Security Analysis for Mobile Ambients

Francesca Levi, Chiara Bodei

Dipartimento di Informatica, Università di Pisa

Corso Italia, 40, I-56125 Pisa, Italy – {levifran,chiara}@di.unipi.it

Introduction. Mobile Ambients (*MA* [4]) has recently emerged as a core programming language for the Web and, at the same time, as a model for reasoning about properties of mobile processes. It is based on the notion of *movement* of ambients, i.e. named locations, that can be nested and can move according to three basic capabilities: $\mathtt{in}\, n$ and $\mathtt{out}\, n$ allow an ambient to enter into or to exit from an ambient n, while $\mathtt{open}\, n$ allows to destroy the boundary of an ambient n. In each case, the affected ambient n has no control on when the action takes place. *Safe Ambients* (*SA* [5]) extends *MA*, by allowing a movement to take place only when the involved ambient agrees, offering the corresponding *coaction*: $\overline{\mathtt{in}}\ n, \overline{\mathtt{out}}\ n, \overline{\mathtt{open}}\ n$. This variation does not change the expressiveness of *MA*, yet reduces the space of possible computations.

The literature presents several techniques to prove security properties of calculi for mobility, among which static techniques based on Type Systems [5, 2, 1, 3] and Control Flow Analysis (CFA) [6, 7, 8]. These techniques predict safe and computable approximations to the set of values or behaviours arising dynamically. Particularly relevant here is the CFA for *MA* developed in [6] and applied to prove some security properties on firewall protocols.

We here refine the analysis in [6], extended to *SA*, by using a sort of contextual information. As usual, our CFA is formulated as a specification of the correctness of a candidate solution, it has best solutions and is sound.

As a simple example, we apply it for proving a *secrecy* property. Ambients are classified into *trustworthy* and *untrustworthy*. Secrecy of data (being it either capabilities or ambients) is preserved if an untrustworthy ambient can never open a trustworthy one. We show that if a program passes a simple static test, then its secret information is dynamically protected.

In proving a security property it is essential to be able to guarantee that properties are preserved when programs work in unknown, possibly hostile contexts. In general, not all contexts preserve our property, but we can say which of them do. As expected, the needed restrictions concern the usage of names, actually some of them cannot be duplicated. We define a tester process E that represents the most hostile context that matches these requirements. If a process P in parallel with E passes the static test, then the secrecy property holds dynamically in any context represented by E.

Many research lines are still open, besides the use of shape analysis [8] for improving the precision of our analysis. Among them, there is the problem of constructing solutions efficiently and of considering also communication.

Mobile Safe Ambients. The syntax of *SA* ([5]) without communication primitives, see below, is based on a countable, infinite set of *names* $\mathcal{N}'$.

$P ::=$	processes	$M ::=$	capabilities
$\mathbf{0}$	*nil*	$\mathtt{in}\, n$	*enter* n
$M.P$	*prefix*	$\overline{\mathtt{in}}\ n$	*allow enter* n
$n[P]$	*ambient*	$\mathtt{out}\, n$	*exit* n
$P \mid P$	*parallel composition*	$\overline{\mathtt{out}}\ n$	*allow exit* n
$(\nu n)P$	*restriction*	$\mathtt{open}\, n$	*open* n
$!P$	*replication*	$\overline{\mathtt{open}}\ n$	*allow open* n

We refer to the usual notions of names, free and bound of a process P, denoted by $\mathrm{n}(P)$, $\mathrm{fn}(P)$, $\mathrm{bn}(P)$, respectively. Also, we assume an external ambient, $\top \notin \mathcal{N}'$. So we let $\mathcal{N} = \mathcal{N}' \cup \{\top\}$.

