Monitoring Confidentiality by Diagnosis Techniques

Jérémy Dubreil, Thierry Jéron, Hervé Marchand

Monitoring Confidentiality by Diagnosis Techniques

Jérémy Dubreil, Thierry Jéron, Hervé Marchand
INRIA Rennes Bretagne-Atlantique, Email: {first.last}@irisa.fr

***Abstract*—We are interested in constructing monitors for the detection of confidential information flow in the context of partially observable discrete event systems. We focus on the case where the secret information is given as a regular language. We first characterize the set of observations allowing an attacker to infer the secret information. Further, based on the diagnosis of discrete event systems, we provide necessary and sufficient conditions under which detection and prediction of secret information flow can be ensured, and construct a monitor allowing an administrator to detect it.**

I. Introduction

There has been an increasing interest in research about computer security in the past decades. Indeed, the emergence of web services and the improvements of the possibilities of mobile and embedded systems allow lots of new and interesting features. But some of these services such as online payment, medical information storage or e-voting system may deal with some critical information. In the meantime, having more applications and devices for accessing these services also increases the possibilities for such information to flow. To avoid security breach, using automatic tools based on formal methods for security analysis can be beneficial. In this context, there has been a growing interest in verification [1], [2] and testing of security properties [3] in past years. In order to specify such automatic analysis methods, security properties are generally separated into three different categories: *availability* (a user can always perform the actions that are allowed by the security policy), *integrity* (something illegal cannot be performed by a user) and *confidentiality* (some secret information cannot be inferred by a user) [4].

In this paper, we focus on confidentiality and more particularly on the notion of opacity as defined in [4]. The general problem of confidentiality consists in determining whether an attacker, having only a partial observations of the system, is able or not to discover some secret behaviors (e.g. a password stored in a file, the value of some hidden variables, etc) occurring during execution. The motivation of this paper is to provide an analysis method for detecting information flows. Therefore we proceed first from an attacker point of view, for generating the set of possible attacks, and second from the administrator point of view interested in monitoring those attacks.

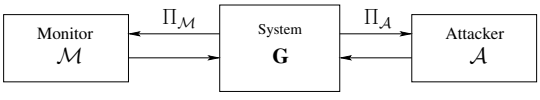


Fig. 1. Architecture

Overview of the problem. We consider three components: a system G, an attacker $\mathcal{A}$ and a monitor $\mathcal{M}$ (modelling for example the administrator of the system). We assume that the system G is modeled by a finite transition system. Users interact with G through an interface, corresponding to the inputs/outputs of the system given as a function $\Pi_{\mathcal{A}}$. For this system, one can define some confidentiality policies. Following the approach [5], [6], a secret is modeled by a property φ given as a regular language over the alphabet Σ of the system. The secret is preserved as far as the attacker cannot surely infer that the property φ is satisfied by the current execution of the system based on the observations performed through the interface $\Pi_{\mathcal{A}}$. We characterize the set of observations allowing the attacker $\mathcal{A}$ to infer the secret information. *A contrario*, the monitor $\mathcal{M}$ tries to analyze the information flow between the system G and the attacker $\mathcal{A}$ in order to raise an alarm whenever the secret has been revealed. To do so, we assume that $\mathcal{M}$ knows the power of the attacker (i.e. he knows the model of the system G and the interface $\Pi_{\mathcal{A}}$ of the attacker). He observes the system through the interface $\Pi_{\mathcal{M}}$ (we do not assume any link between the two interfaces). Further, based on the set of observations allowing the attacker to infer the secret information, we provide necessary and sufficient conditions under which detection and prediction of secret information flow can be ensured, and construct a monitor $\mathcal{M}$ allowing an administrator to detect the attacks. This supervision is performed on-line, the monitor raising an alarm whenever an information flow occurs.

The structure of the document is as follows: Section II introduces the mathematical terminology and notions used throughout the paper. In Section III, we show how to build a monitor in charge of the supervision of the system according to a given property. Section IV defines the notion of opacity formalizing information flow. With this notion, we can characterize the set of observations for which an attacker can infer confidential information. In Section V, we use diagnosis techniques to exhibit necessary and sufficient conditions under which a monitor can diagnose or predict the information flow.

II. Models & Notations

Let Σ be a finite alphabet of events. A *string* is a finite-length sequence of events in Σ. ϵ denotes the empty string. Given a string s, the length of s is denoted by $|s|$. The set of all strings formed by events in Σ is denoted by Σ^*. Any subset of Σ^* is called a *language* over Σ. Let L be a language over Σ. Given a string $s \in L$, $L/s \triangleq \{t \in \Sigma^* \mid s.t \in L\}$ is called the *post-language* of L after s and defined as L/s. L is said to be *extention-closed* when $L.\Sigma^* = L$. We assume

that the systems are modeled as Labelled Transitions Systems (LTS for short). The formal definition of an LTS is as follows.

Definition 1 (LTS): An LTS over Σ is defined by a 4-tuple $G = (Q_G, \Sigma, \rightarrow_G, q_G^0)$ where Q_G is a finite set of states, Σ is the set of events *of G, $q_G^0 \in Q_G$ is the initial state, and $\rightarrow_G \subseteq Q_G \times \Sigma \times Q_G$ is* the partial transition relation. $\diamond$

