Construction de moniteurs pour la surveillance de propriétés de sécurité[1]

Jérémy Dubreil, Thierry Jéron, Hervé Marchand

Projet VerTeCs
INRIA Rennes, Bretagne Atlantique
campus de Beaulieu, 35 042 Rennes Cedex, France
Tel : 02 99 84 74 64
prenom.nom@irisa.fr

RÉSUMÉ. *Nous nous intéressons à la construction de moniteurs permettant de détecter la fuite d'information confidentielle pour des systèmes partiellement observables, modélisés par des systèmes de transition finis. Nous considérons le cas où le secret peut se modéliser par des langages réguliers. Nous commençons par définir la notion d'opacité pour formaliser la fuite d'information. Nous caractérisons l'ensemble des observations pour lesquelles un attaquant infère de l'information confidentielle. En adaptant les techniques de diagnostic sur des systèmes à événement discrets, nous explicitons des conditions nécessaires et suffisantes sur le système pour permettre la détection et/ou la prédiction de cette fuite d'information et construisons un moniteur permettant un administrateur d'assurer cette détection. Nous considérons le cas général où l'attaquant et l'administrateur ont des vues partielles différentes du système.*

ABSTRACT. *In this paper, we are interested in constructing monitors for the detection of confidential information flow in the context of partially observed discrete event systems modelled by finite labelled transitions systems. We focus here on the case where the secret information is given as regular languages. We first characterise the set of observations allowing an attacker to infer secret information. Further, based on the diagnosis of discrete event systems theory, 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. We consider the general case where the attacker and the administrator have different partial views of the system.*

MOTS-CLÉS : *Système à événement discret, Confidentialité, Opacité, Diagnostic, Moniteur*

KEYWORDS: *Discrete Event Systems, Opacity, Information Flow, Diagnosis, Monitoring*

1. Ce travail a été partiellement financé par l'ACI Potestat et le RNRT Politess.

La vérification [BLA 05] et le test de propriété de sécurité [DAR 06] connaissent un développement important depuis quelques années. L'engouement pour de telles recherches suit naturellement l'explosion du nombre de services proposés tant sur les systèmes mobiles et embarqués que sur le réseau Internet. En effet, ce dernier est par nature ouvert à tous et est donc très vulnérable aux attaques de pirates informatiques. Néanmoins, le réseau est devenu le média privilégié pour le déploiement de services qui vont du simple wiki à des services très critiques tels que l'échange d'informations médicales personnelles, des systèmes de vote ou encore des systèmes bancaires. Pour de tels services, la corruption de données ou la fuite d'information peut s'avérer catastrophique. La notion de validation et de certification du niveau de sécurité des applications critiques est donc cruciale pour ce type d'applications.

Les aspects de sécurité sont souvent classés en trois composantes : les aspects de disponibilité, d'intégrité et de confidentialité. Prenons l'exemple d'un système de vote sur Internet. Le fait qu'un attaquant puisse bloquer le déroulement du vote est un problème de disponibilité. Un problème d'intégrité consiste à pouvoir modifier le vote d'un autre utilisateur et pouvoir mettre en relation des votants et leurs choix lors du vote est un problème de confidentialité. On peut également noter que ces trois notions liées à la sécurité sont étroitement corrélées[1].

Dans le cadre de ce papier, nous nous focalisons sur les aspects de confidentialité et plus particulièrement sur la notion d'opacité telle qu'elle est définie par [BRY 06] et [ALU 06]. Nous allons voir que les aspects de confidentialité se distinguent des autres en faisant intervenir plus clairement les hypothèses qui sont faites sur le potentiel de l'attaquant à connaître et à analyser le système. En effet, toute analyse de la capacité d'une application à garder secrètes certaines informations est dépendante de l'observation qui est faite par un utilisateur, potentiellement un attaquant, sur le système. La motivation de ces travaux est d'analyser et d'essayer de corriger un service qui présente des failles de sécurités de ce type. Nous présenterons donc ici une première étape qui consiste à rechercher les cas de fuites d'information liées à des propriétés de confidentialité. En supposant ces fuites d'information possibles, nous rechercherons alors un moyen de les observer, voire de les prédire.

Présentation du problème. Dans cette étude, nous disposons de trois acteurs : un système S, un attaquant $\mathcal{A}$ et un moniteur M (modélisant par exemple un administrateur de ce système) (C.f. Figure 1). On suppose ici que le système S est modélisé par un automate fini sur un alphabet Σ. S fournit un service aux utilisateurs (i.e. aux attaquants potentiels) au travers d'une interface $\Sigma_a \subseteq \Sigma$. Intuitivement $\Sigma_a \subseteq \Sigma$ correspond à l'ensemble des entrées et des sorties possibles pour un utilisateur du service. Ce service S requiert des conditions de confidentialité définies par une politique de sécurité. Suivant l'approche donnée par [JéR 06] pour le diagnostic et par [ALU 06] et [BRY 06], nous modélisons le secret par une propriété φ définie par un langage régulier sur Σ. Nous supposons qu'un attaquant connaît entièrement le modèle S mais

1. Dans un autre domaine plus axé système, une mauvaise programmation pour la gestion des mots de passe administrateur, peut les rendre accessibles à un attaquant et cela peut avoir des conséquences sur la disponibilité ou sur l'intégrité.

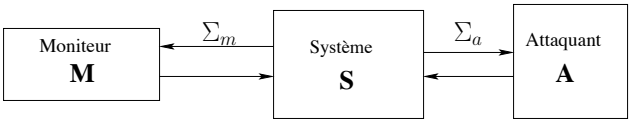


Figure 1 – Architecture

n'observe que Σ_a[2]. Le secret est conservé tant que l'attaquant ne sait pas dire que la propriété φ est vérifiée par l'exécution courante de S en se basant uniquement sur l'observation de Σ_a.

De son côté, le moniteur $\mathcal{M}$ cherche à analyser le flot d'information entre S et $\mathcal{A}$ de manière à signaler la corruption du secret. $\mathcal{M}$ peut aussi chercher à prédire que le secret sera inévitablement dévoilé ou à informer qu'il ne le sera jamais. On suppose pour cela que $\mathcal{M}$ connaît les capacités de $\mathcal{A}$, c'est à dire qu'il connaît S et Σ_a. Il observe l'alphabet $\Sigma_m \subseteq \Sigma$. On ne suppose qu'il n'y a pas de lien particulier entre Σ_a et Σ_m qui sont deux sous alphabets quelconques de Σ.

Dans ce papier, nous montrons comment les notions et techniques de diagnostic de systèmes à événements discrets permettent de caractériser la fuite d'information vers l'attaquant, et la détection par un administrateur d'éventuelles attaques.

Organisation du papier. Après un rappel des modèles et des notations en Section 1, nous définissons, en Section 2, la notion d'inférence de propriétés sous observation partielle et donnons la construction de l'observateur permettant cette inférence. En Section 3, nous définissons la notion d'opacité permettant de formaliser la fuite d'information. Plus précisément, nous caractérisons l'ensemble des observations pour lesquelles un attaquant infère de l'information confidentielle. En Section 4, en adaptant les techniques de diagnostic sur des systèmes à événement discrets, nous explicitons des conditions nécessaires et suffisantes sur le système pour permettre la détection et/ou la prédiction de cette fuite d'information et construisons un moniteur pour un administrateur assurant cette détection.

