Éléments de modélisation pour le test de politiques de sécurité*

Vianney Darmaillacq[1], Jean-Claude Fernandez[2], Roland Groz[1], Laurent Mounier[2], Jean-Luc Richier[1]**

[1] Laboratoire LSR-IMAG
BP 72, 38402 St Martin d'Hères, France
[2] Laboratoire Vérimag
Centre Équation - 2, avenue de Vignate, 38610 Gières, France

Résumé Dans cet article, nous nous intéressons à la notion de conformité de logiciels déployés sur un réseau vis-à-vis d'une politique de sécurité. Plus particulièrement nous nous intéressons à la dérivation automatique de tests à partir d'un modèle formel d'une politique de sécurité. Un certain nombre de questions se posent quant à la méthodologie à adopter. Nous avons abordé ces questions par une étude de cas. Partant de documents régissant la politique de sécurité d'un réseau de campus, nous en avons extrait des recommandations sous forme de règles de sécurité. En analysant une règle, différentes questions peuvent se poser.

Nous situons ce problème par rapport à celui du test de conformité des protocoles. Pour les protocoles de communication, pour qu'une implémentation sous test soit conforme, il suffit qu'après chaque préfixe d'exécution, les réponses observables de l'implémentation fassent parties de celles spécifiées. Ici, la notion de verdicts peut dépendre à la fois d'une analyse "on-line", pendant l'exécution de l'application et d'une analyse "off-line", sur des logs par exemple.

Contrairement au test de conformité des protocoles, où l'implémentation est accessible via une interface homogène (centralisée ou répartie), il n'est pas évident de définir les points de contrôle et d'observation. Les informations que l'on peut observer s'expriment ainsi à différents niveaux : si nous pensons la politique en termes d'automates par exemple, nous sommes confrontés à la correspondance entre différents vocabulaires.

Les formalismes classiquement utilisés pour décrire des politiques de sécurité reposent sur un certain nombre de modalités qui nécessitent différents schémas de test.

1 Objectifs et contexte

Les administrateurs de réseaux et de systèmes informatiques sont chargés de mettre en œuvre et de faire respecter des politiques de sécurité. Le respect

* Ce travail a été soutenu par le projet Potestat de l'ACI Sécurité et le projet IMAG Modeste.

** Email : {Vianney.Darmaillacq,Jean-Claude.Fernandez,Roland.Groz,Laurent.Mounier, Jean-Luc.Richier}@imag.fr

passe par le déploiement de solutions habituelles et la configuration des systèmes destinés à assurer la sécurité à travers des contrôles à différents niveaux. Du fait de l'évolution quotidienne des systèmes, il n'est pas aisé de garantir la cohérence des mises à jour effectuées. Le test peut constituer une façon de vérifier qu'une mise en œuvre reste cohérente avec la politique de sécurité.

Le test du respect d'une politique de sécurité est un test de conformité d'une mise en œuvre (déployée sur l'ensemble des machines en réseau) à une spécification de la sécurité. Il s'agit d'un test fonctionnel a posteriori après déploiement d'un ensemble d'éléments logiciels et matériels. Le test de conformité pour des réseaux a fait l'objet de nombreuses études depuis la mise en place des interconnexions de machines au sein d'architectures ouvertes, en particulier dans le cadre des architectures de type OSI et Internet. Il a donné lieu à l'élaboration de normes [1,2] dédiées au test de conformité des protocoles. Les méthodes de génération automatique basées sur les méthodes formelles ont été largement étudiées que ce soit dans le cadre de la théorie des automates [3] ou dans celui des systèmes de transition [4,5,6,7].

Le test de conformité s'appuie sur des notions essentielles pour une démarche formelle qui permette de développer des outils d'analyse automatique des logiciels déployés pour la sécurité.

- Une modélisation formelle de la politique de sécurité.
- Une définition de l'architecture du test, en référence à l'architecture du système, qui exprime les capacités d'interaction (observation et contrôle) entre le système de test et le système à tester.
- Une formalisation de la relation de conformité (parfois appelée relation d'implantation) entre le système à tester et la spécification qu'il doit respecter.

Dans cet article, nous proposons quelques clés d'analyse de ces notions. Notre objectif principal est de formaliser la notion de conformité vis à vis d'une politique de sécurité. Nous partons d'un exemple et montrons comment les exigences de sécurité et les mécanismes mis en œuvre pour les satisfaire amènent à revisiter le problème du test de conformité.