Notations. In the remainder of this section, we consider a given LTS $G = (Q_{\mathrm{G}}, \Sigma, \rightarrow_{\mathrm{G}}, q_{\mathrm{G}}^0)$. We write $q \xrightarrow{a}_G q'$ if $(q, a, q') \in \rightarrow_G$ and $q \xrightarrow{a}_G$ for $\exists q' \in Q_G$, $q \xrightarrow{a}_G q'$. We extend $\rightarrow_{\mathrm{G}}$ to arbitrary sequences by setting: $q \xrightarrow{\varepsilon}_{\mathrm{G}} q$ for all states q, and $q \xrightarrow{s\sigma}_{\mathrm{G}} q'$ whenever $q \xrightarrow{s}_{\mathrm{G}} q''$ and $q'' \xrightarrow{\sigma}_{\mathrm{G}} q'$, for some $q'' \in Q_{\mathrm{G}}$. $\Sigma(q) \triangleq \{a \in \Sigma \mid q \xrightarrow{a}_{\mathrm{G}}\}$ corresponds to the set of events admissible in state q of G. G is said to be *complete* whenever $\forall q \in Q_{\mathrm{G}}, \Sigma(q) = \Sigma$. It is said to be *live* if $\Sigma(q) \neq \emptyset$, for each $q \in Q_{\mathrm{G}}$. We set $\Delta_{\mathrm{G}}(q, l) \triangleq \{q' \in Q_{\mathrm{G}} \mid q \xrightarrow{l}_{\mathrm{G}} q'\}$. By a slight abuse of notation, for any language $L \subseteq \Sigma^*$, $\Delta_{\mathrm{G}}(q, L) \triangleq \{q' \in Q_{\mathrm{G}} \mid \exists s \in L, q \xrightarrow{s}_{\mathrm{G}} q'\}$. For any $X \subseteq Q_{\mathrm{G}}$, $\Delta_{\mathrm{G}}(X, L) = \bigcup_{q \in X} \Delta_{\mathrm{G}}(q, L)$. Also, X is said to be *stable* if $\Delta_{\mathrm{G}}(X, \Sigma^*) \subseteq X$. We denote by $L(G) = \{l \in \Sigma^*, q_o \xrightarrow{l}_{\mathrm{G}}\}$ the set of trajectories of the system G.

Given a special set of states $F_{\mathrm{G}} \subseteq Q_{\mathrm{G}}$, the notions above are extended in this setting by letting the language $L_{F_{\mathrm{G}}}(G) = \{l \in \Sigma^* \mid \exists q \in F_{\mathrm{G}}, q_o \xrightarrow{l}_{\mathrm{G}} q\}$ be the set of trajectories that end in a state of F_G. Note that F_G is stable if $L_{F_{\mathrm{G}}}(G)$ is extention-closed. Also, if G is complete and F_G is stable, then $L_{F_{\mathrm{G}}}(G)$ is extention-closed.

Definition 2: (Synchronous product) Let $G^i = (Q^i, \Sigma, \rightarrow_{G^i}, q_{G^i}^0)$, $i = 1, 2$ be two LTSs. The synchronous product *between G^1 and G^2 is an LTS $G^1 \times G^2 = (Q^1 \times Q^2, \Sigma, \rightarrow_{G^1 \times G^2}, (q_{G^1}^0, q_{G^2}^0))$, where $(q^1, q^2) \xrightarrow{\sigma}_{G^1 \times G^2} (q'^1, q'^2)$ whenever $q^1 \xrightarrow{\sigma}_{G^1} q'^1$ and $q^2 \xrightarrow{\sigma}_{G^2} q'^2$.*

Clearly, $L(G^1 \times G^2) = L(G^1) \cap L(G^2)$ and for $F_i \subseteq Q^i, i = 1, 2$, we also have $L_{F_1 \times F_2}(G^1 \times G^2) = L_{F_1}(G^1) \cap L_{F_2}(G^2)$. Also, if for $i = 1, 2$ the set F_i is stable in G^i, $F_1 \times F_2$ is stable in $G^1 \times G^2$.

Given a set of states $E \subseteq Q_{\mathrm{G}}$ of an LTS G, the operators $pre_{\mathrm{G}}^{\forall}$ et $pre_{\mathrm{G}}^{\exists}$ are defined as follows:

$$Pre_{\mathrm{G}}^{\exists}(E) = \{q \in Q \mid \exists a \in \Sigma, \Delta_{\mathrm{G}}(q, a) \cap E \neq \emptyset\}$$
$$Pre_{\mathrm{G}}^{\forall}(E) = \{q \in Pre_{\mathrm{G}}^{\exists}(E) \mid \forall a \in \Sigma, \Delta_{\mathrm{G}}(q, a) \subseteq E\}$$

The states belonging to $Pre_{\mathrm{G}}^{\forall}(E)$ are the states such that all immediate successors belong to E, while the states belonging to $Pre_{\mathrm{M}}^{\exists}(E)$ are such that at least one immediate successor belongs to E.

Given a live LTS G, let $Inev_{\mathrm{G}}(E)$ be the set of states that inevitably lead to a set E in a finite number of steps and $CoReach_{\mathrm{G}}(E)$ the set of states from which E is reachable. These sets are given by the following least fix-points (lfp):

$$Inev_{\mathrm{G}}(E) = lfp(\lambda X.E \cup pre_G^{\forall}(X))$$
$$CoReach_{\mathrm{G}}(E) = lfp(\lambda X.E \cup pre_G^{\exists}(X))$$

Observable behavior. The key point of our approach concerns the ability of an user $\mathcal{U}$ to deduce information from a system by observing only a subset of the events or only an abstraction of them. For this purpose, we introduce the concept of *observation mask* modeling the interface between a user and the system. An observation mask is a function $\Pi_{\mathcal{U}} : \Sigma \rightarrow \Sigma_{\mathcal{U}} \cup \{\epsilon\}$, where $\Pi_{\mathcal{U}}$ is defined for all $\sigma \in \Sigma$. The set $\Sigma_{\mathcal{U}}$ is another event set called the *observed events*. We denote by $\Sigma_{\mathcal{U}}^{-1} = \{\sigma \in \Sigma \mid \Pi_{\mathcal{U}}(\sigma) \neq \epsilon\}$ the set of *observable events*, i.e. the events of Σ inducing an observation for $\mathcal{U}$. The observation mask is extended to any trajectory by assigning $\Pi_{\mathcal{U}}(\epsilon) = \epsilon$ and $\forall s \in \Sigma^*, \sigma \in \Sigma$, $\Pi_{\mathcal{U}}(s\sigma) = \Pi_{\mathcal{U}}(s)\Pi_{\mathcal{U}}(\sigma)$. This is further extended to any language $L \subseteq \Sigma^*$ by letting $\Pi_{\mathcal{U}}(L) = \{\Pi_{\mathcal{U}}(s) \mid s \in L\}$. The inverse observation mask for $T \subseteq \Sigma_{\mathcal{U}}^*$ is given by:

$$\Pi_{\mathcal{U}}^{-1}(T) = \{l \in \Sigma^* \mid \Pi_{\mathcal{U}}(l) \in T\}.$$

We say that G is $\Sigma_{\mathcal{U}}$*-live* if $\forall q \in Q, \exists s \in \Sigma^*, \sigma \in \Sigma_{\mathcal{U}}^{-1}, q \xrightarrow{s\sigma}$, meaning that there is no terminal loop of events that cannot be observed by the observation mask.

