Vers une évaluation quantitative de la sécurité informatique

Marc Dacier

Année 1994

THESE

présentée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS
en vue d'obtenir le titre de

Docteur de l'Institut National Polytechnique de Toulouse
Label Européen

Spécialité : Informatique

par

Marc DACIER

Ingénieur civil en informatique
de l'Université Catholique de Louvain

VERS UNE ÉVALUATION QUANTITATIVE DE LA SÉCURITÉ INFORMATIQUE

Soutenue le 20 décembre 1994 devant le Jury :

M.	Jean-Claude LAPRIE	Président
Mme.	Elisa BERTINO	Rapporteurs
M.	Elie MILGROM	Rapporteurs
M.	Alain COSTES	Examinateurs
M.	Yves DESWARTE	Examinateurs
M.	Patrice HUSSON	Examinateurs
M.	Claude NODOT	Examinateurs

Rapport LAAS N° 94488

Cette thèse a été préparée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS
7 avenue du Colonel Roche, 31077 Toulouse CEDEX

A Aurore,

Maxime,

ou whatever ...

Avant-Propos

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) du Centre National de la Recherche Scientifique (CNRS). Je remercie le professeur Alain Costes, Directeur du laboratoire, de m'avoir accueilli au sein du laboratoire et de lui insuffler l'esprit qui y règne. Le plaisir que j'ai retiré de mon travail y est pour beaucoup.

J'exprime ma profonde reconnaissance à Jean-Claude Laprie, Directeur de Recherche CNRS, responsable du groupe "Tolérance aux fautes et Sûreté de Fonctionnement Informatique" (TSF), pour m'avoir accueilli dans ce groupe et accepté de diriger mes recherches. "La porte est toujours ouverte" a-t-il coutume de dire ... et c'est bien vrai. J'ai beaucoup appris devant cette porte ouverte que l'on n'ose pas, à tort, franchir assez souvent.

Je tiens à remercier particulièrement Yves Deswarte, Directeur de Recherche INRIA. Il y a trois ans, il a rendu possible cette belle aventure en m'accordant sa confiance. Durant ma thèse, au cours de discussions souvent passionnées il m'a appris à mieux défendre mes idées. Suivant son exemple, j'ai pu apprendre à reconnaître mes torts et à —un peu— mieux écouter les autres. Ce travail n'aurait pas été le même sans sa rigueur et son perfectionnisme.

Que Mohamed Kaâniche, Chargé de Recherche CNRS, trouve ici l'expression sincère de mon amitié. Toujours présent, toujours disponible pour m'expliquer une chose ou l'autre, toujours franc surtout, il faudra vraiment qu'un jour il m'apprenne comment il y arrive !

J'exprime ma profonde gratitude à Jean-Claude Laprie pour l'honneur qu'il me fait en présidant mon jury de thèse, ainsi qu'à :

- Elisa Bertino, Professeur à l'université de Milan ;
- Alain Costes, Directeur du LAAS-CNRS ;
- Yves Deswarte, Directeur de Recherche INRIA au LAAS-CNRS ;
- Patrice Husson, Project Officer à la Commission des Communautés Européennes, DG3 ;
- Elie Milgrom, Professeur à l'Université Catholique de Louvain ;
- Claude Nodot, Directeur du service dommage risques informatiques à l'Union des Assurances de Paris (UAP) ;

pour l'honneur qu'ils me font en participant à ce jury. Je remercie particulièrement Elisa Bertino et Elie Milgrom qui ont accepté la charge d'être rapporteurs.

Je suis particulièrement fier et heureux de la participation d'Elisa Bertino pour qui la langue dans laquelle est écrite ce manuscrit représente un surcroît de travail.

J'aimerais aussi que mon professeur d'antan, Elie Milgrom, sache l'importance qu'il a eue dans l'accomplissement de ces travaux. Sa curiosité insatiable, son talent de communicateur m'ont donné le goût d'apprendre. Sans lui, il est fort probable que jamais je n'aurais commencé de thèse de doctorat. Je suis heureux qu'il n'en ait pas été ainsi.

Je remercie les membres du comité exécutif du réseau d'excellence européen "Computing Architectures for Basic European Research" (CABERNET) de m'avoir accordé le soutien financier nécessaire à la réalisation de cette thèse.