Notons d'abord que l'interdépendance entre les réseaux et les systèmes d'exploitation des machines conduit à associer les deux à la fois dans la gestion de la sécurité et dans la définition des politiques. C'est pourquoi nous nous intéresserons à des modélisations qui intègrent à la fois les mécanismes de protection (au niveau applicatif) des systèmes d'exploitation et ceux associés aux protocoles des réseaux, également au niveau des couches hautes. Un certain nombre de formalismes ont été proposés pour décrire les politiques à ce niveau ; nous avons en particulier considéré Ponder [8], PDL [9] et OrBAC [10].

Par ailleurs, nous souhaitons aborder le test de la conformité d'un ensemble de machines interconnectées. Des travaux plus spécifiques présentant la modélisation et le déploiement [11,12] ou le test de pare-feux [13] ont déjà été étudiés.

Le travail présenté ici s'inscrit dans le cadre de deux projets auxquels nous participons. MODESTE (MODElisation pour la SécuriTE : test et raffinement en vue d'un processus de certification) est un projet de l'IMAG (Institut d'In-

formatique et de Mathématiques Appliquées de Grenoble). Orienté vers le développement de logiciels certifiés, il s'intéresse en particulier à une modélisation de la sécurité associée aux phases de conception de logiciels embarqués. Dans ce contexte, on associe des propriétés de sécurité aux méthodes ou procédures d'un code logiciel, qu'on doit cependant relier aux exigences de sécurité exprimées dans le SPM (Security Policy Model) selon les critères communs de certification. Dans le cadre d'un test de conformité, on repartirait du SPM pour vérifier l'adéquation d'une implantation. POTESTAT (POlitiques de sécurité : TEST et Analyse par le Test de systèmes en réseau ouvert) est un projet de l'ACI Sécurité du ministère de la recherche. Ce projet rassemble des équipes de l'IMAG et de l'IRISA.

Dans cet article, nous présentons d'abord en partie 2 l'étude de cas qui guidera notre analyse. Nous en avons extrait quelques règles de sécurité significatives. En partie 3, nous analysons comment ces règles pourraient être testées, et les problèmes que soulève la définition des tests. Nous en tirons quelques premières conclusions.

2 L'étude de cas

Nous avons réalisé à l'été 2004 une étude de cas pour recenser les politiques de sécurité appliquées dans le réseau IMAG, qui connecte nos laboratoires [14]. Cette étude comprenait l'analyse de documents fournis aux administrateurs et aux utilisateurs du réseau, et des entretiens. Nous avons collecté des informations à différents niveaux d'administration, du laboratoire au fournisseur d'accès national. L'analyse a permis d'obtenir une description assez large et détaillée d'une politique de sécurité réseau assez typique d'un environnement universitaire.

Pour étudier les problèmes posés par le test de ces politiques, nous avons extrait de cette étude un sous ensemble de règles sur la messagerie. Le jeu de règles a été choisi assez riche pour montrer la plupart des notions à étudier : plusieurs niveaux de politiques et d'organisations interconnectées, variété des services et des méthodes d'accès.

2.1 Description informelle de l'exemple étudié

Dans un réseau, on distingue en général l'intérieur et l'extérieur. L'intérieur du réseau est un ensemble de machines sous la responsabilité d'une organisation. L'extérieur est un ensemble de machines non contrôlées, et considérées, pour la sécurité, comme non fiables. On raffine souvent en considérant d'autres zones, qui diffèrent par le degré d'administration, de fiabilité et de confiance. Les sous-zones de la zone interne correspondent en général à des critères architecturaux, par exemples des séparations entre sous-réseaux physiques, alors que les sous-zones de la zone externes correspondent à des niveaux différents de confiance. Les politiques de sécurité sont exprimées sur ces zones.

On distingue de plus parmi les machines internes celles qui fournissent des services et qui sont accessibles de l'extérieur. Ces machines sont plus visibles et

sont souvent objet d'attaques. C'est pourquoi l'état de l'art en administration de réseau conseille de définir une zone tampon fortement contrôlée pour ces machines. Seules les machines dans cette zone (souvent appelée DMZ ou zone démilitarisée) peuvent communiquer avec l'extérieur, et tout le trafic entre la DMZ et le reste du réseau interne est contrôlé.