Starting from G and a set of observable events $\Sigma_{\mathcal{U}}$, the set of *observed traces* of G is given by $\mathcal{T}_{\mathcal{U}}(G) = \Pi_{\mathcal{U}}(L(G))$.

We define the *semantic* $[\![\mu]\!]_{\mathcal{U}}$ of a trace $\mu \in \mathcal{T}(G)$ as the set of trajectories of G that are *compatible* with the trace μ:

$$[\![\mu]\!]_{\mathcal{U}} \triangleq \begin{cases} \Pi_{\mathcal{U}}^{-1}(\mu) \cap L(G) \cap \Sigma^* \Sigma_{\mathcal{U}}^{-1} \text{ if } \mu \neq \epsilon \\ \{\epsilon\} \text{ otherwise.} \end{cases}$$

This means that (except for the empty trace), trajectories compatible with a trace μ are trajectories of G ending with an observable event and having trace μ. This is consistent with an on-line observation performed by a user of the system for whom the system is only seen through the interface given by the observation mask $\Pi_{\mathcal{U}}$ when we suppose that the observers are reacting faster than the system.

An LTS G is said to be *deterministic* if for all $q \in Q_{\mathrm{G}}$, for all $a \in \Sigma$, $q \xrightarrow{a}_{\mathrm{G}} q'$ and $q \xrightarrow{a}_{\mathrm{G}} q''$ implies $q' = q''$.

In order to build monitors in charge of the observation of the system, we need to build, starting from a non-deterministic LTS G, a deterministic LTS $Det_{\mathcal{U}}(G)$ over the alphabet $\Sigma_{\mathcal{U}}$ preserving the set of traces, i.e. $L(Det_{\mathcal{U}}(G)) = \mathcal{T}_{\mathcal{U}}(G)$.

Definition 3: Let $G = (Q_G, \Sigma, \rightarrow_G, q_G^0)$ be an LTS and $\Pi_{\mathcal{U}}$ an observation mask. The determinization *of G w.r.t. $\Pi_{\mathcal{U}}$ is the LTS $Det_{\mathcal{U}}(G) = (\mathcal{X}, \Sigma_{\mathcal{U}}, \rightarrow_d, X^0)$ where $\mathcal{X} = 2^{Q_G}$ (the set of subsets of Q_G called* macro-states*), $X^0 = \{q_G^0\}$ and $\rightarrow_d = \{(X, \Pi_{\mathcal{U}}(a), \Delta_G(X, (\Sigma \setminus \Sigma_{\mathcal{U}}^{-1})^*.a) \mid X \in \mathcal{X}$ and $a \in \Sigma_{\mathcal{U}}^{-1}\}$.*

Notice that this definition is consistent with the above semantic of observations $[\![.]\!]_{\mathcal{U}}$: the target macro-state X' of a transition $X \xrightarrow{\sigma}_d X'$ is composed of the set of states q' of G which are targets of sequences of transitions $q \xrightarrow{s.a} q'$ ending with an observable event a such that $\Pi_{\mathcal{U}}(a) = \sigma$, with $q \in X$. From the definition of $\rightarrow_d$, we get $\Delta_{Det_{\mathcal{U}}(G)}(X^0, \mu) = \{\Delta_{\mathrm{G}}(q_{\mathrm{G}}^0, [\![\mu]\!]_{\mathcal{U}})\}$. This means that a macro-state that is reached from X^0 by μ in $Det_{\mathcal{U}}(G)$ is composed of states that are reached from q_{G}^0 by trajectories of $[\![\mu]\!]_{\mathcal{U}}$ in G.

III. Inference of properties

In this section, we consider a user $\mathcal{U}$ interacting with a system modeled by a LTS G through an interface modeled by

an observation mask $\Pi_{\mathcal{U}}$. We consider properties modeled by regular languages over Σ that are defined as follows.

Definition 4: A property is given by a marked language $L_{F_\psi}(\psi) \subseteq \Sigma^*$ of a complete and deterministic LTS $\psi = (Q_\psi, \Sigma, \rightarrow_\psi, q^0_\psi)$ equipped with a distinguished set F_ψ.

We say that a trajectory $s \in L(G)$ is recognized by ψ, noted $s \models \psi$ whenever $s \in L_{F_\psi}(\psi)$. As ψ is complete, we get $L(G \times \psi) = L(G)$ and $L_{Q^G \times F_\psi}(G \times \psi) = L(G) \cap L_{F_\psi}(\psi)$ is the set of trajectories of G satisfying ψ.

Let $s \in L(G)$ be a trajectory that has been triggered by the system. The user $\mathcal{U}$ aims to infer whether s satisfies the property ψ by observing $\mu = \Pi_{\mathcal{U}}(s) \in \mathcal{T}_{\mathcal{U}}(G)$. However, the user cannot distinguish s from any trajectory $s' \in [\![\mu]\!]_{\mathcal{U}}$ compatible with the observation μ. Thus, $\mathcal{U}$ can only infer partial information regarding $s \models \psi$ from $[\![\mu]\!]_{\mathcal{U}}$. Especially, $\mathcal{U}$ is sure that $s \models \psi$ if $[\![\mu]\!]_{\mathcal{U}} \subseteq L_{F_\psi}(\psi)$. Meanwhile, if there exists $s' \in [\![\mu]\!]_{\mathcal{U}}$ and $s' \not\models \psi$, then it is impossible for $\mathcal{U}$ to know if the current trajectory is s or s' and then $\mathcal{U}$ cannot infer whether $s \models \psi$ or not. To go further, $\mathcal{U}$ might be also interested in the fact that after observing μ, ψ will be inevitably satisfied, or will not be satisfied anymore by the trajectories of G extending s.

