emptyset) \\
(open) & \beta^{\mathrm{CF}}_{\ell}(open^{\ell^t} n.P) & = \beta^{\mathrm{CF}}_{\ell}(P) \sqcup (\{(\ell, \ell^t)\}, \emptyset)
\end{array}
$$

Fig. 2. Representation Function for Control Flow Analysis

$$
\begin{array}{lll}
(res) & (\hat{I}, \hat{H}) \models^{\mathrm{CF}} (\nu n)P & \text{iff } (\hat{I}, \hat{H}) \models^{\mathrm{CF}} P \\
(zero) & (\hat{I}, \hat{H}) \models^{\mathrm{CF}} \mathbf{0} & \text{always} \\
(par) & (\hat{I}, \hat{H}) \models^{\mathrm{CF}} P \mid Q & \text{iff } (\hat{I}, \hat{H}) \models^{\mathrm{CF}} P \ \wedge\ (\hat{I}, \hat{H}) \models^{\mathrm{CF}} Q \\
(repl) & (\hat{I}, \hat{H}) \models^{\mathrm{CF}} !P & \text{iff } (\hat{I}, \hat{H}) \models^{\mathrm{CF}} P \\
(amb) & (\hat{I}, \hat{H}) \models^{\mathrm{CF}} n^{\ell^a}[\,P\,] & \text{iff } (\hat{I}, \hat{H}) \models^{\mathrm{CF}} P \\
(in) & (\hat{I}, \hat{H}) \models^{\mathrm{CF}} in^{\ell^t} n.P & \text{iff } (\hat{I}, \hat{H}) \models^{\mathrm{CF}} P \wedge \\
 & \quad \forall \ell^a, \ell^{a'}, \ell^{a''} \in \mathbf{Lab}^a : ((\ell^a, \ell^t) \in \hat{I} \ \wedge\ (\ell^{a''}, \ell^a) \in \hat{I} \ \wedge\ (\ell^{a''}, \ell^{a'}) \in \hat{I} & \\
 & \quad\quad \wedge\ (\ell^{a'}, n) \in \hat{H}) \implies (\ell^{a'}, \ell^a) \in \hat{I} & \\
(out) & (\hat{I}, \hat{H}) \models^{\mathrm{CF}} out^{\ell^t} n.P & \text{iff } (\hat{I}, \hat{H}) \models^{\mathrm{CF}} P \wedge \\
 & \quad \forall \ell^a, \ell^{a'}, \ell^{a''} \in \mathbf{Lab}^a : ((\ell^a, \ell^t) \in \hat{I} \ \wedge\ (\ell^{a'}, \ell^a) \in \hat{I} \ \wedge\ (\ell^{a''}, \ell^{a'}) \in \hat{I} & \\
 & \quad\quad \wedge\ (\ell^{a'}, n) \in \hat{H}) \implies (\ell^{a''}, \ell^a) \in \hat{I} & \\
(open) & (\hat{I}, \hat{H}) \models^{\mathrm{CF}} open^{\ell^t} n.P & \text{iff } (\hat{I}, \hat{H}) \models^{\mathrm{CF}} P \wedge \\
 & \quad \forall \ell^a, \ell^{a'} \in \mathbf{Lab}^a : ((\ell^a, \ell^t) \in \hat{I} \ \wedge\ (\ell^a, \ell^{a'}) \in \hat{I} \ \wedge\ (\ell^{a'}, n) \in \hat{H}) & \\
 & \quad\quad \implies \left\{(\ell^a, \ell') \mid (\ell^{a'}, \ell') \in \hat{I}\right\} \subseteq \hat{I} &
\end{array}
$$

Fig. 3. Specification of Control Flow Analysis

- The second component $\hat{H}$ keeps track of the correspondence between names and labels. If a process contains an ambient labelled ℓ^a with name n, then (ℓ^a, n) is expected to belong to $\hat{H}$.

The analysis is defined by a representation function and a specification. [2] They are recalled, respectively, in Figure 2 and Figure 3.

The representation function mainly collects information about all ambient nestings yieldt by a process, in its initial state. The representation of a process P is defined as $\beta^{\mathrm{CF}}_{\ell^a_*}(P)$, where label ℓ^a_* is a special label corresponding to the environment.