La figure 1 est une simplification du réseau IMAG qui indique les flux de trafic autorisés liés au courrier électronique.

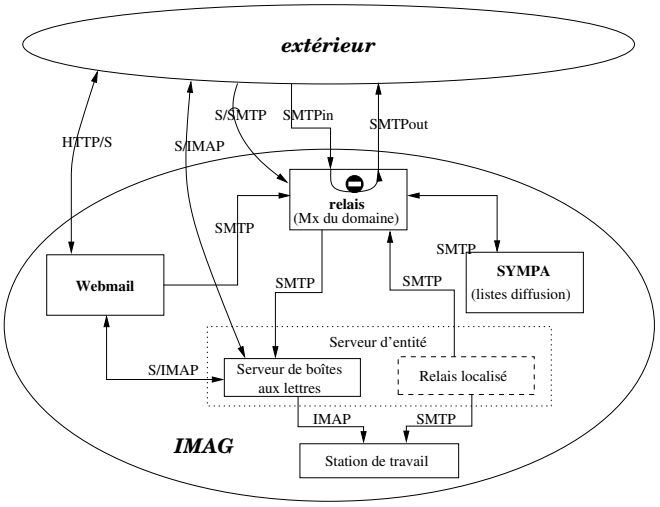

Fig. 1. Trafic courrier dans le réseau IMAG

2.2 Règles de sécurité pour l'exemple étudié

Nous donnons dans la figure 2 un ensemble de règles pour le courrier électronique tiré de notre étude de cas. Cet ensemble ne cherche pas à être exhaustif, il a été choisi pour illustrer les différents aspects d'une politique de sécurité dans un réseau : flux d'informations, séparation en zones, possibilité, obligation ou interdiction de certaines actions, différents niveaux de détails. Pour simplifier le problème, nous supposons que certaines contraintes "globales" sont vraies et n'ont pas à être explicitées : routage correct, systèmes à jour...

3 Problèmes soulevés par le test de règles de sécurité

3.1 Notations

Pour décrire de façon plus formelle les règles de politique de sécurité présentées dans la section précédente nous introduisons tout d'abord un certain nombre de notations : ensembles, fonctions et prédicats.

	Règle
1	Les relais de messagerie ouverts sur l'extérieur doivent être placés dans la DMZ ; On les appellera par la suite "relais principaux".
2	Il n'y aura pas de compte utilisateur sur les relais principaux.
3	Les serveurs de boîtes aux lettres qui contiennent les comptes des utilisateurs seront dans la zone protégée.
4	Les relais principaux sont les seules machines habilitées au dialogue SMTP avec le monde extérieur : le relais du courrier entrant vers les serveurs de boîtes aux lettres et du courrier sortant vers l'extérieur s'effectue par les relais principaux.
5	Les serveurs de boîte aux lettres peuvent être utilisé comme relais internes de messagerie.
6	En entrée de site la politique de filtrage par défaut est que tout ce qui n'est pas explicitement autorisé est interdit.
7	Tous les messages de l'extérieur arrivant sur une machine qui n'est pas un relais principal doivent être redirigés sur un relais principal.
8	Il est interdit de relayer des courriers de l'extérieur vers l'extérieur.
9	[blacklisting] Refus de communiquer avec les relais appartenant à une liste donnée (quotidiennement mise à jour).
10	Des logiciels antivirus et antispams sont installés au niveau des relais de messagerie ou au niveau des boîtes aux lettres. Il est possible de mettre en place une protection au niveau des postes clients.
12	Toute pièce jointe à un courrier électronique doit être contrôlée par un logiciel antivirus au moins une fois avant d'être ouverte (virus, troyens, malwares divers).
13	Tout courrier électronique doit être modifié s'il contient un virus. Le virus est supprimé du courrier et le destinataire en est prévenu.

Fig. 2. Politique pour la messagerie électronique

`Machine` est l'ensemble des machines ; `Address` l'ensemble des adresses ; `Mail` l'ensemble des messages, `Role` = {`mailbox`, `ext-relay` `int-relay`; `station`} indique le rôle qu'une machine peut assurer dans le réseau; `Zone` = {`dmz`, `external`, `internal`, `private`} indique les différentes zones du réseau. A chacun de ces ensembles est associé un prédicat unaire de même nom qui indique l'appartenance d'un élément à cet ensemble.