Next, we formalize these ideas and propose a way to build a function $\mathcal{O}^\psi_{\mathcal{U}}$, inspired by [7], which captures, for each observation $\mu \in \mathcal{T}(G)$ what a user $\mathcal{U}$ can infer about s and ψ. Formally, if s is the current execution of the system and $\mu = \Pi_{\mathcal{U}}(s)$ is the corresponding observation, the verdicts we are interested in are defined by the following function:

$$\mathcal{O}^\psi_{\mathcal{U}} : \Sigma^*_{\mathcal{U}} \rightarrow V = \{Yes, Inev, Inev\_Yes, Never, No, ?\}$$

where the semantic of the verdicts is as follows:

1) $\mathcal{O}^\psi_{\mathcal{U}}(\mu) = Yes$ if $\mathcal{U}$ knows that for the current execution s (s.t. $\Pi_{\mathcal{U}}(s) = \mu$), $s \models \psi$;
2) $\mathcal{O}_{\mathcal{U}}(\mu) = Inev$ if $\mathcal{U}$ knows that $s \not\models \psi$ but also that ψ will eventually be satisfied by all the possible extension of s;
3) $\mathcal{O}^\psi_{\mathcal{U}}(\mu) = Inev\_Yes$ if $\mathcal{U}$ knows that $s \models \psi$ or that ψ will inevitably be satisfied in the future but cannot distinguish between the two cases so far
4) $\mathcal{O}^\psi_{\mathcal{U}}(\mu) = Never$ if $\mathcal{U}$ knows that ψ will never be satisfied by the executions of G extending s;
5) $\mathcal{O}^\psi_{\mathcal{U}}(\mu) = No$ if $\mathcal{U}$ knows that $s \not\models \psi$, but ψ is neither unavoidable nor impossible;
6) $\mathcal{O}^\psi_{\mathcal{U}}(\mu) = \ ?$ in all the other cases, meaning that $\mathcal{U}$ cannot infer any useful information with regards to s and ψ after the observation $\mu = \Pi_{\mathcal{U}}(s)$.

A. Construction of $\mathcal{O}^\psi_{\mathcal{U}}$

In this section, we now explain how to construct the function $\mathcal{O}^\psi_{\mathcal{U}} : \Sigma^*_{\mathcal{U}} \rightarrow V$:

Step 1. Construct the synchronous product $G_\psi = G \times \psi = (Q_{G_\psi}, \Sigma, \rightarrow_{G_\psi}, q^0_{G_\psi})$ as well as the set of final states $F_{G_\psi} = Q^G \times F_\psi$. By the property of the synchronous product, and using the fact that ψ is complete, we get $L(G_\psi) = L(G)$ and $L_{F_{G_\psi}}(G_\psi) = L(G) \cap L_{F_\psi}(\psi)$. Thus, the accepted trajectories of G_ψ in F_{G_ψ}, $L_{F_{G_\psi}}(G_\psi)$, are exactly the trajectories of G accepted by ψ.

Step 2. Compute $Inev_{G_\psi}(F_{G_\psi})$ on G_ψ and consider the following partition: $Q_{G_\psi} = F_{G_\psi} \cup I_{G_\psi} \cup P_{G_\psi} \cup N_{G_\psi}$, where

- $I_{G_\psi} = Inev_{G_\psi}(F_{G_\psi}) \setminus F_{G_\psi}$ is the set of states not belonging to F_{G_ψ} but from which F_{G_ψ} is unavoidable;
- $P_{G_\psi} = Q_{G_\psi} \setminus CoReach_{G_\psi}(F_{G_\psi})$, i.e. the set of states from which F_{G_ψ} is unreachable;
- $N_{G_\psi} = Q_{G_\psi} \setminus (F_{G_\psi} \cup I_{G_\psi} \cup P_{G_\psi})$ is the set of all other states.

Step 3. Build $\chi^\psi_{\mathcal{U}}(G) = Det_{\mathcal{U}}(G_\psi) = (\mathcal{X}, \Sigma_{\mathcal{U}}, \rightarrow_d, X^0)$. We thus have $L(\chi^\psi_{\mathcal{U}}(G)) = \mathcal{T}_{\mathcal{U}}(G)$. For each observation $\mu \in \mathcal{T}_{\mathcal{U}}(G)$, we get $\Delta_{\chi^\psi_{\mathcal{U}}(G)}(X^0, \mu) = \{\Delta_{G_\psi}(q^0_{G_\psi}, [\![\mu]\!]_{\mathcal{U}})\}$.

Step 4. We finally compute the function $\mathcal{O}^\psi_{\mathcal{U}}$ from $\chi^\psi_{\mathcal{U}}(G)$ and the sets $F_{G_\psi}, I_{G_\psi}, P_{G_\psi}, N_{G_\psi}$ as follows: $\forall \mu \in \mathcal{T}_{\mathcal{U}}(G)$,

$$\mathcal{O}^\psi_{\mathcal{U}}(\mu) = \begin{cases} Yes, & \text{if } \Delta_{\chi^\psi_{\mathcal{U}}(G)}(X^0, \mu) \subseteq F_{G_\psi} \\ Inev, & \text{if } \Delta_{\chi^\psi_{\mathcal{U}}(G)}(X^0, \mu) \subseteq I_{G_\psi} \\ Inev\_Yes, & \text{if } \Delta_{\chi^\psi_{\mathcal{U}}(G)}(X^0, \mu) \subseteq (I_{G_\psi} \cup F_{G_\psi}) \\ & \quad \wedge\ \Delta_{\chi^\psi_{\mathcal{U}}(G)}(X^0, \mu) \cap I_{G_\psi} \neq \emptyset \\ & \quad \wedge\ \Delta_{\chi^\psi_{\mathcal{U}}(G)}(X^0, \mu) \cap F_{G_\psi} \neq \emptyset \\ No, & \text{if } \Delta_{\chi^\psi_{\mathcal{U}}(G)}(X^0, \mu) \subseteq N_{G_\psi} \\ Never & \text{if } \Delta_{\chi^\psi_{\mathcal{U}}(G)}(X^0, \mu) \subseteq P_{G_\psi} \\ ? & \text{otherwise.} \end{cases}$$

It is easy to check that the construction of $\mathcal{O}^\psi_{\mathcal{U}}$ conforms to the informal definition previously introduced. For example, for the verdict Yes, consider an execution $s \in L(G)$ together with its corresponding observation $\mu = \Pi_{\mathcal{U}}(s)$ and $\mathcal{O}^\psi_{\mathcal{U}}(\mu) = Yes$. We thus have $\Delta_{\chi^\psi_{\mathcal{U}}(G)}(X^0, \mu) \subseteq F_{G_\psi}$. Now, according to the definition of $\chi^\psi_{\mathcal{U}}(G)$, for all $s' \in [\![\mu]\!]_{\mathcal{U}}$, $\Delta_{S_\psi}(q^0_{G_\psi}, s') \subseteq \Delta_{\chi^\psi_{\mathcal{U}}(G)}(X^0, \mu) \subseteq F_{G_\psi}$, thus $s' \models \psi$. Hence, for all trajectories $s' \in [\![\mu]\!]_{\mathcal{U}}$, $s' \models \psi$ and in particular $s \models \psi$. Similarly for $\mathcal{O}^\psi_{\mathcal{U}}(\mu) = Inev$. It implies that $\Delta_{S_\psi}(q^0_{G_\psi}, [\![\mu]\!]_{\mathcal{U}}) \subseteq I_{G_\psi}$. Then, the trajectories in $[\![\mu]\!]_{\mathcal{U}}$ are for sure not satisfying ψ and all their continuations will evetually satisfy ψ. Then this also holds for s.