The specification mostly amounts to recursive checks of subprocesses. The *open*-capability says that if some ambient labelled ℓ^a has an *open*-capability ℓ^t on an ambient n that may apply due to the presence of a sibling ambient labelled $\ell^{a'}$ whose name is just n, then the result of performing that capability should also be recorded in $\hat{I}$. The *in* and *out* capabilities behave similarly.

The correctness of the analysis is proven by showing that every reduction of the semantics is properly mimicked in the analysis:

Theorem 2.1 *Let P and Q be two processes such that* $\beta^{\mathrm{CF}}_{\ell^a_*}(P) \sqsubseteq (\hat{I}, \hat{H}) \ \wedge \ (\hat{I}, \hat{H}) \models^{\mathrm{CF}} P \ \wedge \ P \to Q$ *then* $\beta^{\mathrm{CF}}_{\ell^a_*}(Q) \sqsubseteq (\hat{I}, \hat{H}) \ \wedge \ (\hat{I}, \hat{H}) \models^{\mathrm{CF}} Q$

Intuitively, whenever $(\hat{I}, \hat{H}) \models^{\mathrm{CF}} P$ and the representation of P is contained in $(\hat{I}, \hat{H})$, we are assured that every nesting of ambients and capabilities in every possible derivative of P is also captured in $(\hat{I}, \hat{H})$.

It is important to recall also that the resulting control flow analysis applies to any process, and that every process enjoys a *least* analysis.

3 Information Flow

In this section, we present a formalization of multilevel security in the setting of Mobile Ambients. Then, a simple syntactical property is given which allows to verify the absence of unwanted information flows.

3.1 Modelling Multilevel Security

In order to define Multilevel security in Mobile Ambients we first need to classify information into different levels of confidentiality. We do that by exploiting the labelling of ambients. In particular, we partition the set of ambient labels $\mathbf{Lab}^a$ into three disjoint sets $\mathbf{Lab}^a_H, \mathbf{Lab}^a_L$ and $\mathbf{Lab}^a_B$, which stand for *high*, *low* and *boundary* labels.

Given a process, the multilevel security policy may be established by deciding which ambients are the ones responsible of confining confidential information. These are all labelled with boundary labels from set $\mathbf{Lab}^a_B$ and we will refer to them as *boundary ambients*. Then, all the ambients initially contained in a boundary ambient, are considered high level ambients and are labelled

[2] In ambient calculus bound names may be α-converted. For the sake of simplicity, here we are assuming that ambient names are *stable*, i.e., n is indeed a representative for a class of α-convertible names. See [8] for more details on how this can be handled.

with labels from set $\mathbf{Lab}^a_H$. Finally, all the external ambients are considered low level ones and consequently labelled with labels from set $\mathbf{Lab}^a_L$. This is how we will always label processes, and corresponds to defining the security policy (what is secret, what is not, what is a container of secrets).

As an example consider the following process:

$$P = container^b[\, hdata^h[\, out^c container.\mathbf{0}\,]\,] \mid Q$$

where $b \in \mathbf{Lab}^a_B, h \in \mathbf{Lab}^a_H$, $c \in \mathbf{Lab}^t$ is a capability label and Q contains some low level ambients. Ambient *container* is a boundary for the high level data *hdata* (note that data are abstractly represented as ambients). This process is an example of a direct information flow. Indeed, P may evolve to $container^b[\,] \mid hdata^h[\,] \mid Q$, where the high level *hdata* is out of any boundary ambient, thus vulnerable and potentially accessible by any ambient or process in Q. [3] This flow of high level data/ambients outside the security boundaries is exactly what we intend to control and avoid.

In distributed and mobile systems, it is unrealistic to consider a unique boundary, containing all the confidential information. As an example consider two different sites *venice* and *lipari*, each with some set of confidential information that need to be protected. This can be modelled by just defining two boundary ambients, one for each site: $venice^b[\, P_1\,] \mid lipari^b[\, P_2\,] \mid Q$. In order to make the model applicable, it is certainly needed a mechanism for moving confidential data from one boundary to another one. This is achieved through another boundary ambient which moves out from the first protected area and into the second one. An example follows:

$$venice^b[\, send^b[\, out^c venice.in^c lipari \mid hdata^h[\,\,]\,]\,] \mid lipari^b[\, open^c send\,] \mid Q$$

that may evolve to:

$$venice^b[\,] \mid lipari^b[\, open^c send \mid send^b[\, hdata^h[\,\,]\,]\,] \mid Q$$

and finally to:

$$venice^b[\,] \mid lipari^b[\, hdata^h[\,\,]\,] \mid Q$$

Note that *send* is labelled as a boundary ambient. Thus, the high level data *hdata* is always protected by boundary ambients, during all the execution.