1. Modèles & Notations

Dans cette section, nous fixons les notations et définissons les opérations utilisées par la suite. Nous considérons ici que les systèmes sont modélisés par des systèmes de transitions étiquetés finis (LTS) :

Définition 1 *Un LTS est un quadruplet $G = (Q_G, \Sigma, \rightarrow_G, q_G^0)$ où Σ est un alphabet fini, Q_G est un ensemble fini d'états, $q_G^0 \subseteq Q_G$ est l'état initial, $\rightarrow \subseteq Q_G \times \Sigma \times Q_G$ est la relation de transition.*

2. Cette connaissance du système peut par exemple avoir été acquise par reverse-ingenieering, ou tout simplement par la connaissance précise des protocoles sous-jacents à ce système.

Notations. Soit G un LTS, nous utiliserons les notations suivantes :

– $q \xrightarrow{a}_G q'$ lorsque $(q, a, q') \in \rightarrow_G$ et $q \xrightarrow{a}_G$ pour $\exists q' \in Q_G$, $q \xrightarrow{a}_G q'$. Nous étendons ces notations par $q \xrightarrow{l}_G q'$ pour $l \in \Sigma^*$ et par $q \xrightarrow{l}_G$ pour $\exists q' \in Q_G$, $q \xrightarrow{l}_G q'$.

– $\Sigma(q) \stackrel{\Delta}{=} \{a \in \Sigma \mid q \xrightarrow{a}_{\mathrm{G}}\}$ correspond à l'ensemble des événements admissibles dans un état q de G. G est *complet* si $\forall q \in Q_{\mathrm{G}}, \Sigma(q) = \Sigma$.

– On pose également $\Delta_{\mathrm{G}}(q, l) \stackrel{\Delta}{=} \{q' \in Q_{\mathrm{G}} \mid q \xrightarrow{l}_{\mathrm{G}} q'\}$ l'ensemble des états que le système peut avoir atteint en tirant la séquence l à partir de q. Cette notation s'étend à un ensemble de langages et d'états comme suit : pour $L \subseteq \Sigma^*$, $\Delta_{\mathrm{G}}(q, L) \stackrel{\Delta}{=} \{q' \in Q_{\mathrm{G}} \mid q \xrightarrow{s}_{\mathrm{G}} q'$ pour $s \in L\}$, et pour $Q' \subseteq Q_{\mathrm{G}}$, $\Delta_{\mathrm{G}}(Q', L) = \bigcup_{q \in Q'} \Delta_{\mathrm{G}}(q, L)$.

– On notera $\mathcal{L}(G) = \{l \in \Sigma^*, q_o \xrightarrow{l}_G\}$ l'ensemble des trajectoires d'un système. Rapidement dans ce papier, nous aurons besoin de distinguer un sous ensemble d'états accepteurs $F_G \subseteq Q_G$. On notera alors $\mathcal{L}_{F_G}(G) = \{l \in \Sigma^* \mid \exists q \in F_G, q_o \xrightarrow{l}_G q\}$ l'ensemble des trajectoires qui se terminent dans un état de F_G.

Définition 2 *(Produit synchrone) Soit $G^i = (Q^i, \Sigma, \rightarrow_{G^i}, q^0_{G^i})$, $i = 1, 2$, deux LTS. Le* produit synchrone *entre G^1 et G^2 est un LTS $G^1 \times G^2 = (Q^1 \times Q^2, \Sigma, \rightarrow_{G^1 \times G^2}, (q^0_{G^1}, q^0_{G^2}))$, où $(q^1, q^2) \xrightarrow{\sigma}_{G^1 \times G^2} (q'^1, q'^2)$ lorsque $q^1 \xrightarrow{\sigma}_{G^1} q'^1$ et $q^2 \xrightarrow{\sigma}_{G^2} q'^2$.*

Clairement, $\mathcal{L}(G^1 \times G^2) = \mathcal{L}(G^1) \cap \mathcal{L}(G^2)$ et pour $F_i \subseteq Q^i, i = 1, 2$, nous avons également $\mathcal{L}_{F_1 \times F_2}(G^1 \times G^2) = \mathcal{L}_{F_1}(G^1) \cap \mathcal{L}_{F_2}(G^2)$.

Opérateurs ensemblistes. Étant donné un LTS G, pour $E \subseteq Q_{\mathrm{G}}$, les opérateurs $pre^{\forall}_{\mathrm{G}}$ et $pre^{\exists}_{\mathrm{G}}$ sont définis de la manière suivante :

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

En d'autres termes, les états de $Pre^{\exists}_{\mathrm{G}}(E)$ sont ceux ayant au moins un successeur immédiat par Σ menant à E et $Pre^{\forall}_{\mathrm{G}}(E)$ est l'ensemble de ceux dont tous les successeurs immédiats par un événement de Σ appartiennent à E. L'ensemble $Inev_{\mathrm{G}}(E)$ des états qui mènent inévitablement dans E et l'ensemble $CoReach_{\mathrm{G}}(E)$ à partir desquels E est atteignable sont donnés par les points fixes :

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

Comportement Observable. Le point central de notre approche est la capacité d'un utilisateur $\mathcal{U}$ à déduire de l'information sur le système en n'observant qu'une partie des événements $\Sigma_u \subseteq \Sigma$.

Par la suite, des éléments de Σ_u^* seront notés μ, μ'. G est dit Σ_u-*vivant* si $\forall q \in Q, \exists s \in \Sigma^*.\Sigma_u, q \xrightarrow{s}$. Pour le sous alphabet $\Sigma_u \subseteq \Sigma$, on définit la projection naturelle $\pi_{\Sigma_u} : \Sigma^* \rightarrow \Sigma_u$ définie en oubliant les événements qui ne sont pas dans Σ_u. Étant donné un langage $L \subseteq \Sigma^*$, cette notation peut être étendue de la manière suivante : $\pi_{\Sigma_u}(L) = \{\pi_{\Sigma_u}(l) \mid l \in L\}$. Partant d'un système G et d'un ensemble d'observables Σ_u, l'ensemble des *traces observables* de G est donc simplement donné par $\mathcal{T}_{\Sigma_u}(G) = \pi_{\Sigma_u}(\mathcal{L}(G))$. La projection inverse d'un ensemble de traces $T \subseteq \Sigma_u^*$ est donnée par $\pi_{\Sigma_u}^{-1}(T) = \{l \in \Sigma^* \mid \pi_{\Sigma_u}(l) \in T\}$. Étant donnée une trace μ de G, nous définissons $[\![\mu]\!]_{\Sigma_u}$ comme l'ensemble des trajectoires possibles de G compatibles avec l'observation de la trace μ.

$$[\![\mu]\!]_{\Sigma_u} \quad \triangleq \quad \begin{array}{l} \pi_{\Sigma_u}^{-1}(\mu) \cap \mathcal{L}(G) \cap \Sigma^*\Sigma_u \text{ si } \mu \neq \epsilon \\ \epsilon \text{ sinon.} \end{array}$$

Nous introduisons maintenant la Σ_u-clôture d'un LTS G qui consiste à s'abstraire des événements inobservables tout en respectant la sémantique de l'observation $[\![.]\!]_{\Sigma_u}$.