To conclude this section, given a system G that is observed by a user $\mathcal{U}$ through the interface $\Pi_{\mathcal{U}}$, we know how to construct a function $\mathcal{O}^\psi_{\mathcal{U}} : \Sigma^*_{\mathcal{U}} \rightarrow V$ that gives access to all the information that the user $\mathcal{U}$ can deduce with respect to the executions of G and the property ψ.

IV. Characterization and verification of opacity

Assume now that the attacker $\mathcal{A}$ is a user of a system G trying to infer confidential information. We assume that the attacker perfectly knows the model of G, but only observes it through the interface $\Pi_{\mathcal{A}}$. We consider a secret φ given by a marked language of a complete deterministic LTS, $\varphi = (Q_\varphi, q^0_\varphi, \Sigma, \rightarrow, F_\varphi)$. We assume that $\mathcal{A}$ knows how to build an observational function as described in the preceding section and our aim is to know if the attacker can know that the current execution $s \in L(G)$ reveals the secret φ.

Example 1: Let G be a LTS with $\Sigma = \{\tau, \tau_\varphi, a, b, c\}$, $\Sigma_{\mathcal{A}} = \{a, b, c\}$ (the observation mask is reduced to the natural

projection). The secret under consideration is the occurrence of the event τ_φ. This should not be revealed to the users of the system, knowing that τ_φ is not observable. However, users

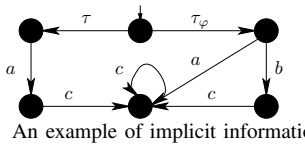


Fig. 2. An example of implicit information dlow

can infer that τ_φ has occurred by observing the event b. Such a system is not secure because the fact that τ_φ occurs during execution is modifying what $\mathcal{A}$ can observe. However, for a different mask, e.g. $\Pi_\mathcal{A}(a) = \Pi_\mathcal{A}(b)$, the occurrence of p does not change the observations and G is safe.

A. Definition of Opacity

Intuitively, a secret φ is said to be opaque with respect to a system G and a mask $\Pi_\mathcal{A}$ if the attacker $\mathcal{A}$ can never be sure that the current execution of G satisfy φ [5], [6], [8].

Definition 5: [Opacity] Given a system G and a secret φ, φ is opaque w.r.t. G and $\Pi_\mathcal{A}$ if $\forall s \in L(S), [\![\Pi_\mathcal{A}(s)]\!]_\mathcal{A} \not\subseteq L_{F_\varphi}(\varphi)$. In other words, φ is opaque w.r.t. G and $\Pi_\mathcal{A}$ if and only if $\forall \mu \in \mathcal{T}_\mathcal{A}(G),\ [\![\mu]\!]_\mathcal{A} \not\subseteq L_{F_\varphi}(\varphi)$, and φ is non-opaque w.r.t. G and $\Pi_\mathcal{A}$ if and only if $\exists \mu \in \mathcal{T}_\mathcal{A}(G),\ [\![\mu]\!]_\mathcal{A} \subseteq L_{F_\varphi}(\varphi)$. Based on the semantics of $\mathcal{O}^\varphi_\mathcal{A}$ described in the preceding section, one can say that φ is opaque with respect to G and $\Pi_\mathcal{A}$ if

$$\forall s \in L(G),\ \mathcal{O}^\varphi_\mathcal{A}(\Pi_\mathcal{A}(s)) \neq Yes$$

B. Verification of Opacity

In this section, we are interested in checking whether a secret φ is opaque with respect to a system G and an interface $\Pi_\mathcal{A}$. This is a particular case of the inference of property presented in Section III. To do so, consider $\chi^\varphi_\mathcal{A}(G) = Det_\mathcal{A}(G \times \varphi) = (\mathcal{X}, \Sigma_\mathcal{A}, \rightarrow_d, X^0)$ equipped with the set of final states $F = 2^{Q^G \times F_\varphi}$. By construction of $\chi^\varphi_\mathcal{A}(G)$, we get

$$[\![L_F(\chi^\varphi_\mathcal{A}(G))]\!]_\mathcal{A} = \{s \in L(S) \cap \Sigma.\Sigma^{-1}_\mathcal{A} \mid [\![\Pi_\mathcal{A}(s)]\!]_\mathcal{A} \subseteq L_{F_\varphi}(\varphi)\}$$

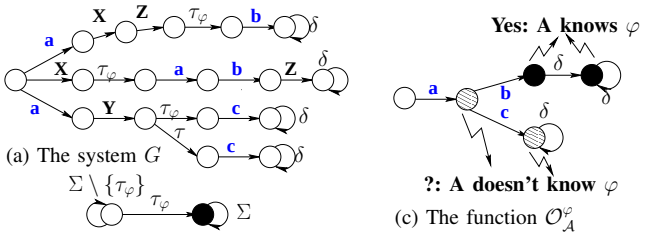
which gives a characterization of opacity:

Proposition 1: φ is opaque with respect to G and the interface $\Pi_\mathcal{A}$ if and only if $L_F(\chi^\varphi_\mathcal{A}(G)) = \emptyset$. ◇

Hence, checking the opacity of a secret φ consists of checking that the set of states F is not reachable in $\chi^\varphi_\mathcal{A}(G)$. If it is reachable, then φ is not opaque and there exists at least one observation allowing the attacker to infer that φ is satisfied. In other words, $L_F(\chi^\varphi_\mathcal{A}(G))$ corresponds to the set of observations for which the attacker $\mathcal{A}$ knows that the current execution reveals φ. In that case, the attacker $\mathcal{A}$, based on the preceding techniques, can compute the LTS $\chi^\varphi_\mathcal{A}(G)$ and deduce an observation function $\mathcal{O}^\varphi_\mathcal{A}$ such that, for a given observation μ of $\mathcal{T}(S)$:

- if $\mathcal{O}^\varphi_\mathcal{A}(\mu) = Yes$, then $\mu \in L_F(\chi^\varphi_\mathcal{A}(G))$ and $[\![\mu]\!]_\mathcal{A} \subseteq L_{F_\varphi}(\varphi)$; the attacker, based on this observation, can deduce that φ is satisfied on G and there is an information flow;
- if $\mathcal{O}^\varphi_\mathcal{A}(\mu) = ?_\mathcal{A}$, $\mathcal{A}$ cannot deduce φ and there is no information flow, where $?_\mathcal{A} = \{No, Inev, Inev\_Yes, Never, ?\}$[1].