J'ai eu la chance de travailler au sein d'un groupe uni. J'aimerais en remercier tous les membres avec une pensée particulière pour les doctorants avec qui j'ai partagé moultes conversations animées durant nos repas journaliers. Parmi eux, j'aimerais citer Eric Jenn et Rodolphe Ortalo qui m'ont aidé par la relecture attentive de ce mémoire. Enfin, je salue Joëlle Penavayre, secrétaire du groupe TSF, dont la bonne humeur n'a d'égal que la bonne humeur.

Tout problème technique a une solution et les gens du service "2I" la connaissent. Je parle d'expérience pour avoir tellement souvent fait appel à leur aide. Ils sont trop nombreux pour pouvoir être tous cités mais que ceux que j'ai embêtés le plus durant ces trois ans portent mes remerciements aux autres: Danielle Barthe, Marie-Dominique Cabanne, Matthieu Herrb, Marie-Jeanne Laubies, Jean-Michel Pons, Marc Vaisset.

Je tiens par ailleurs à remercier l'ensemble des membres des services "Documentation-Edition", "Direction-Gestion", "Réception-Standard" pour leur collaboration. Il n'est rien de plus agréable, en effet, que de trouver un sourire derrière la porte que l'on ouvre lorsqu'il y a un problème à résoudre. Irène Pratviel et Michelle Powell, sont, je pense, de parfaits exemples du sourire "laasien", imperméable y compris aux douches belges ... D'autre part, que Anne Bergez trouve également ici mes remerciements les plus sincères pour le travail qu'elle réalise et l'énergie qu'elle communique.

Le LAAS c'est aussi un lieu de rencontre de cultures, de pays, de religions et de langues différentes. Merci à vous tous Luis Fernando Rust Da Costa Carmo, Rosa Maria Leao Rust Carmo, François Dufour, Mohamed Kaâniche, Raul Jacinto Montes, Francisco Vasques de Carvalho de m'avoir enrichi de vos différences.

Mens sana in corpore sano. Si j'ai pu conserver un semblant d'équilibre, c'est grâce à Yves Peremans, véritable ami de toujours, et à nos frasques montagnardes. La vraie vie n'est pas dans les livres, continue à ne jamais me laisser le temps de l'oublier !

Enfin, je ne peux m'arrêter sans une note particulière pour les gens qui me sont particulièrement chers. Ma famille, parents, frère et soeur, loin des yeux mais si proches du cœur, sachez que ce mémoire est votre œuvre car vous m'avez bâti.

A celle qui partage ma vie, mes joies, mes peines, mes lassitudes et mon mauvais caractère, je n'ai rien à dire car dans mes yeux elle lit. Ce travail n'est pas pour elle, elle en a déjà par trop subi les désagréments. Toutefois, lecteur, sache qu'elle a poussé le "sacrifice" jusqu'à en corriger une version préliminaire ! Si, cela, ce n'est pas une preuve d'amour, je ne m'appelle plus ...

Marc Dacier

Sommaire

Introduction Générale

1. L'idée maîtresse de ce travail

Dans ce mémoire, nous nous intéressons au problème de la sécurité des systèmes d'informations. Nous y défendons la thèse selon laquelle il est nécessaire, d'une part, et possible, d'autre part, de se démarquer d'une conception binaire de la sécurité, de type tout ou rien. En ce sens, notre approche est novatrice pour deux raisons fondamentales :

- Nous préconisons d'*évaluer quantitativement* la sécurité des systèmes d'informations plutôt que de qualifier leurs caractéristiques fonctionnelles.
- Nous évaluons la *sécurité opérationnelle* des systèmes, fonction de leur utilisation, plutôt que les seules caractéristiques statiques de la sécurité, fonction de leurs qualités intrinsèques.

Les deux points qui précèdent constituent notre thèse. Dans ce document, nécessité et possibilité d'adopter ce point de vue nouveau sont envisagées en deux étapes successives :

(1) L'étude de l'état de l'art en matière d'évaluation de la sécurité nous permet de dégager les justifications de notre approche.
(2) La définition théorique d'une nouvelle méthode d'évaluation quantitative et son application concrète à un système réel démontrent la faisabilité de notre approche.

2. Le fil conducteur

Dans un premier temps, nous expliquons pourquoi les méthodes actuelles d'évaluation de la sécurité ne peuvent nous satisfaire. Dans ce but, nous examinons les deux écoles de pensée distinctes qui coexistent en matière d'évaluation de la sécurité des systèmes d'information : les différentes familles de critères d'évaluation d'une part, et les méthodes d'analyse des risques, d'autre part.