Définition 3 *Pour un LTS $G = (Q_G, \Sigma, \rightarrow, q_G^0)$, la Σ_u-*clôture de G, *notée* $\textsc{Obs}_{\Sigma_u}(G)$ *est un LTS $(Q_G, \Sigma_u, \rightarrow_o, q_G^0)$ où pour chaque $q, q' \in Q_G$, $a \in \Sigma_u$, $q \xrightarrow{a}_o q'$ si il existe $s \in (\Sigma \setminus \Sigma_u)^*$ t.q. $q \xrightarrow{s.a}_G q'$.*

En se basant sur la définition précédente, on obtient

$$\mathcal{L}(\textsc{Obs}_{\Sigma_u}(G)) = \mathcal{T}(G) \text{ et } \mathcal{L}_{F_G}(\textsc{Obs}_{\Sigma_u}(G)) = \pi_{\Sigma_u}(\mathcal{L}_{F_G}(G)).$$

On dit qu'un LTS G est *déterministe* sur Σ_u, si il n'a aucune action inobservable et si, lorsque $q \xrightarrow{a}_G q'$ et $q \xrightarrow{a}_G q''$, alors $q' = q''$, $\forall q \in Q_G$ et $\forall a \in \Sigma_u$.

Dans la construction de moniteurs en charge de l'observation du système, nous aurons besoin de construire à partir d'un LTS non déterministe un LTS déterministe et de mêmes traces.

Définition 4 *Soit $G = (Q_G, \Sigma, \rightarrow_G, q_G^0)$ un LTS avec $\Sigma_u \subseteq \Sigma$. Le* déterminisé de G *relativement à Σ_u est un LTS $Det_{\Sigma_u}(G) = (\mathcal{X}, \Sigma_u, \rightarrow_d, X^0)$ où $\mathcal{X} = 2^{Q_G}$ (l'ensemble des parties de Q_G appelées* macro états*), $X^0 = \{q_G^0\}$ et $\rightarrow_d = \{(X, a, \Delta_G(X, (\Sigma \setminus \Sigma_u)^*.a) \mid X \in \mathcal{X}$ et $a \in \Sigma_u\}$.*

Pour cette définition, le macro état X' cible d'une transition $X \xrightarrow{a}_d X'$ est composé des états q' de G qui sont cibles de séquences de transitions $q \xrightarrow{s.a} q'$ se terminant par un événement observable a, avec $q \in X$. On peut déduire de la définition de $\rightarrow_d$ que $\Delta_{Det_{\Sigma_u}(G)}(X^0, \mu) = \{\Delta_G(q_G^0, [\![\mu]\!]_{\Sigma_u})\}$. Cela signifie qu'un macro-état atteint à partir de X^0 par μ dans $Det_{\Sigma_u}(G)$ est composé de l'ensemble des états qui sont atteints à partir de q_G^0 par une trajectoire de $[\![\mu]\!]$ dans G. On note aussi que la procédure de déterminisation préserve les observations, c'est à dire que l'on a $\mathcal{L}(Det_{\Sigma_u}(G)) = \mathcal{T}_{\Sigma_u}(G)$.

2. Inférence de propriétés sous observation partielle

Ces opérations sur les LTS nous permettent maintenant de présenter l'aspect central de notre approche. Soit $\mathcal{U}$ un utilisateur qui interagit avec un service modélisé par un LTS $S = (Q^S, q_0^S, \Sigma, \rightarrow)$ au travers d'une interface $\Sigma_u \subseteq \Sigma$. Dans cet article, nous considérons des propriétés modélisées par des langages réguliers sur Σ.

Définition 5 *Une propriété est donnée par le langage marqué $\mathcal{L}_{F_\psi}(\psi) \subseteq \Sigma^*$ d'un LTS complet déterministe $\psi = (Q^\psi, q_0^\psi, \Sigma, \rightarrow_\psi)$ muni des états finals F_ψ.* •

On assimile la propriété et le LTS qui modélise cette propriété en notant $s \models \psi$ pour $s \in \mathcal{L}_{F_\psi}(\psi)$. ψ étant complet (i.e. $\mathcal{L}(\psi) = \Sigma^*$), on a donc $\mathcal{L}(S \times \psi) = \mathcal{L}(S)$ et $\mathcal{L}_{Q^S \times F_\psi}(S \times \psi) = \mathcal{L}(S) \cap \mathcal{L}_{F_\psi}(\psi)$ est l'ensemble des trajectoires de S qui satisfont ψ.

Soit $s \in \mathcal{L}(S)$ l'exécution courante du système. L'utilisateur $\mathcal{U}$ cherche à savoir si s satisfait la propriété ψ en observant $\mu = \pi_{\Sigma_u}(s) \in \mathcal{T}(S)$. Seulement, il lui est impossible de distinguer s d'une autre trajectoire s' telle que $\pi_{\Sigma_u}(s) = \pi_{\Sigma_u}(s') = \mu$. $\mathcal{U}$ peut donc seulement inférer de l'information partielle sur $s \models \psi$ à partir de $[\![\mu]\!]_{\Sigma_u}$, l'ensemble des trajectoires compatibles de S correspondant à l'observation de μ. Par exemple, $\mathcal{U}$ est sûr que $s \models \psi$ si $[\![\mu]\!]_{\Sigma_u} \subseteq \mathcal{L}_{F_\psi}(\psi)$. Mais s'il existe $s' \in \mathcal{L}(S)$ telle que $\pi_{\Sigma_u}(s') = \mu$ et $s' \not\models \psi$, alors $\mathcal{U}$ ne pourra en déduire aucune information quant à la satisfaction de ψ par l'exécution s. Pour aller plus loin, $\mathcal{U}$ peut aussi chercher à savoir qu'après avoir observé μ, ψ ne sera jamais satisfaite par les exécutions prolongeant s.

Nous formalisons maintenant ces idées et proposons une construction de la fonction $\mathcal{O}_\mathcal{U}^\psi$ inspirée de [JéR 06], qui donne, pour chaque observation $\mu \in \mathcal{T}(S)$ ce que peut déduire un utilisateur $\mathcal{U}$ à propos de s et ψ. Formellement, si s est l'exécution courante du système et $\mu = \pi_{\Sigma_u}(s)$ l'observation qui en est faite, un choix possible pour une fonction d'observation est donnée par

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

où la sémantique des verdicts est donnée par :

1) $\mathcal{O}_\mathcal{U}^\psi(\mu) = Yes$ si $\mathcal{U}$ sait que pour l'exécution courante s (t.q. $\pi_{\Sigma_u}(s) = \mu$), $s \models \psi$;

2) $\mathcal{O}_\mathcal{U}(\mu) = Inev$ si $\mathcal{U}$ sait que $s \not\models \psi$ mais que ψ sera inévitablement satisfaite par tous les prolongements de s ;

3) $\mathcal{O}_\mathcal{U}^\psi(\mu) = Inev\_Yes$ si $\mathcal{U}$ sait que $s \models \psi$ ou que ψ sera inévitablement satisfaite dans le futur mais ne peut pas distinguer les deux cas de figure ;