Les fonctions `src, dest : Mail` $\rightarrow$ `Address` indiquent pour chaque message l'adresse de sa source et sa destination. La fonction `mailbox_of : Address` $\rightarrow$ `Machine` renvoie le serveur de boîte aux lettres.

3.2 Exemple d'un test pour une règle

Considérons la règle 5, qui dit que "Les serveurs de boîte aux lettres peuvent être utilisés comme relais internes de messagerie". En termes logiques, cette règle s'exprime simplement avec une modalité venant de la logique déontique [15,16] et exprimant la *possibilité*, noté ici $\Diamond$:

$$\text{mailbox}(h) \Rightarrow \Diamond\text{intern-relay}(h) \tag{1}$$

Une telle formule semble simple à tester : on choisit une machine h qui est un "serveur de boîte aux lettres", et on vérifie qu'elle peut "relayer" un message :

Contraintes : interior(h), mailbox(h'), mailbox_of(dest(m)) $\neq$ h'

Test : h se connecte à h' et envoie m.

Verdict : si h' accepte m puis le renvoie, la règle est satisfaite.

Le tableau en figure 3 fournit un exemple de cas de test (exprimé dans un formalisme à la TTCN [1]) permettant de tester la règle 5. Intuitivement, ce test est constitué des interactions suivantes :

- Une demande de connexion entre les machine h et h' est émise par le testeur (!connect(h,h')). h doit être une machine interne (interior(h)), et h' doit être un serveur de boîtes aux lettres (mailbox(h')).
- Si cette demande de connexion n'est pas acceptée (?not_ok_connect(h,h')) alors le test est inconcluant.
- Dans le cas contraire (ok_connect(h,h')) h demande alors à h' de transférer le message m dont l'adresse de destination dest(m) est différente de h' (!transfer(h, h', m)).
- Si ce transfert ne peut se faire (?not_ok_transfer(h, h', m)), alors le test échoue.
- Sinon le testeur doit alors observer une demande de transfert du message m entre h' et une machine h" (transfer(h', h", m)) telle que h" est interne (interior(h")) et h" est un relais (relay (h")). Dans le cas contraire le test échoue (h' ne se comporte pas comme un relais interne, le verdict est fail).
- Si cette demande de transfert aboutit (?ok_transfer(h', h", m)) le test est réussi (verdict pass). Si elle échoue (?not_ok_transfer(h, h', m)), alors le test est inconcluant (le transfert n'a pas eu lieu mais la règle 5 est bien respectée par h').

behaviour	**constraints**	**verdict**
!connect(h,h')	interior(h) & mailbox(h')	
?ok_connect(h,h')		
!transfer(h, h', m)	mailbox_of(dest(m)) $\neq$ h'	
?ok_transfer(h, h', m)		
?transfer(h', h", m)	relay (h") & interior(h")	
?ok_transfer(h', h", m)		pass
?not_ok_transfer(h', h", m)		inconc
?otherwise		fail
?not_ok_transfer(h, h', m)		fail
?not_ok_connect(h,h')		inconc

Fig. 3. Cas de test

3.3 Problèmes soulevés

Bien qu'assez intuitif en apparence la définition et l'exécution de ce test soulèvent un certain nombre de questions. Nous en citons ici quelques unes :

- En premier lieu la règle testée est une règle qui exprime une *possibilité*. Une implémentation peut donc être tout à fait conforme à cette règle sans pour autant permettre que des serveurs de boîtes aux lettres soient utilisées comme relais internes. Dans un tel cas le verdict obtenu pour ce test serait systématiquement "inconcluant".
- D'autre part, les événements qui apparaissent dans ce test sont exprimés à un certain niveau d'abstraction. Il resterait par exemple à préciser comment implémenter au niveau du testeur un événement comme " !connect" (envoi d'un PDU particulier), ou l'absence de réponse " ?not_ok_connect" (temporisation, code d'erreurs, etc.). Or ces événements ne se situent pas forcément à une interface avec le testeur ou ne correspondent pas à des actions de communication.
- De plus, les interactions mises en œuvre lors du test se situent à différents niveaux de l'architecture, comme effectuer une demande de connexion sur une machine h, ou tester si une demande de transfert entre une machine h' et h" a correctement eu lieu.
- Enfin, certaines informations recueillies par le testeur peuvent s'avérer non fiables, en particulier si h' est compromis. Par exemple la valeur du paramètre h", déduit du message transmis initialement, peut être erronée. Cela pourrait changer la suite du test ou la pose du verdict.