La première école, issue du "*livre orange*" [TCSEC 1985][140], propose une approche essentiellement fonctionnelle et statique. Les méthodes mises en œuvre évaluent l'efficacité d'un système à résoudre un problème de protection particulier. Elles évaluent également la confiance que l'on peut avoir dans ce résultat. Cette façon de faire se révèle particulièrement bien adaptée aux systèmes qui exigent un haut niveau de protection. En revanche, leur utilisation reste problématique lorsque l'attention se porte sur des environnements qui exigent, à la fois, flexibilité et protection. En particulier, elles gèrent difficilement l'influence de vulnérabilités qui ne peuvent être retirées sans restreindre la facilité d'utilisation.

Nous verrons que les méthodes d'analyse de risques, elles, s'intéressent plus particulièrement à ce dernier type de situation. Tout en analysant les qualités de ces méthodes, nous relevons les critiques émises par leurs détracteurs. En particulier, nous soulignons que la plupart de ces travaux se sont plus ingéniés à définir une méthodologie qu'à modéliser le système à étudier. En cela, elles sont génériques et applicables dans de nombreux contextes. Cependant, nous montrons que cette absence de modèle est préjudiciable dans le cas précis de l'analyse des risques informatiques. La définition d'un modèle de représentation du système, qui puisse également servir durant les phases d'évaluation, apparaît comme un préalable nécessaire à toute méthode d'évaluation des risques informatiques. Il est donc nécessaire de déterminer si de tels modèles existent.

Dans un deuxième temps, fort de l'analyse précédente, nous nous intéressons aux travaux portant sur la modélisation formelle de la sécurité et apportons deux contributions personnelles :

- Nous montrons que les modèles de sécurité sont utilisés pour vérifier formellement qu'un ou plusieurs objectifs de protection sont satisfaits au sein du système étudié. Dans ce but, ils adoptent une *hypothèse de pire cas* sur le comportement des utilisateurs du système : toute intrusion se fait en tirant parti de la complicité possible de tous les autres utilisateurs du système. Nous définissons un nouveau modèle qui permet de s'affranchir de cette hypothèse dans le cadre d'un modèle formel bien connu, le modèle *Take-Grant* [Lipton et Snyder 1977][95].
- Afin de pallier le faible pouvoir d'expression des modèles précédents, nous en proposons un nouveau, le *graphe des privilèges* [Dacier et Deswarte 1994][42] obtenu par enrichissement d'un autre modèle basé sur une matrice de contrôle d'accès typée [Sandhu 1992b][121].

Afin de montrer le bien-fondé de nos propos, nous appliquons ce nouveau modèle dans le cadre des systèmes Unix. Nous montrons comment construire ce graphe et comment l'utiliser pour étudier l'influence, sur la sécurité, de collusions éventuelles d'agresseurs.

Reprenant l'idée d'*évaluer quantitativement* la sécurité, nous montrons ensuite comment dériver à partir de ce graphe de privilèges un modèle mathématique sur lequel calculer un ensemble de valeurs représentatives de l'état de la sécurité du système. Les propriétés mathématiques de ces résultats sont démontrées et nous envisageons un exemple d'utilisation de la méthode, dans sa globalité.

En résumé, nous serons parvenu, au terme de ce mémoire, à justifier les raisons qui motivent notre approche, en nous basant sur l'étude de l'existant. De plus, nous aurons montré comment mettre en œuvre cette vision nouvelle de la sécurité. En effet, nous aurons défini un nouveau modèle de représentation utilisable comme modèle d'évaluation quantitative de la sécurité opérationnelle d'un système. Son application à un système réel aura été donnée et des axes de recherches futures dégagées.

3. Structure du mémoire

Ce mémoire est composé de six chapitres succinctement résumés ci-après :

Le premier chapitre a pour but de montrer les lacunes des approches classiques d'évaluation de la sécurité. Il fixe, de surcroît, la terminologie que nous utilisons tout au long de ce mémoire. Nous y décrivons les travaux actuels en matière de critères d'évaluation et montrons que le problème important de l'adéquation des règles de sécurité d'un système vis-à-vis de ses objectifs de protection n'y est traité que de façon laconique. Dans le même chapitre, nous étudions les méthodes d'analyse des risques et arrivons à la conclusion que les travaux dans ce domaine pèchent par l'importance excessive donnée aux aspects méthodologiques au détriment de la modélisation des systèmes à étudier.