Example 2: Consider the system G described in Fig. 3 (a). The alphabet of G is $\Sigma = \{a, b, c, X, Y, Z, \tau_\varphi, \tau, \delta\}$. We assume here that the secret property is given by the LTS described in Fig. 3 (b); The marked state is represented by the black state. In this example, the attacker tries to infer the occurrence of the event τ_φ in the system.



(a) The system G

(b) The opacity property φ

(c) The function $\mathcal{O}^\varphi_\mathcal{A}$

Fig. 3. G, φ and the function $\mathcal{O}^\varphi_\mathcal{A}$ based on $\chi^\mathcal{A}_\varphi$

The observation mask is here reduced to the natural projection. The interface of the attacker is reduced to $\Sigma_\mathcal{A} = \{a, b, c, \delta\}$. The observer $\mathcal{O}^\varphi_\mathcal{A}$ that the attacker $\mathcal{A}$ can build is given by the LTS depicted in Fig. 3(c). If $\mathcal{A}$ observes $a.b.\delta^*$ then φ is revealed $\mathcal{A}$ is then sure that the event τ_φ occurred in $\mathcal{S}$ (the set of compatible trajectories is $a.X.Z.\tau_\varphi.b.\delta^*$ and $X.\tau_\varphi.a.b.Z.\delta^*$). A contrario, if $\mathcal{A}$ simply observes a or $a.c.\delta^*$, then he is not sure that φ is satisfied or not. Some of the compatible trajectories satisfy the secret and some other do not, thus $\mathcal{A}$ cannot infer the secret.

Remark 1: Wihin our framework, it is also possible to consider other kinds of opacity, like the one introduced in [5]). See [9] for details) ◇

V. Monitoring Opacity

Given a secret φ, based on the techniques described in the preceding sections, it is possible to check whether φ is opaque w.r.t. G and the interface $\Pi_\mathcal{A}$. When φ is not opaque, it can be important for an administrator to supervise the system online by means of a monitor $\mathcal{M}$ and raise an alarm as soon as an information flow occurs. For this, we assume that $\mathcal{M}$ knows the model of the system G and observes it through the interface $\Pi_\mathcal{M}$. Moreover, $\mathcal{M}$ knows the ability of the attacker $\mathcal{A}$, meaning that the monitor knows that $\mathcal{A}$ observes the system via the interface $\Pi_\mathcal{A}$ and that he can construct an observation function $\mathcal{O}^\varphi_\mathcal{A}$. We do not assume any relation between $\Pi_\mathcal{A}$ and $\Pi_\mathcal{M}$. Thus, $\mathcal{M}$ has to infer the attacker's knowledge based on the observation of $\mathcal{T}_\mathcal{M}(G) \subseteq \Sigma^*_\mathcal{M}$.

If φ is not opaque w.r.t. the system G and the interface $\Pi_\mathcal{A}$, an administrator can build an observation function to diagnose the fact that the secret has been revealed. One can also be more accurate and try to predict the fact that the secret will be inevitably known by the attacker strictly before the information flow, or that the secret will never be revealed anymore.

[1]Compared with (1), we consider here that the attacker $\mathcal{A}$ is only interested by the detection of the satisfaction of the secret.

Note that it is not necessary to diagnose the fact that the system performed a sequence satisfying the secret if this sequence does not correspond to a non-opaque execution (this sequence does not reveal anything to the attacker); only the executions that lead to an information flow have to be taken into account. Indeed, the secret φ is revealed to the attacker by an execution $s \in L(G)$ if and only if $\Pi_{\mathcal{A}}(s) \in L_F(\chi^{\varphi}_{\mathcal{A}}(G))$. In other words, we are interested in diagnosing the property: "The secret φ has been revealed to the attacker", which corresponds to the extention-closed language: $\Pi^{-1}_{\mathcal{A}}(L_F(\chi^{\varphi}_{\mathcal{A}}(G))) \cdot \Sigma^*$. This language can be recognized by an LTS Ω, equipped with a set of final states F_Ω such that:

$$\mathcal{L}_{F_\Omega}(\Omega) = \Pi^{-1}_{\mathcal{A}}(L_F(\chi^{\varphi}_{\mathcal{A}}(G))) \cdot \Sigma^* \quad (1)$$

Example 3: To illustrate the computation of (1), let us come back to Example 2. The corresponding LTS Ω is shown in Fig. 3

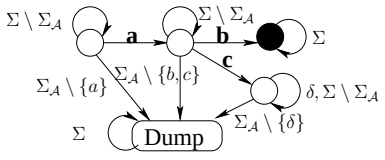


Fig. 4. The LTS Ω computed from $\chi^{\varphi}_{\mathcal{A}}(G))$

A. Supervision of Information Flow

Given a system G, an attacker $\mathcal{A}$ observing G via the interface $\Pi_{\mathcal{A}}$ and a secret φ (that we assume to be non-opaque), we describe now a method allowing an administrator $\mathcal{M}$ observing G via the interface $\Pi_{\mathcal{M}}$ to know whether there is an information flow or not. We assume that the monitor in charge of the supervision has a full knowledge of G and knows the observation mask $\Pi_{\mathcal{A}}$.

As mentioned in the introduction of this section, $\mathcal{M}$ does not directly observe φ. Only the trajectories causing an information flow have to be supervised. We consider then the stable property Ω corresponding to the trajectories of G inducing an information flow from G to $\mathcal{A}$ (see (1)).

In order to construct the observer $\mathcal{O}^{\Omega}_{\mathcal{M}}$ in charge of the supervision of Ω (i.e. corresponding to the information leak of φ), we first build $G_\Omega = G \times \Omega$ and the sets F_{G_Ω}, I_{G_Ω}, P_{G_Ω}, N_{G_Ω} (as described in Step 2., Section III-A).