3.2 Verifying Absence of Information Flows

In this section, we study how to verify that no leakage of secret data/ambients outside the boundary ambients is possible. A natural approach could be the

[3] Note that the presence of an ambient may be tested by trying to open it or by entering and then exiting from it. A low level ambient may thus test if *hdata* is present. This may be seen as reading such high level information.

direct application of the control flow of [8] reported in section 2.2. As a matter of fact, consider again the example presented above:

$$venice^b[\, send^b[\, out^c venice.in^c lipari \mid hdata^h[\ ]\,]\,] \mid lipari^b[\, open^c send\,]$$

The least analysis for this process can be easily shown to be the following:

$$\hat{I} = \{(l^a_*, b), (b, b), (b, h), (b, c)\}$$
$$\hat{h} = \{(b, venice), (b, send), (b, lipari), (h, hdata)\}$$

The important thing is that h is always contained inside b, i.e., a boundary ambient. This basically proves that the system is secure and no leakage of h data may happen.

However, the fact that the analysis simply collects all the potential nesting without considering the temporal ordering of the events, may sometimes be too approximated. As an example, consider again the previous process and suppose that high level data is willing to enter some filter process, which could possibly be low level code:

$$venice^b[\, send^b[\, out^c venice.in^c lipari \mid hdata^h[\, in^{ch} filter\,]\,]\,] \mid$$
$$\mid lipari^b[\, open^c send\,] \mid filter^m[\, in^c send\,] \mid open^{cl} filter$$

Note that the filter behaves correctly with respect to multilevel security rules, i.e., it only enters boundaries. In particular, this means that it will never transport high level data outside the security boundaries. However, if we perform the control flow analysis we obtain the following least solution:

$$\hat{I} = \{(l^a_*, b), (l^a_*, h), (l^a_*, m), (l^a_*, cl), (b, b), (b, h), (b, m), (b, c), (h, ch),$$
$$(m, h), (m, c)\}$$
$$\hat{h} = \{(b, venice), (b, send), (b, lipari), (h, hdata)\}$$

Note that h appears at the environment level, showing a potential attack. However, as observed before, there is no execution leading to such a situation. The reason why the analysis looses precision here, is due to the fact that h enters a m ambient which might be opened at the environment level, but the analysis does not capture the fact that h enters m only after it has crossed the boundary and can never return back.

In the following, we study a different (syntactic) condition on processes that is sufficient to prove the absence of leakage of secret data/ambients outside the boundary ambients. Moreover, such a condition properly deals with the situation discussed before.

The idea is to control the $out^l n$ and $open^l n$ capabilities executed on a boundary ambient n. In particular, we require that such capabilities may only be performed by boundary ambients.

First, we characterize a subset of capability labels, in order to mark *out* and *open* capabilities that refer to boundary ambients. Let $\mathbf{Lab}^t_O \subseteq \mathbf{Lab}^t$ be the

subset of labels that refer to *out* and *open* capabilities, and let $\mathbf{Lab}^t_{BM} \subseteq \mathbf{Lab}^t_O$ be a subset of this set of *out* and *open* capability labels. BM stands for *boundary moves* capabilities. Let also $\phi : \mathbf{Lab}^t \to \wp(\mathbf{Amb})$ be a function that given a capability label ℓ^t, returns the set of ambient names on which all the capabilities labelled with ℓ^t operate.

Given a process P, the conditions that should be imposed on $\beta^{\mathrm{CF}}_{\ell^a_*}(P)$ to guarantee absence of information leakage are the following.

$$(i) \quad (\ell^a, n) \in \hat{H}, \ell^a \in \mathbf{Lab}^a_B, n \in \phi(\ell^t), \ell^t \in \mathbf{Lab}^t_O \Rightarrow \ell^t \in \mathbf{Lab}^t_{BM}$$

$$(ii) \quad (\ell, \ell') \in \hat{I}, \ell' \in \mathbf{Lab}^t_{BM} \Rightarrow \ell \in \mathbf{Lab}^a_B$$

Observe that condition (i) results in a well-formedness labelling. It requires that all the *out* and *open* capabilities that operate on boundary ambients are labelled as boundary moves (i.e., with labels in set $\mathbf{Lab}^t_{BM}$). If this condition is initially satisfied by P (i.e., by $\beta^{\mathrm{CF}}_{\ell^a_*}(P)$), then it will hold also for every derivative of P, as the labelling cannot change during process execution.