Le deuxième chapitre envisage la modélisation formelle de la sécurité et arrive à la conclusion qu'aucun modèle existant ne peut également servir de modèle d'évaluation. En effet, tous adoptent une hypothèse de pire cas vis-à-vis du comportement des utilisateurs des systèmes étudiés. Dans un premier temps, nous montrons à l'aide du modèle *"Take-Grant"* comment relâcher cette hypothèse. Ensuite, nous proposons un nouveau modèle, le graphe des privilèges, par enrichissement du modèle *"Typed Access Matrix"*, plus efficace que ce dernier pour modéliser les systèmes auxquels nous avons choisi de nous intéresser.

Le troisième chapitre donne, dans le cadre des systèmes d'exploitation UNIX, un exemple d'application concrète du graphe des privilèges.

Au quatrième chapitre, nous montrons comment réaliser l'évaluation quantitative de la sécurité en nous servant du modèle de graphe des privilèges. Ce chapitre est composé de trois parties. Dans la première, nous définissons les variables que nous allons étudier dans cette évaluation, à savoir le temps et l'effort que doit dépenser un attaquant pour violer un objectif de protection. Ensuite, nous définissons le modèle mathématique de représentation choisi et ses relations avec le graphe des privilèges. Enfin, nous nous intéressons au modèle d'évaluation proprement dit et à ses propriétés mathématiques.

Le cinquième chapitre reprend l'ensemble des étapes nécessaires pour mettre en œuvre l'évaluation quantitative de la sécurité. Il se base pour cela sur un prototype expérimental que nous avons développé pour valider les hypothèses théoriques. Un exemple d'application est fourni qui nous permet de mettre en avant les avantages et limites du prototype ainsi que de la méthode.

Enfin, le sixième et dernier chapitre dresse une conclusion critique de ce travail et dégage des axes de recherches qui pourraient utilement le prolonger.

CHAPITRE I *Ecoles de pensée classiques de l'évaluation de la sécurité*

Préambule

Ce chapitre est constitué de trois parties. La première est consacrée à un bref exposé terminologique afin de lever toute ambiguïté sur les termes que nous employons tout au long de ce manuscrit.

Ensuite, nous présentons l'état de l'art en matière de critères d'évaluation de la sécurité. Nous évoquons, en particulier, les travaux américains, canadiens, européens et japonais. Nous montrons que ces méthodes définissent un cadre conceptuel de travail mais restent souvent sibyllines sur la manière de mener à bien les évaluations préconisées. De plus, les travaux les plus récents admettent l'existence de vulnérabilités résiduelles dont il est nécessaire de prévoir les conséquences sur les objectifs de protection, à défaut de pouvoir les retirer du système.

Ce type d'analyse relève, typiquement, des méthodes dites d'analyse de risque ; nous nous y intéressons dans la troisième partie de ce chapitre. Nous y expliquons les concepts principaux de ces méthodes et tentons de clarifier les idées du lecteur en proposant différentes classifications. Nous examinons ensuite plus en profondeur les critiques dont ces méthodes font l'objet ainsi que les arguments de leurs défenseurs. Nous en concluons que ces méthodes contribuent effectivement à l'effort de développement des méthodes de prévision des fautes mais qu'elles pèchent cependant par un manque d'intérêt vis-à-vis de la modélisation des systèmes qu'elles étudient. Se voulant génériques et utilisables dans de multiples domaines d'applications, les travaux les concernant se limitent souvent à définir des méthodologies sans véritablement prendre en compte les spécificités des systèmes étudiés.

I.1 Taxonomie

Tout au long de ce document, nous serons confronté à plusieurs concepts développés au sein de différentes communautés d'esprit qui ne partagent pas toujours le même vocabulaire. C'est pourquoi il nous a semblé utile de préciser dès le départ certaines notions, tout en privilégiant un souci d'abstraction maximale dans les définitions. C'est une tentative de consensus, *à la belge*, sans aucune autre prétention que de faciliter la lecture de ce document. Comme toute solution tempérée, elle rencontrera sans doute les foudres des plus virulents extrémistes de chacune des communautés.

