Poster: Modelling Distributed Systems Security by Labelled Mobile Ambients

N.V.Narendra Kumar
Doctoral Candidate
Tata Institute of Fundamental Research
Mumbai 4000005, India
naren@tifr.res.in

R.K. Shyamasundar
School of Tech. and Computer Science
Tata Institute of Fundamental Research
Mumbai 4000005, India
shyam@tifr.res.in

I. Introduction

DIFC proposed by Andrew Myers and Barbara Liskov [2], allows users to share information with a distrusted component yet control how that component propagates the shared information to others and provides an end-to-end security guarantees and several real-life implementations exist. The model extends the lattice-based information flow models by allowing users to declassify information in a decentralized way, and improves support for fine-grained data sharing. DIFC became very popular as it provides end-to-end security guarantees and there exist several implementations and applications both at the programming language level and at the level of operating systems. Language-based DIFC systems [3], [4] augment the type system to include secrecy and integrity constraints enforced by the bytecode generator. Mobile ambients [1], has been proposed to capture all aspects of mobility) within a single framework that encompasses mobile agents, interaction of ambients and their mobility.

In this paper, we propose a variant of Mobile ambients called Labelled Mobile Ambients (LMA) model and show that it forms a succinct basis for DIFC in a language independent manner and subsumes concepts of declassification and endorsement. We define the semantics of LMA and show that it realizes DIFC. This is realized by an appropriate interpretation of in, out and $open$ actions in terms of the standard $read$ and $write$ operations, with additional constraints based on the labels of the ambients involved in the action to succeed. Further, we show that LMA (i) enhances the security of distributed computations by integrating well known security models, enabling a wide range of applications like safe execution of untrusted components downloaded from the internet etc., (ii) succinctly demonstrates multilevel security properties, (iii) realizes declassification and endorsement, and (iv) provides obligations to be discharged by a programming language to realize multi-level security (MLS). The model enables derivation of necessary and sufficient conditions needed for realizing MLS properties in a LMA. Further, static analysis techniques can be used to certify proper information flow in such a model and avoid run-time information flow checks as much as possible; such an analysis can be used to validate whether a given implementation of an application satisfies the desired security properties and arrive at frameworks for realizing a given security policy.

In the sequel, we shall highlight some of the crucial features of LMA only due to limited space.

n			Names
P, Q	::=		Processes
		$(\nu n)P$	restriction
	|	$\mathbf{0}$	inactivity
	|	$P \mid Q$	composition
	|	$!P$	replication
	|	$n[P]$	ambient
	|	$M.P$	action
M	::=		Capabilities
		$\mathbf{in}\ n.P$	can enter n
	|	$\mathbf{out}\ n.P$	can exit n
	|	$\mathbf{open}\ n.P$	can open n

Figure 1. Mobile Ambients Syntax

II. A Labelled Mobile Ambient Model

In this section, we propose an advanced mobile ambient model referred to as Labelled Mobile Ambients (LMA). LMA model is the same as mobile ambients (syntax is shown in Fig. 1) except that

- ambients are labelled (labels assigned to an ambient are static, i.e. they do not change at run time) and that we have a function $label$ that takes the name of an ambient as input and returns its label as output. Labels of the ambients come from the lattice $\mathcal{L} = (L, \leqslant, \oplus)$
- the semantics of the operations in, out and $open$ have to be redefined taking the labels associated with the ambients (that get affected by the execution of the operation) into account

Given a process of the *form* $n[opr\ m.P \mid Q]$, we say that *subject* n intends to perform operation *opr* on *object* m. We now present the semantics of in, out and $open$ primitives in LMA and discuss the interpretation of modelling the *environment*. The latter is important from the perspective of secrecy/integrity as one needs to define "**trust**".

Semantics of "in"

$n_1[n_2[P] \mid n_3[in\ n_2.Q] \mid R] \mid S \rightarrow n_1[n_2[P \mid n_3[Q]] \mid R] \mid S$

The *subject* of the operation $in\ n_2$ in the above process is n_3 and the *object* is n_2(although, indirectly n_1 is also affected by it). If we look at the changes after the reduction, we deduce that the *contents* of n_1 and that of n_2 have been *modified*. So, we consider this as a **write** operation by n_3. We interpret the above reduction as follows: (i) n_3 writes n_2 and (ii) n_3 writes n_1. This would mean that information flows (i) from n_3 to n_2 and (ii) from n_3 to n_1. Under the constraints of the lattice model of information flow, for these information flows to succeed we need that $label(n_3) \leqslant label(n_2)$ and $label(n_3) \leqslant label(n_1)$.