Condition (ii) requires that every *out* and *open* boundary move is executed inside a boundary ambient. Note that, in general, this may be not preserved when P evolves. Indeed, the following theorem states that also condition (ii) above is preserved, in every execution of P.

Theorem 3.1 *If the representation function $\beta^{\mathrm{CF}}_{\ell^a_*}(P)$ initially fulfills conditions $(i)-(ii)$, then the least solution $(\hat{I}, \hat{H}) \models^{\mathrm{CF}} P$ to the control flow analysis enjoys these conditions as well.*

Sketch of the proof. We have stated that condition (i) is trivially preserved. Let us consider condition (ii). The fact that we consider the least solution means that all the elements in $\hat{I}$ and $\hat{H}$ are either in $\beta^{\mathrm{CF}}_{\ell^a_*}(P)$ or introduced by the three rules for *in*, *out* and *open* of Figure 3. Capabilities *in* and *out* move ambients as a whole, therefore the corresponding rules do not affect condition (ii). Then, the only interesting rule is the last one, when *open* arises. Let a new pair (ℓ^a, ℓ') be introduced in $\hat{I}$ by that rule, with $\ell' \in \mathbf{Lab}^t_{BM}$ and $(\ell^{a'}, \ell') \in \hat{I}$. By induction, we may assume $\ell^{a'} \in \mathbf{Lab}^a_B$. As $(\ell^{a'}, n) \in \hat{H}$, ambient n is labelled within $\mathbf{Lab}^a_B$. By condition (i) on well-formedness of labelling, $\ell^t \in \mathbf{Lab}^t_{BM}$, yielding $\ell^a \in \mathbf{Lab}^a_B$, that concludes the proof.

Condition (ii) basically states two important properties on P execution: every time a boundary ambient is opened, this is done inside another boundary ambient; the only ambients that may exit from a boundary ambients are boundary ambients. By induction on reduction rules of Mobile Ambients it is now easy to prove the following information flow theorem:

Theorem 3.2 *If $\beta^{\mathrm{CF}}_{\ell^a_*}(P)$ fulfills conditions $(i) - (ii)$, then, in every Q s.t. $P \to Q$, every high level ambient is always inside at least one boundary ambient.*

Note that the conditions are really simple to check. As an example consider again the two example presented above. In particular,

$$P = container^b[\, hdata^h[\, out^c container.\mathbf{0}\,]\,] \mid Q$$

does not satisfy condition (ii) as $out^c container$, by condition (i), should be labelled as a boundary move. However this makes a boundary move executable in a high level ambient, invalidating condition (ii). On the other side, the second example

$$venice^b[\, send^b[\, out^c venice.in^c lipari \mid hdata^h[\ ]\,]\,] \mid lipari^b[\, open^c send\,] \mid Q$$

fulfills both the conditions, with $c \in \mathbf{Lab}^t_{BM}$. This proves that $hdata$, in every execution, is always inside a boundary ambient.

The syntactic conditions successfully applies also to the extended example with $hdata$ entering the filter:

$$venice^b[\, send^b[\, out^c venice.in^c lipari \mid hdata^h[\, in^{ch} filter\,]\,]\,] \mid$$
$$\mid lipari^b[\, open^c send\,] \mid filter^m[\, in^c send\,] \mid open^{cl} filter$$

Also in this case, we are able to prove that $hdata$, in every execution, is always inside a boundary ambient. Note that this was not provable through the presented control flow analysis.