Deux documents, [Laprie 1992][92] et [ARAGO-15 1994][6], conçus dans le cadre général de la sûreté de fonctionnement, sont à l'origine de la terminologie qui suit. Nous utilisons également des notions présentées dans [Landwehr et al. 1993][91] où les auteurs définissent une taxonomie des fautes de sécurité dans les programmes informatiques. D'autres références sont également utilisées par endroit et, le cas échéant, citées dans le corps du texte.

I.1.1 Sûreté de fonctionnement

La **sûreté de fonctionnement** d'un système informatique est la propriété qui permet à ses utilisateurs de placer une confiance justifiée dans le service qu'il leur délivre. Le **service** délivré par un système est son comportement tel que perçu par son, ou ses utilisateurs ; un **utilisateur** est un autre système (humain ou physique) qui interagit avec le système considéré [ARAGO-15 1994][6].

La sûreté de fonctionnement peut être vue selon des propriétés différentes mais complémentaires, qui permettent de définir ses *attributs* :

- Le fait d'*être prêt à l'utilisation* conduit à la **disponibilité** ;
- La *continuité de service* conduit à la **fiabilité** ;
- La *non-occurrence de défaillances catastrophiques* conduit à la **sécurité-innocuité** ;
- La *non-occurrence de divulgations non-autorisées de l'information* conduit à la **confidentialité** ;
- La *non-occurrence d'altérations indésirées de l'information* conduit à l'**intégrité** ;
- L'*aptitude aux réparations et aux évolutions* conduit à la **maintenabilité**.

L'association de la confidentialité, de l'intégrité et de la disponibilité conduit à la **sécurité-confidentialité**. Puisque notre étude ne porte que sur ce seul attribut, nous utilisons dans le reste de ce texte le terme *sécurité* en lieu et place de *sécurité-confidentialité*.

I.1.2 Faute - Erreur - Défaillance

Une **défaillance** du système survient lorsque le service délivré dévie de l'accomplissement de la fonction du système, c'est-à-dire de ce à quoi le système est destiné. Une **erreur** est la partie

de l'état du système qui est susceptible d'entraîner une défaillance. La cause adjugée ou supposée d'une erreur est une **faute** [ARAGO-15 1994][6].

Dès qu'un système devient un tant soit peu complexe, l'expérience montre qu'un objectif de **zéro-faute** est illusoire. En revanche, l'utilisation combinée d'un ensemble de méthodes d'obtention de la sûreté de fonctionnement —*prévention des fautes, tolérance aux fautes, élimination des fautes, prévision des fautes*— aide à tendre vers un objectif de **zéro-défaillance**. Notons dès maintenant, en raison de leur intérêt ultérieur, que les méthodes de prévision des fautes sont utilisées non seulement pour estimer la présence et la création des fautes mais aussi leurs conséquences.

Le travail effectué récemment par Landwehr et al. est très riche d'enseignement sur ce que peut être une **faute** dans le cadre de la sécurité [Landwehr et al. 1993][91]. Nous en retenons que trois paramètres sont nécessaires pour caractériser une faute : *sa genèse, le moment de son introduction* dans le système (phase de développement, de maintenance ou opérationnelle) et *sa position logique* dans le système (système d'exploitation, support, applicatifs, matériel). En ce qui concerne la genèse des fautes, les auteurs relèvent deux classes particulières : les **fautes accidentelles** et les **fautes intentionnelles**. Cette dernière classe se subdivise elle-même en deux sous-classes : les *fautes intentionnelles* **malveillantes,** d'une part, et les **non malveillantes,** d'autre part. Une discussion de ces notions a également été proposée dans [Laprie et Deswarte 1993][93].

Cette taxonomie attire notre attention sur le fait qu'une faute ne désigne pas forcément quelque chose de *mauvais*. En effet, certains systèmes possèdent des fonctionnalités qui peuvent les mener à défaillance dans *certaines* circonstances mais qui sont également nécessaires à leur bon fonctionnement[1]. Dans [Landwehr et al. 1993][91], les auteurs valident leur classification en l'accompagnant de nombreux exemples, nous persuadant par là du bien fondé de leur démarche. Nous reviendrons en temps utiles sur l'importance de l'existence de ces fautes intentionnelles non malveillantes vis-à-vis de la sécurité des systèmes d'information.