4) $\mathcal{O}_\mathcal{U}^\psi(\mu) = Never$ si $\mathcal{U}$ sait que ψ ne sera jamais satisfaite par les exécutions de S qui prolongent s ;

5) $\mathcal{O}_\mathcal{U}^\psi(\mu) = No$ si $\mathcal{U}$ sait que $s \not\models \psi$, que ψ n'est ni inévitable ni impossible ;

6) $\mathcal{O}_\mathcal{U}^\psi(\mu) = ?$ dans tous les autres cas, c'est à dire que $\mathcal{U}$ ne peut inférer aucune des informations précédentes liant s et ψ d'après l'observation $\mu = \pi_{\Sigma_u}(s)$

2.1. *Construction de* $\mathcal{O}_{\psi}^{\mathcal{U}}$

Nous proposons maintenant une construction de la fonction $\mathcal{O}_{\mathcal{U}}^{\psi} : \Sigma_u^* \to V$:

Étape 1 : On construit le produit synchrone $S_\psi = S \times \psi = (Q_{S_\psi}, \Sigma, \to_{S_\psi}, q^0_{S_\psi})$ ainsi que l'ensemble des états marqués $F_{S_\psi} = Q^S \times F_\psi$. D'après la propriété du produit synchrone et ψ étant complet, $\mathcal{L}(S_\psi) = \mathcal{L}(S)$ et $\mathcal{L}_{F_{S_\psi}}(S_\psi) = \mathcal{L}(S) \cap \mathcal{L}_{F_\psi}(\psi)$. Ainsi, $\mathcal{L}_{F_{S_\psi}}(S_\psi)$ correspond à l'ensemble des trajectoires de S qui vérifient ψ.

Étape 2 : Cette étape consiste à calculer $Inev_{S_\psi}(F_{S_\psi})$ sur S_ψ et la partition suivante : $Q_{S_\psi} = F_{S_\psi} \cup I_{S_\psi} \cup P_{S_\psi} \cup N_{S_\psi}$, avec

– $I_{S_\psi} = Inev_{S_\psi}(F_{S_\psi}) \setminus F_{S_\psi}$ l'ensemble des états n'appartenant pas à F_{S_ψ} mais à partir desquels F_{S_ψ} est inévitable ;

– $P_{S_\psi} = Q_{S_\psi} \setminus CoReach_{S_\psi}(F_{S_\psi})$, i.e. l'ensemble des états à partir desquels F_{S_ψ} n'est pas accessible ;

– $N_{S_\psi} = Q_{S_\psi} \setminus (F_{S_\psi} \cup I_{S_\psi} \cup P_{S_\psi})$ l'ensemble des autres états.

Étape 3 : On calcule $\chi_u^\psi(S) = Det_{\Sigma_u}(S_\psi) = (\mathcal{X}, \Sigma_u, \to_d, X^0)$. On obtient alors $\mathcal{L}(\chi_u^\psi(S)) = \mathcal{T}_{\Sigma_u}(S)$. Pour chaque observation $\mu \in \mathcal{T}_{\Sigma_u}(S)$, on a $\Delta_{\chi_u^\psi(S)}(X^0, \mu) = \{\Delta_{S_\psi}(q^0_{S_\psi}, [\![\mu]\!]_{\Sigma_u}\}$.

Étape 4 : On calcule la fonction d'observation $\mathcal{O}_{\mathcal{U}}^{\psi}$ à partir de $\chi_u^\psi(S)$ et des ensembles $F_{S_\psi}, I_{S_\psi}, P_{S_\psi}, N_{S_\psi}$ comme suit :

$$\forall \mu \in \mathcal{T}_{\Sigma_u}(S),\ \mathcal{O}_{\mathcal{U}}^{\psi}(\mu) = \begin{cases} Yes, & \text{si} \quad \Delta_{\chi_u^\psi(S)}(X^0, \mu) \subseteq F_{S_\psi} \\ Inev, & \text{si} \quad \Delta_{\chi_u^\psi(S)}(X^0, \mu) \subseteq I_{S_\psi} \\ Inev\_Yes, & \text{si} \quad \Delta_{\chi_u^\psi(S)}(X^0, \mu) \subseteq (I_{S_\psi} \cup F_{S_\psi}) \\ & \quad \wedge\ \Delta_{\chi_u^\psi(S)}(X^0, \mu) \cap I_{S_\psi} \neq \emptyset \\ & \quad \wedge\ \Delta_{\chi_u^\psi(S)}(X^0, \mu) \cap F_{S_\psi} \neq \emptyset \\ No, & \text{si} \quad \Delta_{\chi_u^\psi(S)}(X^0, \mu) \subseteq N_{S_\psi} \\ Never & \text{si} \quad \Delta_{\chi_u^\psi(S)}(X^0, \mu) \subseteq P_{S_\psi} \\ ? & \text{autrement.} \end{cases} \tag{1}$$

On peut facilement vérifier que cette construction de $\mathcal{O}_{\mathcal{U}}^{\psi}$ est bien conforme avec la définition informelle donnée précédemment. Par exemple pour le verdict Yes, considérons une exécution $s \in \mathcal{L}(S)$ d'observation $\mu = \pi_{\Sigma_u}(s)$ et $\mathcal{O}_{\mathcal{U}}^{\psi}(\mu) = Yes$. On a alors $\Delta_{\chi_u^\psi(S)}(X^0, \mu) \subseteq F_{S_\psi}$. D'après la définition de $\chi_u^\psi(S)$, pour tout $s' \in \mathcal{L}(S)$

telle que $\pi_{\Sigma_u}(s') = \mu$, $\Delta_{S_\psi}(q^0_{S_\psi}, s') \subseteq \Delta_{\chi^\psi_u(S)}(X^0, \mu) \subseteq F_{S_\psi}$, donc $s' \models \psi$. Ainsi, pour toute trajectoire $s' \in [\![\mu]\!]$, $s' \models \psi$ et en particulier $s \models \psi$.

En conclusion de cette partie : étant donné un système S qu'un utilisateur $\mathcal{U}$ observe via une interface Σ_u, nous savons construire une fonction $\mathcal{O}^\psi_u : \Sigma^*_u \to V$ qui donne des informations que peut déduire $\mathcal{U}$ liant les exécutions de S et la propriété ψ.