The reduction semantics of *SA* is defined as for *MA* except for the following basic movement rules, up to the usual structural congruence $\equiv$:

$$\mathtt{In}:\ n[\,\mathtt{in}\, m.\,P_1 \mid P_2\,] \mid m[\,\overline{\mathtt{in}}\ m.\,Q_1 \mid Q_2\,] \longrightarrow m[\,n[\,P_1 \mid P_2\,] \mid Q_1 \mid Q_2\,]$$

$$\mathtt{Out}:\ m[\,n[\,\mathtt{out}\, m.\,P_1 \mid P_2\,] \mid\ \overline{\mathtt{out}}\ m.\,Q_1 \mid Q_2\,] \longrightarrow n[\,P_1 \mid P_2\,] \mid m[\,Q_1 \mid Q_2\,]$$

$$\mathtt{Open}:\ m[\,n[\,\mathtt{open}\, m.\,P_1 \mid P_2\,] \mid \overline{\mathtt{open}}\ m.\,Q_1 \mid Q_2\,] \longrightarrow n[\,P_1 \mid P_2\,] \mid m[\,Q_1 \mid Q_2\,]$$

Capabilities and ambients are ranged over by α. Also, we say that

1. α is *top level* in a process P iff α does not occur inside any ambient;
2. α is *enabled* in a process P iff α is top level in P and it does not occur underneath a prefix.

Whenever α is enabled in a process P, then α may be exerced (if it is a capability), and may interact with other processes at the same level (if it is an ambient).

A *context* $C[-]$ is *flat* iff the hole is enabled in $C[-]$. Also, we call Q *configuration* of P if $P \equiv C[Q]$. When C is empty we write $\top[P]$.

We say that n is *opened* by k in P iff $k[\,\boldsymbol{\nu}\tilde{p}(n[\,R_1\,] \mid R_2)\,]$ is a configuration of P and $\overline{\texttt{open}}\,n$ and $\texttt{open}\,n$ are enabled in R_1 and in R_2, respectively.

Control Flow Analysis for *SA*. Our analysis predicts for each ambient n which capabilities and ambients may be contained inside n at run-time, when n occurs within a selected ambient. The solution to the analysis of a process P is a function ϕ. A typical element of $\phi(n)$ is $\alpha^{(\ell_1,\ell_2)}$, where ℓ_1, ℓ_2 are sets of names. Intuitively, when $\alpha^{(\ell_1,\ell_2)} \in \phi(n)$, the derivatives of P may include the following configurations:

1. for each $k \in \ell_1$, $k[\,\boldsymbol{\nu}\tilde{p}\,(R \mid n[\,Q\,])\,]$ where α is *top level* in Q;
2. for each $k \in \ell_2$, $k[\,\boldsymbol{\nu}\tilde{p}\,(R \mid n[\,P\,])\,]$ where α is *enabled* in P.

Although $\ell_2 \subseteq \ell_1$ (α enabled implies it is also top level), both sets are useful for an accurate prediction of process behaviour. For $\alpha^L \in \phi(n)$:

1. ℓ_2 approximates the set of ambients that surround n when α can be exerced (if capability) or is ready to interact (if ambient);
2. ℓ_1 approximates the set of ambients that can acquire α by opening n.

We do not show the formal clauses to validate the correctness of a solution. Instead, we illustrate the above through the following simple example.

Let $P = n[\,\texttt{in}\,k.\,\overline{\texttt{out}}\,n \mid m[\,\texttt{out}\,n.\,\overline{\texttt{open}}\,m.\,\texttt{in}\,a\,]\,] \mid k[\,\overline{\texttt{in}}\,k.\,\texttt{open}\,m\,]$ and let ϕ be one of its solutions.