I.1.3 Notions de politique de sécurité

"*La **politique de sécurité** d'un système est l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique*" [ITSEC 1991][77]. Elle doit identifier les objectifs de sécurité du système et les menaces auxquelles celui-ci devra faire face.

Cette notion de politique de sécurité peut être affinée en trois branches distinctes : les **politiques de sécurité physique**, **administrative** et **logique**. La première s'occupe de tout ce qui touche, comme son nom l'indique, à la situation physique du système à protéger. En particulier, y sont définies les mesures contre le vol par effraction, le feu, les catastrophes

1. Pour s'en convaincre, le lecteur intéressé est invité à consulter les explications des problèmes liés aux commandes *UNIX* "mkdir" [Tanenbaum 1987][138] et "su" [Bishop 1982][22]. Un résumé peut en être trouvé dans [Landwehr et al. 1993][91] aux pages 40 et 42.

naturelles, les émanations électromagnétiques, etc. Les procédures administratives traitent de tout ce qui ressort à la gestion de la sécurité d'un point de vue organisationnel au sein de l'entreprise ; la sélection et la gestion du personnel responsable de la sécurité des systèmes informatiques en font partie. Enfin, la **politique de sécurité logique**, encore appelée **politique de sécurité technique**, s'intéresse au contenu du système informatique. Elle spécifie *qui* doit avoir accès *à quoi* et dans *quelles circonstances*.

Pour cela, il est nécessaire d'établir une relation claire entre les entités du monde réel (les personnes en particulier) et les objets que le système manipule. Ainsi, tout utilisateur, avant de se servir du système, devra décliner son identité et prouver qu'il est bien la personne qu'il prétend être. Ces deux phases sont définies dans la **politiques d'identification et d'authentification**. Une fois la relation établie, les actions légitimes que peut faire cet utilisateur sont déterminées par la **politique d'autorisation**.

I.1.4 Les objectifs de protection

La politique d'autorisation définit des objectifs de protection qui portent sur trois attributs de la sûreté de fonctionnement : confidentialité, intégrité et disponibilité. Les objectifs de protection stipulent un ensemble de contraintes sur chacun de ces attributs. Cet ensemble de contraintes représente une spécification de la fonction du système de sécurité.

L'implémentation de cette fonction passe par la définition de règles élémentaires qui régissent les droits d'accès que chaque utilisateur possède sur chaque objet du système. L'ensemble de ces règles définit ce que nous appelons le **schéma d'autorisation** [Sandhu 1992a][120]. Il se matérialise sous la forme d'un **moniteur de référence** [Landwehr 1983][90] par où toutes les requêtes d'accès sur des objets doivent transiter afin d'être autorisées, préalablement à leur exécution.

Idéalement, il faut construire les règles de telle sorte qu'aucune séquence valide d'applications des règles ne puisse amener le système dans un état tel qu'un objectif de protection y soit violé. Ceci suppose l'utilisation d'une méthode formelle de construction des règles du schéma d'autorisation à partir d'une spécification formelle des objectifs de protection. C'est, malheureusement, très rarement le cas. Dès lors, il est plus classique de vérifier *a posteriori* que les règles sont **correctes** pour un objectif de protection donné, c'est-à-dire qu'aucune séquence ne mène à un état où l'une des contraintes est violée.

Résoudre ce que l'on appelle **le problème de protection** (*safety problem* [Harrison et Ruzzo 1976][65]) revient à vérifier qu'un schéma d'autorisation est correct vis-à-vis d'un ensemble d'objectifs de protection. Cette tâche a fait et fait encore l'objet de nombreux travaux qui tentent d'offrir un cadre mathématique apte à définir formellement les schémas d'autorisation et les confronter à des objectifs de protection. Nous les décrivons dans le Chapitre III où nous reviendrons plus en détail sur les notions définies ci-dessus. Notons dès maintenant qu'un même modèle formel de sécurité peut servir à représenter différentes

politiques d'autorisation. De plus, chaque politique d'autorisation peut se traduire par la définition de différents schémas d'autorisation et de différents objectifs de protection.