Now, based on the techniques of the section III-A, one can compute the LTS $\chi^{\Omega}_{\mathcal{M}}(G)$ over $\Sigma_{\mathcal{M}}$ from which we can derive an observer $\mathcal{O}^{\Omega}_{\mathcal{M}}$ with the following verdicts: for $\mu \in \mathcal{T}_{\mathcal{M}}(G)$,

- $\mathcal{O}^{\Omega}_{\mathcal{M}}(\mu) = Yes$: $\mathcal{M}$ infers that Ω is satisfied and thus can deduce that $\mathcal{A}$ knows φ;
- $\mathcal{O}^{\Omega}_{\mathcal{M}}(\mu) = No$: $\mathcal{M}$ knows that $\mathcal{A}$ does not know φ but might know it in the future;
- $\mathcal{O}^{\Omega}_{\mathcal{M}}(\mu) = Inev$: $\mathcal{M}$ knows that $\mathcal{A}$ will inevitably know φ but does not know it yet;
- $\mathcal{O}^{\Omega}_{\mathcal{M}}(\mu) = Inev\_Yes$: $\mathcal{M}$ knows that $\mathcal{A}$ already knows or will know φ;
- $\mathcal{O}^{\Omega}_{\mathcal{M}}(\mu) = Never$: $\mathcal{M}$ knows that $\mathcal{A}$ will never know φ.
- $\mathcal{O}^{\Omega}_{\mathcal{M}}(\mu) =?$ means that $\mathcal{M}$ cannot deduce anything about the knowledge of $\mathcal{A}$.

Unfortunately, the case $\mathcal{O}^{\Omega}_{\mathcal{M}}(\mu) =?$ does not imply that the attacker $\mathcal{A}$ does not know φ. As $\mathcal{M}$ and $\mathcal{A}$ observe the system via different interfaces, it might be the case that $\mathcal{A}$ already knows φ and that $\mathcal{M}$ will never infer this information. This corresponds to the non-diagnosability of Ω [7]. This can occur when there exist two arbitrarily long trajectories s and s' corresponding to the same observation μ such that $s \in L_{F_\Omega}(\Omega)$ (thus a non-opaque trajectory of φ) and $s' \notin L_{F_\Omega}(\Omega)$. In the next section, we will give necessary and sufficient conditions under which this case does not occur.

B. Necessary and sufficient conditions for detection/prediction of information flow

Consider the system G as well as the property Ω described in the previous section.

1) Diagnosability & predictability: Intuitively, G is Ω-diagnosable ([10], [7]) if there exists $n \in \mathbb{N}$ such that for any trajectory s of G such that $s \models \Omega$, Ω becomes non-opaque after waiting for at most n observations. This can be formalized as follows

Definition 6: Given a system G, a stable property Ω and an interface $\Pi_{\mathcal{M}}$, G is Ω-diagnosable if,

$$\begin{aligned}\exists n \in \mathbb{N}, \forall s \in L(G) \cap L_{F_\Omega}(\Omega) \cap \Sigma^*.\Sigma^{-1}_{\mathcal{M}},\\ \forall t' \in L(G), t' = s \cdot t \ \wedge \ |\Pi_{\mathcal{M}}(t)| \geq n\\ \Rightarrow [\![\Pi_{\mathcal{M}}(s \cdot t)]\!]_{\mathcal{M}} \subseteq L_{F_\Omega}(\Omega)\end{aligned}$$

The Ω-diagnosability property means that if a trajectory s of the system satisfies Ω, then whatever the extension t of s, t having at least n observable events w.r.t. $\Pi_{\mathcal{M}}$, all trajectories compatible with the observation $\Pi_{\mathcal{M}}(s.t)$ satisfy Ω.

In the case of monitoring opacity, this means that when the monitor is observing a trace in $L_{F_\Omega}(\Omega)$, a "Yes" answer should be produced by the observer after finitely many observed events. Indeed, if there exists $s \in L(G)$ triggered by the system such that φ is non-opaque for $\mathcal{A}$, it means that $\Pi(s) \in L_F(\chi^{\varphi}_{\mathcal{A}}(G))$ and consequently $s \in L_{F_\Omega}(\Omega)$. then according to the definition of diagnosability, $\mathcal{M}$ will surely know it at most n observed events after the observation of $\Pi_{\mathcal{M}}(s)$.

Based on these remarks, we can state that

Proposition 1: With the preceding notations, an information flow from G to $\mathcal{A}$ is surely detected by $\mathcal{M}$ if and only if G is Ω-diagnosable w.r.t. $\Pi_{\mathcal{M}}$, where Ω is computed as in (1).

If the system is Ω-diagnosable, then it might be interesting to refine the verdict by predicting the satisfaction of the property strictly before its actual occurrence [11]. Roughly speaking, Ω is predictable if it is always possible to detect the future satisfaction of Ω, strictly before this happens, only based on the observations.

Definition 7: Given a system G, a property Ω and an interface $\Pi_{\mathcal{M}}$, G is Ω-predictable if

$$\begin{aligned}\exists n \in \mathbb{N}, \forall s \in \mathcal{L}(G) \cap L_{F_\Omega}(\Omega) \cap \Sigma^*.\Sigma^{-1}_{\mathcal{M}},\\ \exists t \in (\mathcal{L}(G) \cap \Sigma^*.\Sigma^{-1}_{\mathcal{M}}) \cup \{\epsilon\}, \ t < s \ \wedge \ t \notin L_{F_\Omega}(\Omega) \text{ s.t.}\\ \forall u \in [\![\Pi_{\mathcal{M}}(t)]\!]_{\mathcal{M}}, \forall v \in \mathcal{L}(G)/u, |\Pi_{\mathcal{M}}(v)| \geq n\\ \Rightarrow u.v \in L_{F_\Omega}(\Omega)\end{aligned}$$

This property means that for any trajectory s that satisfies Ω, there exists a strict prefix t that does not satisfy Ω, such

that any trajectory u compatible with observation $\Pi_{\mathcal{M}}(t)$ will inevitably be extended into a trajectory $u.v$ satisfying Ω. Note that predictability implies diagnosability [11].