Finally, the approach may also be adapted to the case in which the external environment (e.g. any malicious process put in parallel with the analyzed process P) does not fulfill the required conditions. This is indeed reasonable in a distributed open system. The idea is to suitably restrict the scope of boundary ambients and provide low level ambients with some "taxi" processes that, once entered, bring the client inside restricted boundaries. Let $b_1, \ldots, b_n$ represent all the boundary ambients of process P. Then consider process

$$(\nu b_1, \ldots, b_n)(P \mid !t_1[in^l b_1] \mid \ \ldots \ \mid !t_n[in^l b_n]) \mid Q$$

As $b_1, \ldots, b_n$ are restricted names, they may not appear in Q. As a consequence, if P fulfills the conditions $(i) - (ii)$, this is sufficient to prove that the whole system (whatever Q is considered) satisfies such conditions, too. It is indeed simple to prove the following:

Proposition 3.3 *If $\beta^{\mathrm{CF}}_{\ell^a_*}(P)$ fulfills conditions $(i) - (ii)$, then, for all Q (labelled in $\mathbf{Lab}^a_L \cup \mathbf{Lab}^t \setminus \mathbf{Lab}^t_{BM}$),*

$$\beta^{\mathrm{CF}}_{\ell^a_*}((\nu b_1, \ldots, b_n)(P \mid !t_1[in^l b_1] \mid \ \ldots \ \mid !t_n[in^l b_n]) \mid Q)$$

fulfills conditions $(i) - (ii)$.

Note that processes $!t_i[in^l b_i]$ allow any low level ambient to enter boundary b_i. So, legitimate flows from level to high level are possible even if boundaries

are restricted. Note also that the condition on the labelling of Q simply means that Q just contains low level ambients and its capabilities are not (incorrectly) labelled as boundary moves.

4 Conclusions

The main novelty of the approach presented in this paper is that we model multilevel information flow within a "pure" mobile ambient setting, without considering either channels or recently proposed restrictions of Mobile Ambients designed for security issues (like Secure Safe Ambients [3]). We have also proposed a syntactic condition which is sufficient to guarantee absence of unwanted information flow. Such a condition is really simple to verify and, at the same time, seems to be not restrictive.

As a future work, we intend to extend the approach to other versions of Mobile Ambients, and, in particular, to the full calculus with communication channels. It is our opinion that if only communication within ambients is considered, the approach should carry on very naturally. We also intend to compare our approach with other control flow analyses proposed for particular versions of Mobile Ambients, like, e.g., the one for Safe Ambients [1]. It would be also interesting to study if the approach proposed here could be applied, via some suitable encoding, also to "classical" calculi, like pi-calculus.

Finally, the way we propose to express a multilevel information flow policy through a syntactic characterization of ambient and capability labels, in our opinion, may also lead to interesting applications when trying to model cryptographic protocols by the same formalism.

Acknowledgements

We would like to thank the anonymous referees for their helpful comments and suggestions.

References

[1] P. Degano, F. Levi, C. Bodei. Safe Ambients: Control Flow Analysis and Security. In proceedings of *ASIAN'00*, LNCS 1961, 2000, pages 199-214.

[2] C. Bodei, P. Degano, F. Nielson, and H.R.Nielson. Static Analysis of Processes for No Read-Up and No-Write-Down. In In Proc. FoSSaCS'99, volume 1578 of *Lecture Notes in Computer Science*, pages 120–134, Springer-Verlag, 1999.

[3] M. Bugliesi and G. Castagna. "Secure Safe Ambients". Proc. 28th ACM Symposium on Principles of Programming Languages (POPL'01), pp. 222-235, London. 2001.

[4] L. Cardelli and A. Gordon. "Mobile Ambients". In Proc. FoSSaCS'98, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155, Springer-Verlag, 1998.

[5] R. Focardi and R. Gorrieri. "A Classification of Security Properties for Process Algebras", Journal of Computer Security, 3(1): 5-33, 1995.

[6] R. Focardi and R. Gorrieri, "The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties, IEEE Transactions on Software Engineering, Vol. 23, No. 9, September 1997.

[7] R. Focardi, R. Gorrieri, F. Martinelli, "Information Flow Analysis in a Discrete Time Process Algebra", in Proc. of 13th IEEE Computer Security Foundations Workshop (CSFW13), (P.Syverson ed), IEEE CS Press, 170-184, 2000.

[8] R. R. Hansen, J. G. Jensen, F. Nielson, and H. R. Nielson, Abstract Interpretation of Mobile Ambients. In Proc. Static Analysis Symposium SAS'99, volume 1694 of *Lecture Notes in Computer Science*, pages 134–148, Springer-Verlag, 1999.

[9] M. Hennessy, J. Riely. "Information Flow vs. Resource Access in the Asynchronous Pi-Calculus". ICALP 2000: 415-427.

[10] G. Smith, D.M. Volpano, "Secure Information Flow in a Multi-Threaded Imperative Language". In Proc. of POPL 1998: 355-364.