Tous ces problèmes font que les techniques de test de conformité doivent être étendues pour traiter des politiques de sécurité.

3.4 Architecture de test, contrôle, interfaces

Dans le test de conformité de type boîte noire, on distingue généralement les actions contrôlables, émises par le testeur, des actions observables, observées par le testeur. Dans le cadre du test de conformité, l'implantation est accessible via une interface homogène, c'est-à-dire que la communication entre le testeur et le protocole testé se situe à des niveaux bien définis de la couche réseau (la couche supérieure ou inférieure du protocole).

A l'inverse, dans le cadre du test de politique de sécurité, les actions de contrôle et d'observation effectuées par le testeur peuvent se placer à des niveaux très différents. En effet, une politique de sécurité est définie globalement sur l'ensemble du réseau et concerne donc plusieurs niveaux de l'architecture du réseau. Cette hétérogénéité se retrouve dans le test. Par exemple, tester l'installation de logiciels antivirus (règle 10) ne se situe par au même niveau que vérifier qu'une machine se comporte comme un relais (règle 5). De même, le testeur, qui est inséré comme une machine dans le réseau, a un rôle particulier ; il contrôle les émissions vers d'autres machines, et observe non seulement les messages qui lui sont destinés mais aussi le trafic existant entre deux autres machines. Ces observations externes peuvent être effectuées soit par des "sondes" placées sur le

réseau, soit par exemple par la lecture de "journaux de bord" ou de fichiers de configuration.

3.5 Niveaux de description

Dans la théorie du test de conformité les différents modèles utilisés (spécification et implantation) ont le même vocabulaire d'entrée et de sortie.

Pour les politiques de sécurités, il existe une différence significative de *niveau d'abstraction* entre d'une part la politique de sécurité et, d'autre part, l'implantation effective [17,18]. Ainsi, les prédicats "être un relais", "être une machine interne" ou l'action "transférer un message" sont exprimés au niveau de l'implantation par des séquences ou des arbres d'actions et d'événements de plus bas niveau. Par exemple, "transférer un message m de la machine h vers la machine h"' peut requérir les actions suivantes : une demande de connexion de h vers h', suivie d'une demande de transfert de m de h à h'. Chacune de ces actions entraîne une réponse de h' indiquant si l'action correspondante a été ou non correctement effectuée (acquittement, code d'erreur, etc.).

Cette différence de niveau d'abstraction peut être formalisée par une relation de correspondance partielle qui associe à chaque séquence ou arbre d'exécution abstraite une séquence ou arbre d'actions concrètes. Notons que dans cette correspondance peuvent intervenir des actions fonctionnelles en plus des actions spécifiques à la politique de sécurité. La figure 4 illustre cette notion de raffinement d'actions.

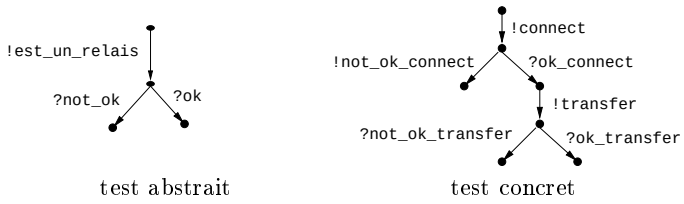

Fig. 4. Un exemple de raffinement d'actions dans un cas de test

En pratique, cette relation de correspondance peut être soit intégrée dans la génération pour produire directement un test concret, soit intégrée au testeur pour transformer lors de l'exécution du test les actions abstraites en actions concrètes.

Un exemple de formalisation d'une telle relation de correspondance est proposé dans [19], mais la notion de raffinement utilisé reste très simple : elle autorise uniquement le remplacement de certaines actions atomiques de contrôle par des séquences d'actions de contrôle. Il reste donc à étendre cette notion pour prendre en compte des schémas de raffinement plus généraux.