Remark 2: There is an algorithm of polynomial complexity for verifying that a system G is Ω-diagnosable or Ω-predictable. More details can be found in [7], [11].
In our setting, this means that $\mathcal{M}$ can always predict that $\mathcal{A}$ will know φ and then the system operator can be warned in time to halt the system or can take counter-measures in order to avoid the secret to be revealed. In other words, if $\mathcal{M}$ observes a trace $\mu \in \mathcal{T}_{\mathcal{M}}(G)$ such that $\mu = \Pi_{\mathcal{M}}(t)$, then $\mathcal{M}$ knows that the secret is not revealed to $\mathcal{A}$, but will be after at most n observations.

Proposition 2: With the preceding notations, an information flow from G to $\mathcal{A}$ is surely detected by $\mathcal{M}$ strictly before its occurrence if and only if G is Ω-predictable w.r.t. $\Pi_{\mathcal{M}}$, where Ω is computed as in (1).

Example 4: To illustrate this section, we still consider the system G and the secret φ defined in Example 2. The property Ω and the set of non-opaque trajectories (i.e. the ones that reveal the secret φ) are given by the LTS described in Fig. 3.

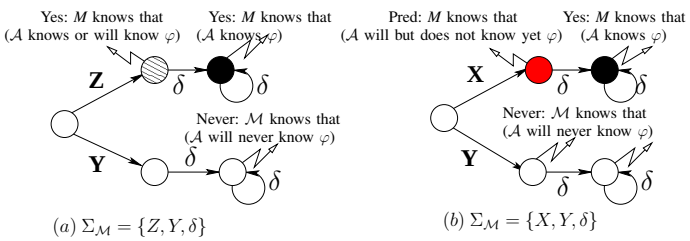


Fig. 5. Observation function $\mathcal{O}^{\Omega}_{\mathcal{M}}$ w.r.t. two different interfaces

Assume that the interface of the monitor $\mathcal{M}$ is reduced to $\Sigma_{\mathcal{M}} = \{Z, Y, \delta\}$. Then, one can show that G is Ω-diagnosable, but not Ω-predictable. The corresponding $\mathcal{O}^{\Omega}_{\mathcal{M}}$ is represented in Figure 5(a). A contrario, if the interface of the monitor $\mathcal{M}$ is $\Sigma_{\mathcal{M}} = \{X, Y, \delta\}$, then the system is Ω-predictable. Indeed, after the observation of X, $\mathcal{M}$ knows that all the possible extensions will satisfy Ω and thus that the secret will be revealed (C.f. Figure 5(b)).

VI. Conclusion

In this paper, given a system modeled by a labeled transition system and a secret property modeled by a regular language, we have shown how to characterize cases of confidential information flow. Then we exposed how an administrator can construct a monitor raising an alarm whenever an attack is detected. Further, we provide necessary and sufficient conditions for such information flows to be always detected by the administrator; in a bounded delay in the case of diagnosability or before they occur in the case of predictability.

Future Work: We first plan to extend these results to more expressive models mixing control and data. Moreover, in this paper, we focused on the detection of information flow. The monitors are passive, they raise an alarm whenever an attack occurred. We plan to go further by investigating the on-line control of the system in order to avoid the secret to be revealed (for example by synthesizing a dynamical observation mask that would minimise the unobservable events). This would, in some points extend the work done by Schneider on security automata [12], and subsequently extended to edit automata [13]. Also, the attacker is interested in deducing a given secret and the administrator is concerned with what does the attacker knows. Epistemic logic seems to be a good candidate to generalize this approach to more than two participants. Finally, following the first results in [9], [4], we plan to investigate the detection of information flow in case an attacker can infer information knowing only an abstraction of the system. Moreover, when dealing with abstraction, the initial knowledge plays an important role in our approach. An interesting extension would be to consider an attacker having an arbitrary initial knowledge of the system and using learning techniques. This attacker will try to acquire a more precise model of the system to likely infer some confidential information.

References

[1] B. Blanchet, A. M., and C. Fournet, "Automated Verification of Selected Equivalences for Security Protocols," in *20th IEEE Symposium on Logic in Computer Science (LICS 2005)*, 2005, pp. 331–340.

[2] G. Lowe, "Towards a completeness result for model checking of security protocols," *Journal of Computer Security*, vol. 7, no. 2-3, 1999.

[3] V. Darmaillacq, J.-C. Fernandez, R. Groz, L. Mounier, and J.-L. Richier, "Test generation for network security rules," in *TestCom 2006*, ser. LNCS, vol. 3964, 2006.

[4] J. Bryans, M. Koutny, L. Mazaré, and P. Y. A. Ryan, "Opacity generalised to transition systems," *International Journal of Information Security*, 2008.

[5] R. Alur, P. Černý, and S. Zdancewic, "Preserving secrecy under refinement," in *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*. Springer, 2006, pp. 107–118.

[6] J. Bryans, M. Koutny, L. Mazaré, and P. Y. A. Ryan, "Opacity generalised to transition systems," in *Revised Selected Papers of the 3rd International Workshop on Formal Aspects in Security and Trust (FAST'05)*, ser. LNCS, vol. 3866, 2006, pp. 81–95.

[7] T. Jéron, H. Marchand, S. Pinchinat, and M.-O. Cordier, "Supervision patterns in discrete event systems diagnosis," in *Workshop on Discrete Event Systems, WODES'06*, Ann-Arbor (MI, USA), July 2006.

[8] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau, "Concurrent secrets," in *8th Workshop on Discrete Event Systems, WODES'06*, S. Lafortune, F. Lin, and D. Tilbury, Eds., Ann Arbor, Michigan, USA, July 2006.

[9] J. Dubreil, T. Jéron, and H. Marchand, "Monitoring information flow by diagnosis techniques," IRISA, Tech. Rep. 1901, August 2008.

[10] M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis., "Diagnosability of discrete event systems," *IEEE Transactions on Automatic Control*, 1995.

[11] T. Jéron, H. Marchand, S. Genc, and S. Lafortune, "Predictability of sequence patterns in discrete event systems," in *IFAC World Congress*, Seoul, Korea, July 2008.

[12] F. B. Schneider, "Enforceable security policies," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 1, pp. 30–50, 2000.

[13] J. Ligatti, L. Bauer, and D. Walker, "Edit automata: Enforcement mechanisms for run-time security policies," *International Journal of Information Security*, vol. 4, no. 1–2, pp. 2–16, Feb. 2005.