Dans la mesure où nous admettons que la fonction du système de sécurité est définie par les objectifs de protection, nous pouvons dire, au vu des définitions précédentes, qu'il y aura *défaillance* du système de sécurité si une séquence donnée d'actions permises par le schéma d'autorisation mène à la violation d'au moins une contrainte. Une telle séquence constitue une *faute* selon nos définitions précédentes puisqu'elle peut mener le système à défaillance. Son existence peut induire trois attitudes différentes :

(1) Les règles sont modifiées *(élimination de fautes)*;
(2) Des mécanismes annexes sont installés *(tolérance aux fautes)* ;
(3) Les conséquences de la défaillance sont étudiées et, éventuellement, pour des raisons d'efficacité par exemple, l'occurrence potentielle de défaillances est admise *(prévision de fautes).*

Dans ce mémoire, nous soulignons l'existence de fautes intentionnelles non malveillantes qui peuvent effectivement mener à défaillance, au sens défini ci-dessus. Nous expliquons que, dans certains cadres d'applications, le bénéfice retiré de l'existence de ces fautes doit être comparé aux inconvénients que causerait leur élimination. Enfin, nous montrons que la probabilité d'occurrence d'une défaillance, en raison de leur existence, peut se révéler suffisamment faible pour décider de conserver le système en l'état. Bien entendu, une telle attitude n'est possible que si une évaluation quantitative de l'influence de ces fautes a été menée à bien. Définir une méthode apte à réaliser une telle évaluation fait l'objet de ce mémoire. Ce n'est, bien sûr, qu'une des facettes de la problématique de la gestion de la sécurité. Spécifier les objectifs de protection et réaliser leur mise en œuvre en sont d'autres que nous ne traitons pas directement dans ce mémoire.

I.2 Les critères d'évaluation

I.2.1 Historique

En 1967, un groupe de travail fut constitué aux Etats-Unis afin d'étudier les moyens à mettre en œuvre pour protéger les informations classifiées stockées dans les systèmes informatiques. A la suite de la publication de son rapport final, en 1970 [Ware 1970][144], différentes initiatives furent prises qui aboutirent en 1983 à la publication du maintenant fameux rapport "*Trusted Computer Security Evaluation Criteria*", plus connu sous le nom de *TCSEC* ou "livre orange" en raison de la couleur de sa couverture [TCSEC 1983][139]. Ce document devint une norme du Département de la Défense américaine (*DoD*) en 1985 après avoir subi quelques modifications mineures [TCSEC 1985][140]. Pendant des années, ce document fut la seule référence en matière d'évaluation de la sécurité des systèmes informatiques. Il fit cependant l'objet de nombreuses critiques en raison de son manque de souplesse, de sa difficulté de mise en œuvre et de son

incapacité à prendre en compte d'autres contraintes que celles de confidentialité. Il fut à l'origine de plusieurs autres travaux que nous décrivons brièvement ci-dessous et qui ont tous en commun une volonté de flexibilité et d'élargissement du domaine d'application. Il serait illusoire d'espérer décrire en quelques pages une somme de documents qui doit atteindre plusieurs milliers de pages. Aussi ne traiterons-nous dans la suite que des caractéristiques majeures de ces différents critères en privilégiant celles qui servent plus particulièrement notre propos.

I.2.2 Les *TCSEC*

Les *TCSEC* [TCSEC 1985][140] désignent par **produit** tout matériel informatique commercialement disponible capable de traiter des informations de manière automatique. Ils désignent par **système** un produit utilisé dans un contexte opérationnel donné. Les critères offrent une classification à sept niveaux de sécurité, regroupés en quatre classes (*A,B,C,D*). Quatre familles de critères sont définies pour chaque niveau ; elles traitent respectivement de *la politique d'autorisation*, de *l'audit*, de *l'assurance* et de *la documentation*. L'évaluation d'un produit consiste à lui attribuer un des sept niveaux de sécurité. Cette attribution n'aboutit que si le produit répond à tous les critères du niveau en question.

Chaque critère impose un certain nombre d'exigences auxquelles doit se conformer le produit :

- La *politique d'autorisation* stipule une politique précise à suivre en fonction des différents niveaux.
- Le *critère d'audit* précise les fonctions requises en matière d'identification, d'authentification et d'enregistrement des actions des différents utilisateurs.
- Le *critère d'assurance* fixe des recommandations afin d'augmenter la confiance de l'évaluateur en ce que le système implémente bien les fonctionnalités qu'il prétend avoir et en ce qu'elles concourent à la satisfaction de la politique de sécurité.
- Le *critère de documentation* spécifie l'ensemble des documents qui doivent être fournis avec le produit lors de la phase d'évaluation.