$\phi(n)$	$\{m^{(\{\top,k\},\{\top,k\})}, \texttt{in}\,k^{(\top,\top)}, \overline{\texttt{out}}\,n^{(\{\top,k\},k)}\}$
$\phi(m)$	$\{\texttt{out}\,n^{(n,n)}, \overline{\texttt{open}}\,m^{(\{n,k\},k)}, \texttt{in}\,a^{(\{n,k\},k)}\}$
$\phi(k)$	$\{\overline{\texttt{in}}\,k^{(\top,\top)}, \texttt{open}\,m^{(\top,\top)}, \texttt{in}\,a^{(\{n,k,\top\},\{k,\top\})}, n^{(\top,\top)}, k^{(\top,\top)}\}$

• $\phi(n)$ contains: $m^{(\{\top,k\},\{\top,k\})}$ because m is enabled when n is both in the external ambient $\top$ and in k; $\texttt{in}\,k^{(\top,\top)}$ because $\texttt{in}\,k$ is enabled when n is inside $\top$ and may be acquired if n is opened by a process in $\top$; $\overline{\texttt{out}}\,n^{(\{\top,k\},k)}$ because $\overline{\texttt{out}}\,n$ is enabled only when $\texttt{in}\,k$ has been consumed (and so n is inside k), while it may be acquired by $\texttt{open}\,n$ both in $\top$ and k.

• $\phi(m)$ contains: $\mathtt{out}\, n^{(n,n)}$ because $\mathtt{out}\, n$ is enabled when m is inside n; $\overline{\mathtt{open}}\, n$ and $\mathtt{in}\, n$ with label $(\{n,k\},k)$ because $\overline{\mathtt{open}}\, n$ and $\mathtt{in}\, n$ are top level when m is inside n or k, while they are enabled only after the execution of $\mathtt{out}\, n$. Note that the k in $\overline{\mathtt{out}}\, n^{(\{\top,k\},k)} \in \phi(n)$ predicts the effect of $\mathtt{out}\, n$ on the movements of m. Intuitively, m is allowed to exit from n only where $\overline{\mathtt{out}}\, n$ is enabled and thus in k.
• $\phi(k)$ contains: $n^{(\top,\top)}$ and $m^{(\top,\top)}$, because both ambients may enter in k (n by exercing $\mathtt{in}\, k$ and m by exercing $\mathtt{out}\, n$); $\mathtt{in}\, k^{(\top,\top)}$ and $\mathtt{open}\, m^{(\top,\top)}$, because m is always contained in $\top$; $\mathtt{in}\, a^{(\{n,k,\top\},\{k,\top\})}$, because $\mathtt{in}\, a$ is acquired from ambient m through $\mathtt{open}\, m$. Note that capabilities $\mathtt{out}\, n$ and $\overline{\mathtt{open}}\, m$ contained in $\phi(m)$ are not acquired by k, because they do not contain k in the first component of the label.

A Security Property. An ambient m may acquire the information contained in another ambient n by opening it. We wish to maintain some information confidential to a group of ambients, considered trustworthy. To do so, names $\mathcal{N}$ (including $\top$) are partitioned into *trustworthy* $\mathcal{T}$ and *untrustworthy* $\mathcal{U}$. Secrecy of data is preserved if an untrustworthy ambient can never open a trustworthy one. Based on this partition, we first formalise our dynamic notion of secrecy: *protection*; then we give a static notion that implies the dynamic one. (Below $\Longrightarrow$ stands for the transitive closure of $\longrightarrow$.)

- **Dynamic Property** A process P is *protected* iff $\forall n,k \in \mathcal{N}$ with $n \in \mathcal{T}$, whenever $P \Longrightarrow Q$ and n is opened by k in Q, then $k \in \mathcal{T}$.
- **Static Property** A process P is *defended* if there exists a solution ϕ such that $\phi \models P$ and, $\forall n,k \in \mathcal{N}$ with $n \in \mathcal{T}$, whenever $\mathtt{open}\, n^{L_1} \in \phi(k)$, $\overline{\mathtt{open}}\, n^{L_2} \in \phi(n)$ and $k \in \ell_{2,2}$, then $k \in \mathcal{T}$.