3. Caractérisation et vérification de l'opacité

On suppose maintenant que l'attaquant $\mathcal{A}$ est un utilisateur d'un service S qui cherche à inférer des informations confidentielles sur le système. Nous supposons que l'attaquant connaît entièrement le modèle S mais n'observe que $\Sigma_a \subseteq \Sigma$. On considère un secret φ donné par le langage marqué d'un LTS complet et déterministe, $\varphi = (Q^\varphi, q^0_\varphi, \Sigma, \to, F_\varphi)$. On suppose que $\mathcal{A}$ sait construire une fonction d'observation comme décrite dans la section précédente. On cherche à savoir si l'attaquant peut deviner que φ est vérifiée par l'exécution courante $s \in \mathcal{L}(S)$. Voici un exemple de propriété de confidentialité qui peut se modéliser par un LTS :

Exemple 1 *Considérons un système S pour lequel il existe deux commandes internes (donc inobservables)* LOCK *et* UNLOCK *qui, respectivement, active et désactive la protection en lecture sur un fichier de mot de passe. L'attaquant peut chercher à savoir lorsque le fichier est accessible en lecture sans avoir la possibilité d'observer ni* LOCK *ni* UNLOCK*. Considérons la propriété φ suivante, modélisée par le LTS de la figure 2. Le langage $\mathcal{L}(S) \cap \mathcal{L}_{F_\varphi}(\varphi) \subseteq \mathcal{L}(S)$ représente alors l'ensemble des exécutions possibles de S pour lesquelles le fichier est accessible en lecture.*

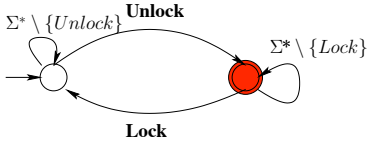


Figure 2 – Exemple de propriété de confidentialité

3.1. *Définition de l'opacité*

Intuitivement, on dira qu'une propriété est opaque pour un système S et Σ_a si l'attaquant $\mathcal{A}$ ne peut jamais deviner de manière certaine d'après ce qu'il observe si φ est vérifiée sur S [ALU 06, BRY 06, BAD 05].

Définition 6 (Opacité) *Étant donné un système S et une propriété φ. On dira que φ est opaque sur S relativement à Σ_a si*

$$\forall s \in \mathcal{L}(S), [\![\pi_{\Sigma_a}(s)]\!]_{\Sigma_a} \not\subseteq \mathcal{L}_{F_\varphi}(\varphi) \tag{2}$$

En se basant sur la sémantique de $\mathcal{O}_{\mathcal{A}}^{\varphi}$ décrite dans la section précédente, on peut donc dire que φ est opaque sur S relativement à Σ_a si

$$\forall s \in \mathcal{L}(S),\ \mathcal{O}_{\mathcal{A}}^{\varphi}(\pi_{\Sigma_a}(s)) \neq Yes$$

Remarque 1 *φ est opaque sur S relativement à Σ_a si et seulement si*

$$\forall \mu \in \mathcal{T}_{\Sigma_a}(S),\ [\![\mu]\!]_{\Sigma_a} \not\subseteq \mathcal{L}_{F_\varphi}(\varphi),$$

et φ est non-opaque sur S relativement à Σ_a si et seulement si

$$\exists \mu \in \mathcal{T}_{\Sigma_a}(S),\ [\![\mu]\!]_{\Sigma_a} \subseteq \mathcal{L}_{F_\varphi}(\varphi)$$

3.2. *Vérification de l'opacité*

On se pose maintenant la question de la vérification de l'opacité, c'est à dire, un attaquant peut-il inférer que la propriété est vérifiée sur le système sur la base de l'observation de l'exécution courante. On retrouve ici un cas particulier de l'inférence de propriété présenté dans la partie précédente. On pose alors $\chi_{\varphi}^{\Sigma_a}(S) = Det_{\Sigma_o}(S \times \varphi) = (\mathcal{X}, \Sigma_a, \rightarrow_d, X^0)$ muni des états marqués $F = 2^{Q^S \times F_\varphi}$. Par construction, on a alors la propriété suivante :

$$[\![\mathcal{L}_F(\chi_{\varphi}^{\Sigma_a}(S))]\!]_{\Sigma_a} = \{s \in L(S) \mid [\![\pi_{\Sigma_a}(s)]\!]_{\Sigma_a} \subseteq \mathcal{L}_{F_\varphi}(\varphi)\}$$

Ceci nous donne une caractérisation de l'opacité :

Proposition 1 *φ est opaque sur S pour l'observation $\Sigma_a \Leftrightarrow \mathcal{L}_F(\chi_{\varphi}^{\Sigma_a}(S)) = \emptyset$.* $\diamond$

Vérifier l'opacité de φ sur S consiste donc à vérifier que l'ensemble d'états F n'est pas accessible dans $\chi_{\varphi}^{\Sigma_a}(S)$. Si cet ensemble est accessible, alors φ n'est pas opaque et il existe une observation qui permet à l'attaquant d'inférer φ. En d'autres termes, $\mathcal{L}_F(\chi_{\varphi}^{\Sigma_a}(S))$ est l'ensemble des observations pour lesquelles l'attaquant $\mathcal{A}$ sait que l'exécution courante du système satisfait φ.

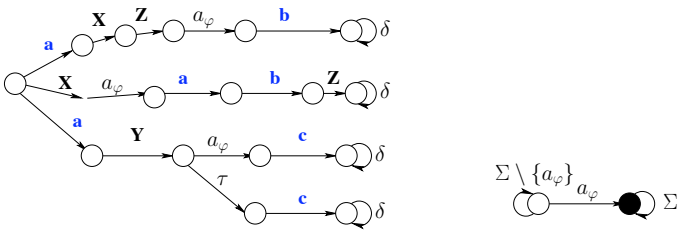
Dans ce cas, l'attaquant $\mathcal{A}$, en se basant sur les techniques de la section précédente, peut calculer le LTS $\chi_{\varphi}^{\Sigma_a}(S)$ et en déduire une fonction d'observation $\mathcal{O}_{\mathcal{A}}^{\Omega}$ telle que pour une observation donnée μ de $\mathcal{T}(S)$,

– si $\mathcal{O}_{\mathcal{A}}^{\varphi}(\mu) = Yes$, alors $\mu \in \mathcal{L}_F(\chi_{\varphi}^{\Sigma_a}(S))$ et donc $[\![\mu]\!]_{\Sigma_a} \subseteq \mathcal{L}_{F_\varphi}(\varphi)$; l'attaquant, en se basant sur cette observation, peut en déduire que φ est vérifiée sur S et il y a donc une fuite d'information ;

– si $\mathcal{O}_{\mathcal{A}}^{\varphi}(\mu) =?_{\mathcal{A}}$, $\mathcal{A}$ ne déduit aucune information sur φ.

A noter que l'on suppose ici que $\mathcal{A}$ est uniquement intéressé par la détection de la satisfaction de la propriété. Ainsi, par rapport à (1), les verdicts No, $Inev$, $Inev\_Yes$, $Never$, et ? ont été confondus en un seul verdict $?_{\mathcal{A}}$.

Exemple 2 *Considérons le système S décrit par la figure 3 (a). L'alphabet de S est donné par $\Sigma = \{a, b, c, X, Y, Z, a_\varphi, \tau, \delta\}$. on suppose que la propriété de confidentialité est donnée par le LTS de la Figure 3 (b) ; l'état marqué est représenté par le cercle noir. Sur cet exemple, on cherche donc à savoir si l'événement a_φ s'est produit dans le système.*



(a) le système S (b) la propriété de confidentialité φ

Figure 3 – S et φ

On suppose que l'interface de l'attaquant se réduit à $\Sigma_a = \{a, b, c, \delta\}$. L'observateur $\mathcal{O}^\varphi_\mathcal{A}$ que $\mathcal{A}$ peut alors construire est donné en Figure 4.