Semantics of "out"

$n_1[n_2[n_3[out\ n_2.P] \mid Q] \mid R] \mid S \rightarrow n_1[n_2[P] \mid n_3[Q] \mid R] \mid S$

The *subject* of the operation $out\ n_2$ in the above process is n_3 and the *object* is n_2(although, indirectly n_1 is also affected by it). If we look at the changes after the reduction, we deduce that the *contents* of n_1 and that of n_2 have been *modified*. So, we consider this as a **write** operation by n_3. We interpret the above reduction as follows: (i) n_3 writes n_2 and (ii) n_3 writes n_1. This would mean that information flows (i) from n_3 to n_2 and (ii) from n_3 to n_1. Under the constraints of the lattice model of information flow, for these information flows to succeed we need that $label(n_3) \leqslant label(n_2)$ and $label(n_3) \leqslant label(n_1)$.

Semantics of "open"

$n_1[open\ n_2.P \mid n_2[Q] \mid R] \mid S \rightarrow n_1[P \mid Q \mid R] \mid S$

The *subject* of the operation $open\ n_2$ in the above process is n_1 and the *object* is n_2. If we look at the changes after the reduction we deduce that n_1 now has the access to the contents of n_2. So, we consider this as a **read** operation by n_1. We interpret the above reduction as n_1 reads n_2. This would mean that information flows from n_2 to n_1. Under the constraints of the lattice model of information flow, for this information flow to succeed we need that $label(n_2) \leqslant label(n_1)$.

Modelling the Environment

We treat the *environment* as the "**root**" (within which all the activities happen) and we assign it the label $*$ (the highest label) which stands for *all*. The properties the environment would then follow are: (i) the environment cannot write anything (it has no siblings or ancestors), (ii) no one can read the environment (it has no parent), (iii) we trust the environment not to read anything (i.e. not have any open capabilities) and (iv) anyone can write to the environment (in fact every ambient which wants to cross administrative boundaries would have to necessarily pass through the environment).

The LMA model realizes DIFC follows from:

- each ambient specifies an independent flow policy and retains control over the dissemination of its data
- code in an ambient with the authority of its owner ambient can modify its flow policy. It can declassify its data part by adding additional readers (through *open* operation). This is done on a per-ambient basis and there is no central declassification ambient
- each ambient ensures its data flow policy through in, out and $open$ capabilities
- the environment satisfies the initial flow policy and does not have any active role in controlling the data flow

III. Security properties of LMA

One of the advantages of the LMA is that using static analysis techniques, we can derive a set of constraints on the labels of the ambients that when satisfied guarantee that the process does not leak information The advantages of such a static analysis are (i) it eliminates the need for runtime checks and (ii) enables certification of components (for example generated by a trusted compiler).

Observation:

The label of an ambient must be higher than the label of any of its child ambients

Rules for constructing the set of constraints: Let $constraints_l(P)$, where process P is in an ambient labelled l be the set of constraints to be satisfied by the labels of ambients of the process so that no possible execution leaks information. It can be derived using the following rules using structural induction:

1) $constraints_l((\nu n)P) = constraints_l(P)$
2) $constraints_l(0) = \emptyset$
3) $constraints_l(P \mid Q) = constraints_l(P) \cup constraints_l(Q)$
4) $constraints_l(!P) = constraints_l(P)$
5) $constraints_l(n[P]) = constraints_{label(n)}(P) \cup \{label(n) \leqslant l\}$
6) $constraints_l(in\ n.P) = constraints_l(P) \cup \{l \leqslant label(n)\}$
7) $constraints_l(out\ n.P) = constraints_l(P)$
8) $constraints_l(open\ n.P) = constraints_l(P)$

The information leak is captured below.

Definition 1: We say that a process P leaks information iff $\exists Q$ $P \rightarrow^* Q$ and ambient n_1 is a child of the ambient n_2 in Q and $label(n_2) < label(n_1)$.

Proposition 1: If P and Q are such that $P \rightarrow Q$ then the following hold

1) $constraints(P) \supseteq constraints(Q)$
2) $constraints(P) \models P \Rightarrow constraints(P) \models Q$.

Corollary 1: If P is such that $constraints(P) \models P$ then P will not leak information.

For several practical applications, strict enforcement of non-interference is too inflexible. However, it is very sensitive. In LMA. The process of declassification and endorsement envisaged in several works by Myers and others, can be realized in LMA through trusted principals.

IV. Discussions

To our knowledge, this is the first attempt that shows an adaptation of ambient calculus to succinctly model DIFC and multi-level security properties. The LMA model provides a general framework for multi-level security in a programming language independent way. While using explicit labels for large systems may not be easy, we believe that using the notion of trusted principals, it should be possible to integrate with legacy systems. Further, generalizations of the model to dynamic labelling are being explored.

References

[1] L. Cardelli and A. D. Gordon, "Mobile ambients," *Theoretical Computer Science*, vol. 240, no. 1, pp. 177–213, 2000.

[2] A. C. Myers and B. Liskov, "A decentralized model for information flow control," in *SOSP '97*. New York, NY, USA: ACM, 1997, pp. 129–142.

[3] A. C. Myers, "Jflow: Practical mostly-static information flow control," in *POPL*, 1999, pp. 228–241.

[4] V. Simonet and I. Rocquencourt, "Flow caml in a nutshell," in *Proc.of the 1st APPSEM-II workshop*, 2003, pp. 152–165.