Lemma 1 *If P is* defended *then it is* protected*.*

The property above is not enough, because it does not guarantee that a *defended* process P will still be such when plugged in an hostile context $C[-]$, unless the whole $C[P]$ is analysed again. However, we can characterize those contexts that do not break the secrecy of P, and also characterize the tester process, called enemy, which represents the most hostile context in which P can be put.

Definition 2 (Enemy) *The* enemy *of a process P w.r.t. $N \subseteq \mathrm{fn}(P)$ is $E(P,N) = (|_{m \in \mathrm{fn}(P)\setminus N}\ m[\,Q\,]) \mid Q$, where $Q = |_{n \in \mathrm{fn}(P)}\ (!\mathtt{in}\, n \mid !\mathtt{out}\, n \mid !\mathtt{open}\, n \mid !\overline{\mathtt{in}}\, n \mid !\overline{\mathtt{out}}\, n \mid !\overline{\mathtt{open}}\, n)$.*

We now make sure that a process is protected if it is defended when put in parallel with its enemy.

Theorem 3 *If* $P \mid E(P, N)$ *is* defended *for* $N \subseteq \mathrm{fn}(P)$*, then* $C[P]$ *is* protected *for every* flat *context* $C[-]$ *with no configuration* $n[\,Q\,]$*,* $n \in N$*.*

An Example. Consider an ambient a that wants to send a message D to an ambient b. Assuming that the path from a to b is know, an abstract specification could be found below. (For the sake of simplicity, we assume here that D is a passive datum, that may only occur in the same position of $\mathbf{0}$ in processes, and that requires no analysis.)

$$\begin{array}{lcl} SYS & = & a[\,\overline{\texttt{out}}\,a.\,\overline{\texttt{in}}\,a.\,\texttt{open}\,mail \mid mail[\,S\,]\,] \mid b[\,\overline{\texttt{in}}\,b.\,\texttt{open}\,msg\,] \\ S & = & \texttt{out}\,a.\,\texttt{in}\,b.\,\overline{\texttt{out}}\,mail.\,\texttt{out}\,b.\,\texttt{in}\,a.\,\overline{\texttt{open}}\,mail \mid \\ & & msg[\,\texttt{out}\,mail.\,\overline{\texttt{open}}\,msg.\,D\,] \end{array}$$

Ambient $mail$ goes out of a and carries within b the message, included in msg. When there, msg exits from $mail$ and b reads D through $\texttt{open}\,msg$. Then, $mail$ goes back to a, ready to deliver the ack of b through $\overline{\texttt{open}}\,mail$.

When the message is secret, it is essential to guarantee that no ambient can open msg and therefore read the message, except for the receiver b. Assuming that $\mathcal{T} = \{b, msg\}$ is the set of trustworthy ambients and that all the other ambients, including $\top$, form the complementary set $\mathcal{U}$, we wish to prove that SYS is protected. To prove it, we show that $SYS \mid E(SYS, \{b, msg, mail\})$ is defended. Then, Theorem 3 suffices to prove that $C[SYS]$ is protected in any context where b, msg and $mail$ do not occur (including obviously the empty context).

The solution shown below satisfies the requirements of the static property: msg can be opened by b only, because $\overline{\texttt{open}}\,msg^{(\mathcal{U}\cup\{b\},b)} \in \phi(msg)$; b cannot be opened, because $\overline{\texttt{open}}\,b$ does not occur in $\phi(b)$.

The constraint on the use of name b in the context is necessary for both the dynamic and the static property to hold. Otherwise $mail$ could enter inside another ambient b which can be a *betrayer*, like $b[\,\overline{\texttt{in}}\,b.\,\texttt{open}\,msg.\,\texttt{in}\,e.\,\overline{\texttt{open}}\,b\,]$, which reveals D to the untrustworthy ambient e. By contrast, the restriction on ambients $msg, mail$ in the context is needed to prove the property statically. The analysis cannot actually distinguish different occurrences of ambients with the same name. needless to say, we can prove that, for all contexts C, $C[\boldsymbol{\nu}(msg, mail, b)\,(SYS)]$ is protected.