3.6 Modèle d'exécution

Dans le cadre du test de conformité, on distingue généralement deux modes d'exécution des tests :

- Le test actif, dans lequel le testeur stimule (contrôle) l'implantation et observe ses réactions. En fonction de celles-ci, il décide du prochain stimulus à émettre.
- Le test passif, dans lequel les stimuli ne sont pas contrôlés par le testeur. Cette séquence est alors appliquée à l'implantation, et une séquence d'observations est obtenue en retour. L'analyse de cette séquence permet au testeur d'émettre un verdict soit à l'issue du test, soit pendant le test si une faute se produit.

Dans le cadre du test de la politique de sécurité ces deux modes restent envisageables.

3.7 Structure des tests

Dans un test actif, l'exécution d'un test consiste en une suite d'interactions entre le testeur et l'implantation, des actions de contrôle et des observations.

Dans le cas du test de conformité, après chaque observation le testeur peut choisir la prochaine action à exécuter, ou le verdict à émettre. Le cas de test peut donc être représenté par une structure arborescente.

Dans le cas du test de politique de sécurité, certains événements observables (interaction entre deux machines distantes, variables d'environnement mises à jour par le système d'exploitation, etc.) ne peuvent pas toujours être connus du testeur pendant l'exécution du test. Ces événements, stockés au fur et à mesure dans un journal de bord, ne sont alors disponibles qu'a posteriori.

Le contrôle effectué par le testeur n'est donc que partiel et un certain nombre de décisions doivent être prises arbitrairement, sans connaître complètement l'état du système sous test.

De même, le verdict de test ne pourra être émis qu'après analyse de la séquence d'exécution, en fonction de l'état atteint dans l'arbre de test.

3.8 Modalités

Les formalismes classiquement utilisés pour décrire des politiques de sécurité (Ponder, OrBAC, PDL, etc.) reposent sur un certain nombre de modalités. Celles-ci doivent être prises en compte pour appliquer différents schémas de test. Par exemple :

La possibilité : "un serveur de boîtes aux lettres peut être utilisé comme relais interne ..." (règle 5).

Tester cette modalité consiste à exhiber une séquence d'exécution qui réalise le comportement attendu (sur l'exemple il s'agit de solliciter un serveur de boîtes aux lettres pour le mettre en situation de relayer un message interne). Ce type de test peut nécessiter de modifier la configuration du réseau, par exemple en introduisant de nouvelles machines (testeur).

Selon les formalismes, cette modalité correspond aux notions de *permission*, ou encore d'*autorisation*.

L'obligation : "toute pièce jointe à un courrier électronique doit être contrôlée par un logiciel antivirus..." (règle 12).

Tester cette modalité consiste à exhiber une séquence d'exécution qui met en défaut la propriété attendue. Pour la règle 12 il pourrait s'agir par exemple d'identifier un ensemble de scénarios "critiques" sur une spécification du service de messagerie. Notons que cette modalité peut être conditionnée par l'arrivée d'un événement, ou la validité d'un prédicat qu'il faudra alors prendre en compte dans le test.

L'interdiction : "il est interdit de relayer des courriers de l'extérieur vers l'extérieur" (règle 8).

Tester cette modalité consiste à essayer d'exécuter une séquence "interdite" et vérifier qu'elle est bien rejetée par l'implantation. En pratique il est souvent nécessaire de raffiner le scénario abstrait décrit par la propriété en y intégrant des actions fonctionnelles.

4 Conclusion

L'étude de cet exemple est encore incomplète, nous n'avons pas encore écrit les cas de tests complets. En effet, pour pouvoir définir un cadre pour la génération automatique de tests, il reste à étudier les modalités définies dans la logique déontique, les liens entre la sémantique de la logique en termes de structure de Kripke et les automates (ou systèmes de transition) représentant les tests. Pour étudier ces liens, qui représentent une correspondance entre deux niveaux d'abstraction, il faudra certainement étudier les notions de raffinement et de répartition.

Cette première expérience nous permet toutefois de penser que le modèle du test de conformité des protocoles peut s'étendre aux politiques de sécurité : nous avons toujours des événements échangés entre composants, des arbres de décision. Nous pouvons donc espérer que les méthodes de génération automatique de test pourront elles aussi être étendues.