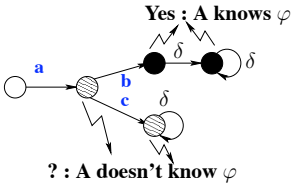


Figure 4 – $\mathcal{O}^\varphi_\mathcal{A}$ issu de $\chi^{\Sigma_a}_\varphi$

Si $\mathcal{A}$ observe $ab\delta^*$ alors il est sûr que φ est satisfaite et donc que l'événement φ s'est produit dans S (l'ensemble des séquences possibles est $a.X.Z.a_\varphi.b\delta^*$ ou $X.a_\varphi.a.b.Z.\delta^*$). Par contre, si il observe simplement a ou encore $ac\delta^*$ alors il ne sait pas si φ est satisfaite ou non. Certaines des séquences compatibles avec l'observation satisfont la propriété, d'autres non. $\mathcal{A}$ ne peut donc pas en déduire d'information.

Remarque 2 *Il est également possible de considérer d'autres types d'opacité :*

– Dans certain cas, $\mathcal{A}$ peut être intéressé par l'information : "φ est satisfaite par l'exécution courante de S ou sera inévitablement satisfaite dans le futur". Dans ce cas, on dira qu'une propriété est opaque si $\forall \mu \in \mathcal{T}_{\Sigma_a}(S), [\![\mu]\!]_{\Sigma_a} \not\subseteq \mathcal{L}_{Inev_{S_\varphi}(F_{S_\varphi})}(S_\varphi)$. La vérification de l'opacité et la construction de l'observateur sont similaires.

– Il est également possible de considérer le cas où il y a une fuite d'information lorsque l'attaquant sait que, soit φ soit $\neg\varphi$ est vérifiée (c.f. [ALU 06]). En d'autres

termes, être opaque (pour cette définition) signifie que φ est opaque et $\neg\varphi$ est opaque (pour la définition 6).

L'observateur qui en découle aura alors 3 verdicts $\{Yes,\ No_{\mathcal{A}},\ ?_{\mathcal{A}}\}$, où le verdict Yes correspond au verdict décrit en (1), le verdict $No_{\mathcal{A}}$ regroupe les verdicts $No, Inev, Never$ tandis $?_{\mathcal{A}}$ regroupe tous les autres cas. ◇

4. Surveillance de la non-opacité

Étant donné φ, un secret, en se basant sur les techniques de la section précédente, il est possible de vérifier si φ est opaque sur S. Si le résultat est négatif, il est important pour un administrateur de surveiller le système. Cette surveillance peut se faire en ligne par l'intermédiaire d'un moniteur $\mathcal{M}$ de manière à envoyer une alarme quand il y a une fuite d'information.

On suppose pour cela que $\mathcal{M}$ connaît le système S et observe l'alphabet $\Sigma_m \subseteq \Sigma$. Il connaît aussi les capacités de $\mathcal{A}$, c'est à dire qu'il connaît Σ_a et qu'il sait construire $\mathcal{O}^{\varphi}_{\mathcal{A}}$. On ne suppose aucune relation entre Σ_a et Σ_m qui sont deux sous-alphabets quelconques de Σ. $\mathcal{M}$ doit donc inférer les connaissances que peut avoir $\mathcal{A}$ en fonction des observations de $\mathcal{T}_{\Sigma_m}(S) \subseteq \Sigma_m^*$.

Si φ est non-opaque sur S, on peut construire une fonction d'observation pour diagnostiquer que le secret a été corrompu. On peut toutefois essayer d'être plus précis et chercher à prédire la fuite d'information que le secret sera inévitablement dévoilé à l'attaquant ou à informer qu'il ne le sera jamais.

En fait, il n'est pas nécessaire de diagnostiquer que le système a exécuté une séquence qui satisfait φ, si celle-ci correspond à une exécution opaque (cette exécution ne donnant aucune information à l'attaquant) ; seules les exécutions engendrant une fuite d'information sont à prendre en compte. En d'autres termes, il ne s'agit pas pour $\mathcal{M}$ de diagnostiquer/prédire l'occurrence de φ, mais plutôt de diagnostiquer/prédire la fuite d'information caractérisé par Ω. En ffet, le secret φ est corrompu lors d'une exécution $s \in \mathcal{L}(S)$ si $\pi_{\Sigma_a}(s) \in \mathcal{L}_F(\chi_{\varphi}^{\Sigma_o}(S))$. On s'intéresse à diagnostiquer la propriété sur les trajectoires : "le secret φ a été corrompu", ce qui correspond au langage : $\pi^{-1}_{\Sigma_a}(\mathcal{L}_F(\chi_{\varphi}^{\Sigma_o}(S))) \cdot \Sigma^*$. Ce langage est modélisée par un LTS marqué Ω, tel que :

$$\mathcal{L}_{F_\Omega}(\Omega) = \pi^{-1}_{\Sigma_a}(\mathcal{L}_F(\chi_{\varphi}^{\Sigma_o}(S))) \cdot \Sigma^* \tag{3}$$

Remarque 3

Exemple 3 *A titre indicatif, le LTS Ω de l'exemple 2 est donné en Figure 3*

Ω ainsi construite est stable, c'est à dire $\mathcal{L}_{F_\Omega}(\Omega) = \mathcal{L}_{F_\Omega}(\Omega) \cdot \Sigma^*$. Si $s \models \Omega$ alors pour tout prolongement s' de s, on a $s' \models \Omega$. En d'autres termes, Ω est la négation d'une propriété de sûreté.

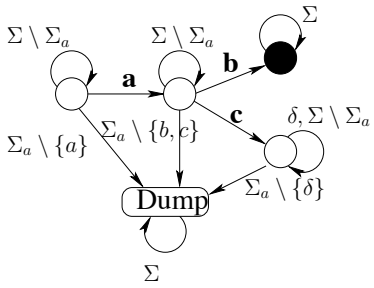


Figure 5 – Le LTS Ω issu de $\chi_{\varphi}^{\Sigma_a}(S))$

4.1. *Surveillance de fuite d'information*

Étant donnés un système S, un attaquant $\mathcal{A}$ (observant S via Σ_a) et un secret φ supposée non-opaque, nous décrivons maintenant la démarche permettant à un administrateur $\mathcal{M}$ (observant S via Σ_m), de savoir s'il y a eu fuite d'information ou non.

De manière à construire l'observateur $\mathcal{O}_{\mathcal{M}}^{\Omega}$ en charge de la surveillance de Ω (la fuite d'information sur φ), on construit, sur $S_\Omega = S \times \Omega$, les ensembles F_{S_Ω}, I_{S_Ω}, P_{S_Ω}, N_{S_Ω} (comme décrit dans l'étape 2. de la section 2.1).