$\phi(mail)$	$\{\mathtt{out}\, a^{(\mathcal{U},\mathcal{U})}, \mathtt{in}\, b^{(\mathcal{U},\mathcal{U})}, \overline{\mathtt{out}}\, mail^{(\mathcal{U}\cup\{b\},b)}, \mathtt{out}\, b^{(\mathcal{U}\cup\{b\},\mathcal{U}\cup\{b\})},$ $\mathtt{in}\, a^{(\mathcal{U}\cup\{b\},\mathcal{U})}, \overline{\mathtt{open}}\, msg^{(\mathcal{U}\cup\{b\},a)}\} \cup \{msg^{(\mathcal{U}\cup\{b\},\mathcal{U}\cup\{b\})}\}$
$\phi(b)$	$\{\bigcup_{n\in\mathcal{U}} n^{\mathcal{U},\mathcal{U}}, msg^{(\mathcal{U},\mathcal{U})}\}\cup$ $\{\overline{\mathtt{in}}\, b^{(\mathcal{U},\mathcal{U})}, M^{(\mathcal{U}\cup\{b\},\mathcal{U}\cup b\})}, \mathtt{open}\, msg^{(\mathcal{U},\mathcal{U})}\}$
$\phi(msg)$	$\{\mathtt{out}\, mail^{(\mathcal{U},\mathcal{U})}, \overline{\mathtt{open}}\, msg^{(\mathcal{U}\cup\{b\},b)}, M^{(\mathcal{U}\cup\{b\},b)}\}$
$\phi(n)$	$\{\bigcup_{m\in\mathcal{U}\cup\mathcal{T}} m^{(\mathcal{U},\mathcal{U})}\} \cup \{\bigcup_{m\in\mathcal{U}\cup\mathcal{T}} \mu^{(\mathcal{U},\mathcal{U})}, \bar{\mu}^{(\mathcal{U},\mathcal{U})}\},\ n \in \{a, \top\}$

Remark. Now, we can better illustrate the role of our labels ℓ_1 and ℓ_2. If we adapt to SA the analysis of [6], we cannot say that msg may only end up (after the execution of $\mathtt{out}\, mail$) in b and not in $\top$ and a. Hence, we would imprecisely predict that $\overline{\mathtt{open}}\, msg$ may be exerced when msg is inside b and in the untrustworthy ambients $\top$ and a.

References

[1] L. Cardelli, G. Ghelli and A. Gordon *Mobility types for mobile ambients.* Proc. of ICALP'99, LNCS 1644, pp. 230-239. Springer-Verlag, 1999.

[2] L. Cardelli and A. Gordon *Types for mobile ambients.* Proc. of POPL'99, pp 79-92. ACM Press, 1999.

[3] L. Cardelli and G. Ghelli *Ambient Groups and Mobility Types.* Proc. of IFIP TCS'00.

[4] L. Cardelli and A. Gordon. *Mobile Ambients.* Proc. of FoSSaCS'98, LNCS 1378, pp. 140-155. Springer-Verlag, 1998.

[5] F. Levi and D. Sangiorgi *Controlling Interference in Ambients.* Proc. of POPL' 00, pp. 352-364. ACM Press, 2000.

[6] F. Nielson, H.R. Nielson, R.R. Hansen and J.G. Jensen *Validating firewalls in mobile ambients.* Proc. of CONCUR'99, LNCS 1664, pp. 463-477. Springer-Verlag, 1999.

[7] R. R. Hansen, J. G. Jensen, F. Nielson, H. R.Nielson *Abstract Interpretation of Mobile Ambients.* Proc. of SAS'99, LNCS 1694. Springer-Verlag, 1999, pp. 135-148.

[8] H. R. Nielson and F. Nielson *Shape Analysis for Mobile Ambients.* Proc. of POPL'00, pp. 135-148, ACM Press, 2000.