Toutefois cette expérience montre aussi que le modèle est plus complexe, à la fois en termes de niveau d'abstraction (les événements ne sont plus de simples échanges de "Protocol Data units"), mais aussi en termes de visibilité d'action et d'accessibilité : on ne peut pas toujours espérer qu'un élément sous test acceptera certaines commandes si on n'a pas un contrôle suffisant (par exemple pour envoyer un message acceptable, il faut connaître un nom de boîte aux lettres valide, ce qui peut être interdit).

Le test de politiques de sécurité est donc plus difficile à mettre en œuvre, et il sera sans doute nécessaire de prendre de compte de multiples architectures de test.

Références

1. ISO : Information Technology, Open Systems Interconnection, Conformance Testing Methodology and Framework. International Standard IS-9646, CCITT X290-X.294, ISO (1991)
2. UIT : Cadre général des méthodes formelles appliquées aux tests de conformité. Recommandation UIT-T Z.500, IUT (1997)
3. Lee, D., Yannakakis, M. : Principles and methods of testing finite state machines - a survey. In : Proceedings of the IEEE, 84(8). (1996) 1090–1123
4. Jéron, T., Morel, P. : Test generation derived from model-checking. In Halbwachs, N., Peled, D., eds. : CAV'99, Trento, Italy. Volume 1633 of LNCS., Springer-Verlag (1999) 108–122
5. Tretmans, J. : Test Generation with Inputs, Outputs, and Repetitive Quiescence. In Margaria, T., Steffen, B., eds. : Second Int. Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96). Volume 1055 of Lecture Notes in Computer Science., Springer-Verlag (1996) 127–146
6. Tretmans, J., Brinksma, E. : Côte de Resyste – Automated Model Based Testing. In Schweizer, M., ed. : Progress 2002 – 3^{rd} Workshop on Embedded Systems, Utrecht, The Netherlands, STW Technology Foundation (2002) 246–255
7. Brinksma, E., Tretmans, J. : An annotated bibliography. In : MOdelling and VErification of Parallel processes. Volume 2067 of Lecture Notes in Computer Science., Springer-Verlag (2001) 187–195
8. Damianou, N., Dulay, N., Lupu, E., Sloman, M. : The Ponder Policy Specification Language. In : International Workshop on Policies for Distributed Systems and Networks. (2001)
9. Bhatia, R., Lobo, J., Kohli, M. : Policy Evaluation for Network Management. In : INFOCOM 2000, 19th Conference IEEE Computer and Communications Societies. (2000)
10. Abou El Kalam, A., El Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miège, A., Saurel, C., Trouessin, G. : Organization Based Access Control. In : IEEE 4th International Workshop on Policies for Distributed Systems and Networks. (2003)
11. Bartal, Y., Mayer, A., Nissim, K., (A.), A.W. : Firmato : A Novel Firewall Management Toolkit. In : IEEE Computer Society Symposium on Security and Privacy. (1999)
12. Cuppens, F., Cuppens-Boulahia, N., Sans, T., Miège, A. : A formal approach to specify and deploy a network security policy. In : Second Workshop on Formal Aspects in Security and Trust (FAST), Toulouse, France. (2004)
13. Senn, D., Basin, D., Caronni, G. : Firewall Conformance Testing. In : TestCom 2005, 17th IFIP TC6/WG6.1 International Conference on Testing of Communicating Systems, Montréal, LNCS 3502. (2005)
14. Boutonnet, S. : Sécurité des réseaux informatiques de l'IMAG. Rapport interne, LSR-IMAG (2004)
15. von Wright, G.H. : Deontic Logic. Mind **60** (1951) pp 1–15
16. Bieber, P., Cuppens, F. : Computer Security Policies and Deontic Logic. In : First Workshop on Deontic Logic and Computer Science, Amsterdam. (1991)

17. Masullo, M., Calo, S. : Policy Management : An Architecture and Approach. In : IEEE First International Workshop On Systems Management, Los Angeles, IEEE. (1993)
18. Wies, R. : Policies in Network and Systems Management - Formal Definition and Architecture. Journal of Network and Systems Management, Plenum Publishing Corp. **2** (1994) pp 63–83
19. van der Bilj, M., Rensink, A., Tretmans, J. : Action Refinment in Conformance Testing. In : TestCom 2005, 17th IFIP TC6/WG6.1 International Conference on Testing of Communicating Systems, Montréal, LNCS 3502. (2005)