Toujours en se basant sur les techniques de la section 2.1, on peut alors calculer le LTS $\chi_{\Omega}^{\Sigma_m}(S)$ sur Σ_m duquel on dérive l'observateur $\mathcal{O}_{\mathcal{M}}^{\Omega}$ qui donne les verdicts suivants pour $\mu \in \mathcal{T}_{\Sigma_m}(S)$:

– $\mathcal{O}_{\mathcal{M}}^{\Omega}(\mu) = Yes$: $\mathcal{M}$ sait que Ω est vérifiée et peut en déduire que $\mathcal{A}$ connaît φ ;

– $\mathcal{O}_{\mathcal{M}}^{\Omega}(\mu) = No$: $\mathcal{M}$ sait que $\mathcal{A}$ ne connaît pas φ mais peut le connaître dans le futur ;

– $\mathcal{O}_{\mathcal{M}}^{\Omega}(\mu) = Inev$: $\mathcal{M}$ sait que Ω sera inévitablement vérifiée et donc que $\mathcal{A}$ va connaître φ ;

– $\mathcal{O}_{\mathcal{M}}^{\Omega}(\mu) = Inev\_Yes$: $\mathcal{M}$ sait que $\mathcal{A}$ connaît déjà ou va connaître φ ;

– $\mathcal{O}_{\mathcal{M}}^{\Omega}(\mu) = Never$: $\mathcal{M}$ sait que Ω ne sera jamais vérifiée. Il peut donc en déduire que $\mathcal{A}$ ne connaîtra jamais φ.

– $\mathcal{O}_{\mathcal{M}}^{\Omega}(\mu) =?$ signifie que $\mathcal{M}$ ne peut déduire aucune information sur Ω. Cela n'implique pas que l'attaquant $\mathcal{A}$ ne connaît rien sur φ. $\mathcal{M}$ et $\mathcal{A}$ observant le système via une interface de communication différente ,il se peut que $\mathcal{A}$ connaisse φ mais $\mathcal{M}$ ne peut pas inférer cette information.

Dans le cas $\mathcal{O}_{\mathcal{M}}^{\Omega}(\mu) =?$, le problème est que $\mathcal{M}$ ne puisse jamais découvrir que $\mathcal{A}$ connaît φ. Ceci correspond à la non-diagnosabilité de Ω. Ce cas de figure peut se produire lorsqu'il existe deux séquences s et s' arbitrairement longues de même observation μ, $s \in \mathcal{L}_{F_\Omega}(\Omega)$ (donc une séquence non-opaque de φ) et $s' \notin \mathcal{L}_{F_\Omega}(\Omega)$. Dans

la section suivante, nous allons donc donner des conditions nécessaires et suffisantes sur le système pour que ce cas de figure ne se produise pas.

4.2. *Condition nécessaire et suffisante pour la détection / prédiction d'intrusion*

Considérons le système S et la propriété Ω décrite dans la section précédente. Intuitivement, S est Ω-diagnosticable [SAM 95, JéR 06] s'il existe $n \in \mathbb{N}$ tel que pour toutes les exécutions s de S telles que $s \models \Omega$, Ω devient non-opaque après au plus n observations supplémentaires. Cela se traduit formellement par

Définition 7 *Soit S un système et Ω une propriété stable sur le système. Alors, S est Ω-diagnosticable si $\exists n$,*

$$\forall s \in \mathcal{L}(S) \cap \mathcal{L}_{F_\Omega}(\Omega),\ \forall t' \in \mathcal{L}(S), t' = s{\cdot}t \wedge \|\pi_{\Sigma_m}(t)\| \geq n \Rightarrow [\![\pi_{\Sigma_m}(s{\cdot}t)]\!]_{\Sigma_m} \subseteq \mathcal{L}_{F_\Omega}(\Omega)$$

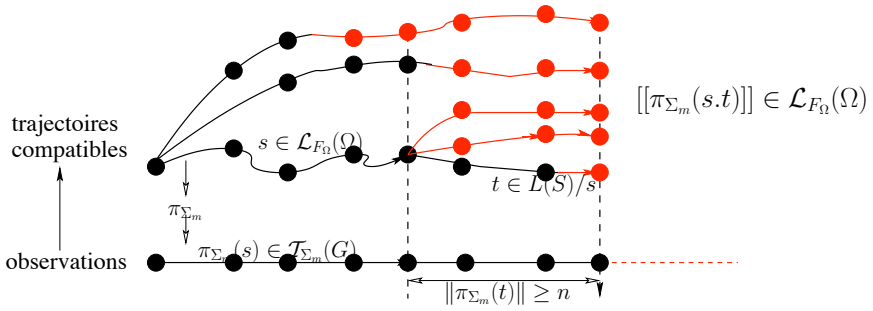


Figure 6 – intuition de la diagnosticabilité

La Ω-diagnosticabilité signifie que lorsqu'une trajectoire s du système satisfait Ω, alors quelque soit le prolongement possible t, t comprenant au moins n événements observables de Σ_m, alors toutes les trajectoires compatibles avec la trace $\pi_{\Sigma_m}(s.t)$ satisfont Ω. Dans notre cadre, ceci signifie que s'il existe $s \in \mathcal{L}(S)$ telle que φ est non-opaque pour $\mathcal{A}$, alors $\mathcal{M}$ le saura après au pire n observations après avoir observé $\pi_{\Sigma_m}(s)$.

Si le système est Ω-diagnosticable, il est alors parfois possible d'affiner le verdict en cherchant non plus à diagnostiquer *a posteriori* que la propriété a été satisfaite, mais en cherchant à le prédire *a priori* [JéR 07].

Définition 8 *S est Ω-prédictif s'il existe $n \in \mathbb{N}$ t.q.*

$$\begin{array}{l}\forall s \in \mathcal{L}(S) \cap \mathcal{L}_{F_\Omega}(\Omega) \cap \Sigma^*.\Sigma_m, \\ \quad \exists t \in (\mathcal{L}(S) \cap \Sigma^*.\Sigma_m) \cup \{\epsilon\},\ t < s\ \wedge\ t \notin \mathcal{L}_{F_\Omega}(\Omega)\ \textit{t.q.} \\ \quad\quad \forall u \in [\![\pi_{\Sigma_m}(t)]\!]_{\Sigma_m}, \forall v \in \mathcal{L}(S)/u, \|\pi_{\Sigma_m}(v)\| \geq n \Rightarrow u.v \in \mathcal{L}_{F_\Omega}(\Omega)\end{array}$$

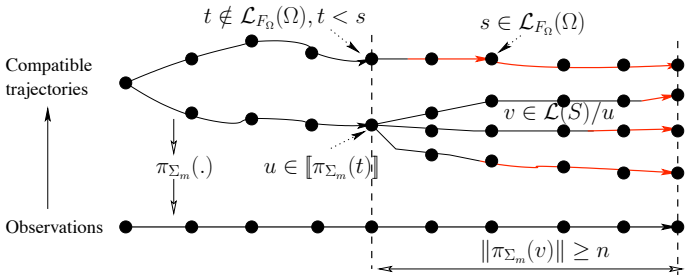


Figure 7 – Intuition de la Ω-prédiction

Cette définition signifie que si une trajectoire s du système satisfait Ω, alors il existe un préfixe t de s qui ne satisfait pas Ω tel que toute trajectoire u compatible avec $\pi_{\Sigma_m}(t)$ sera inévitablement prolongée par des trajectoires qui satisfont Ω. Notons que prédictif implique diagnosticable. Dans notre cadre, cela signifie que $\mathcal{M}$ peut toujours prédire que $\mathcal{A}$ connaîtra φ et donc prendre des mesures adaptées avant que le secret ne soit corrompu.

Vérification de la diagnosticabilité et de la prédictabilité. Tout système n'est pas Ω-diagnosticable (ou Ω-prédictible). Dans la suite de cette section, nous donnons les conditions nécessaires et suffisantes pour que celui-ci le soit. A cet effet, partant de S et de Ω, on commence par construire $S_\Omega = S \times \Omega$ et on calcule les ensembles d'états F_{S_Ω}, I_{S_Ω}, N_{S_Ω}, P_{S_Ω} (C.f. Section 2.1, *étape 2*).

On pose $\Gamma = \textsc{Obs}_{\Sigma_m}(S_\Omega) \times \textsc{Obs}_{\Sigma_m}(S_\Omega)$. On a alors,

Théorème 1 *[JéR 06, JéR 07] Soit S un système et Ω une propriété. Alors*

– S est Ω-diagnosticable ssi il n'existe pas dans Γ de cycles d'états de la forme $F_{S_\Omega} \times (I_{S_\Omega} \cup N_{S_\Omega} \cup P_{S_\Omega})$.

– S est Ω-prédictif ssi $(Pre^{\exists}_{\textsc{Obs}_{\Sigma_m}(S_\Omega)}(F_{S_\Omega}) \setminus F_{S_\Omega}) \times (N_{S_\Omega} \cup P_{S_\Omega}) = \emptyset$ dans Γ.

Exemple 4 *Pour illustrer cette dernière partie, considérons une nouvelle fois le système S et la propriété φ définie dans l'example 2. La propriété Ω modélisant l'ensemble des séquences non-opaques de S relativement à φ est donnée par le LTS décrit en Figure 3.*

Supposons dans un premier temps que l'interface de $\mathcal{M}$ se réduit à $\Sigma_m = \{Z, Y, \delta\}$. On peut alors montrer que S est Ω-diagnosticable, mais n'est pas Ω-prédictif. La fonction d'observation $\mathcal{O}^{\Omega}_{\mathcal{M}}$ qui en découle est donnée par le LTS représenté en Figure 8(a). A contrario, si l'interface de $\mathcal{M}$ est $\Sigma_m = \{X, Y, \delta\}$, alors le système est Ω-prédictif. En effet, après avoir observé X, $\mathcal{M}$, sait que toutes les continuations possibles satisferont Ω et donc qu'une fuite d'information aura lieu (C.f. Figure 8(b)).

$(a)\ \Sigma_m = \{Z, Y, \delta\}$ $(b)\ \Sigma_m = \{X, Y, \delta\}$

Figure 8 – Fonction d'observation $\mathcal{O}^{\Omega}_{\mathcal{M}}$

5. Conclusion

Dans cet article, nous avons montré comment expliciter la fuite d'informations à partir d'un système modélisé par un système de transition fini et d'un secret donné par un langage régulier. Une notion importante développée dans cet article est celle d'inférence de propriété sous observation partielle. En effet, en réutilisant cette notion, nous avons pu présenter de manière homogène les notions d'opacité, de diagnosticabilité et de prédictabilité. Nous avons ainsi pu donner des conditions nécessaires et suffisantes pour permettre à un administrateur de détecter et/ou prédire les fuites d'information. Nous avons aussi présenté comment construire un tel moniteur. Dans une certaine mesure, ceci étend les travaux de Schneider [SCH 00] qui ne considère pas d'observation partielle.

Ce travail ouvre de nombreuses perspectives. On peut envisager notamment d'étendre ces résultats à des modèles plus expressifs tels que les systèmes avec des données à domaines non bornés afin d'appréhender les questions de confidentialité de données. Le point délicat devient alors le calcul de la fonction d'observation qui nécessite de passer par des approximations. D'autre part, nous nous sommes restreint dans cet article à la détection des fuites d'informations. Pour aller plus loin, l'administrateur pourrait vouloir jouer sur les entrées contrôlables de Σ_m afin d'éviter que le secret soit corrompu. Ce travail est donc une première étape nécessaire pour synthétiser un contrôleur sur l'alphabet Σ_m pour éviter la fuite d'information secrète de S. Ceci permettrait d'enrichire l'approche de [BAD 05] dans laquelle tous les événements sont contrôlables. Enfin, nous supposons ici que l'attaquant connaît parfaitement le système et cette hypothèse joue un rôle essentiel. En effet, l'opacité est une propriété sur les ensembles de séquences de même observation. Par conséquent, un attaquant ayant une connaissance partielle du système ne pourra toujours conclure avec certitude qu'une observation révèle de l'information secrète. Cependant on peut rechercher des relations entre une implémentation et son modèle abstrait afin d'assurer que les résultats d'opacité sur le modèle s'appliquent aussi à l'implémentation. Ceci permettrait d'appliquer notre méthologie à des cas pratiques tel que l'analyse de web services où l'accès au modèle complet du système est souvent impossible.

6. Bibliographie

[ALU 06] ALUR R., ČERNÝ P., ZDANCEWIC S., « Preserving Secrecy Under Refinement », *ICALP '06 : Proceedings (Part II) of the 33rd International Colloquium on Automata, Languages and Programming*, Springer, 2006, p. 107–118.

[BAD 05] BADOUEL E., BEDNARCZYK M., BORZYSZKOWSKI A., CAILLAUD B., DARONDEAU P., « Concurrent Secrets », rapport n° 5771, November 2005, IRISA.

[BLA 05] BLANCHET B., M. A., FOURNET C., « Automated Verification of Selected Equivalences for Security Protocols », *20th IEEE Symposium on Logic in Computer Science (LICS 2005)*, Chicago, IL, juin 2005, IEEE Computer Society, p. 331–340.

[BRY 06] BRYANS J., KOUTNY M., MAZARÉ L., RYAN P. Y. A., « Opacity Generalised to Transition Systems », DIMITRAKOS T., MARTINELLI F., RYAN P. Y. A., SCHNEIDER S. A., Eds., *Revised Selected Papers of the 3rd International Workshop on Formal Aspects in Security and Trust (FAST'05)*, vol. 3866 de *Lecture Notes in Computer Science*, Newcastle upon Tyne, UK, 2006, Springer, p. 81-95.

[DAR 06] DARMAILLACQ V., FERNANDEZ J.-C., GROZ R., MOUNIER L., RICHIER J.-L., « Test Generation for Network Security Rules », *TestCom 2006*, vol. 3964 de *LNCS*, 2006.

[JéR 06] JÉRON T., MARCHAND H., PINCHINAT S., CORDIER M.-O., « Supervision Patterns in Discrete Event Systems Diagnosis », *Workshop on Discrete Event Systems, WODES'06*, Ann-Arbor (MI, USA), July 2006.

[JéR 07] JÉRON T., MARCHAND H., GENC S., LAFORTUNE S., « Predictive diagnosis for Discrete Event Systems », rapport n° 1834, March 2007, IRISA.

[SAM 95] SAMPATH M., SENGUPTA R., LAFORTUNE S., SINAAMOHIDEEN K., TENEKETZIS. D., « Diagnosability of discrete event systems », *IEEE Transactions on Automatic Control*, , 1995.

[SCH 00] SCHNEIDER F. B., « Enforceable security policies », *ACM Trans. Inf. Syst. Secur.*, vol. 3, n° 1, 2000, p. 30–50, ACM Press.