La politique d'autorisation distingue deux types de politique : une politique **discrétionnaire** et une politique **par mandats** (encore appelée politique *obligatoire* ou *multi-niveaux*). Contrairement à certaines idées reçues, aucune de ces deux politiques n'exclut l'autre. En réalité, elles sont complémentaires. Les critères permettent l'utilisation d'une politique discrétionnaire jusqu'au niveau C2 mais imposent l'ajout d'une politique par mandats à partir du niveau B1.

Une *politique d'autorisation discrétionnaire* est telle qu'il est possible à un individu de spécifier explicitement le type d'accès que d'autres utilisateurs peuvent avoir sur l'information qu'il contrôle. Ceci permet d'implémenter ce qu'il est couramment appelé le principe de **besoin d'en connaître** (*need-to-know*) : les droits d'accès sont ajustables pour que les utilisateurs aient accès aux informations nécessaires à leur travail, et seulement à celles-là.

Une *politique d'autorisation par mandats* suppose que sujets et objets aient été étiquetés. Classiquement, les objets se voient ainsi assigner un niveau de sensibilité, tandis que les sujets possèdent un niveau d'habilitation. Les règles qui régissent les autorisations d'accès sont basées sur une comparaison du niveau d'habilitation de l'utilisateur et de sensibilité de l'objet auquel il veut accéder. Ces règles assurent que le système vérifie des propriétés générales, de confidentialité ou d'intégrité par exemple. Leur pouvoir d'expression est généralement trop faible pour spécifier une politique complète d'autorisation. En fait, les règles de la politique obligatoire sont souvent utilisées pour stipuler un ensemble de conditions nécessaires mais non suffisantes d'accès à un objet. Ceci explique la complémentarité des deux politiques mentionnées. C'est le cas, par exemple, dans un système où un utilisateur n'a accès à une information que si *i)* il a été jugé digne de confiance pour accéder à une information de ce niveau de confidentialité (*politique par mandats*) et si *ii)* il a besoin d'avoir accès à cette information pour pouvoir accomplir le travail qui lui incombe (*politique discrétionnaire*).

La politique par mandats imposée par le livre orange est celle définie par Bell et La Padula [Bell et La Padula 1976][15]. Sa particularité est d'apporter la preuve formelle que les règles de contrôle d'accès qu'elle définit satisfont effectivement un problème de protection précis. Elle passe par la réalisation de l'objectif de *zéro faute* pour aboutir à celui de *zéro défaillance*, en nous référant à la terminologie définie aux sections I.1.2 et I.1.4. Cette solution, par sa rigueur, est séduisante. Cependant, cette politique de sécurité est loin de rallier tous les suffrages pour les raisons suivantes :

- Elle ne s'intéresse qu'à la seule contrainte de confidentialité or de nombreux domaines d'application requièrent de préserver d'autres attributs tels l'intégrité ou la disponibilité des données. Biba propose dans [Biba 1977][18] un modèle dual de celui de Bell-La Padula où il considère l'intégrité en lieu et place de la confidentialité.
- Elle ne permet pas de rendre compte de pratiques habituelles dans certains secteurs d'activité. En particulier, Clark et Wilson montrent en 1987 [Clark et Wilson 1987][33] qu'elle est inadaptée pour modéliser les notions de transactions bien formées et de séparation des pouvoirs.
- D'autres politiques par mandats, comme celle dite de "la muraille de Chine" [Brewer et Nash 1989][27], sont utilisées dans certains milieux professionnels et ne peuvent que difficilement être représentées par un modèle multi-niveaux [Sandhu 1993][123].

Toutes ces critiques ont suscité la création de nouveaux critères dans différents pays. Bien souvent, et pour des raisons facilement compréhensibles, l'un de leurs objectifs est de permettre à des produits déjà évalués en termes de niveaux *TCSEC* de trouver une place dans leurs propres niveaux sans nécessiter de nouvelle évaluation. Nous allons passer les plus marquants en revue, laissant l'étude des différents modèles de sécurité pour le chapitre III.

I.2.3 Les *CTCPEC*

La première version des critères canadiens, les *Canadian Trusted Computer Product Evaluation Criteria (CTCPEC)*, fut publiée en 1989. A la différence des *TCSEC*, les *CTCPEC* s'affranchissent de la classification en sept niveaux. Ils évaluent distinctement l'existence de services de sécurité et leur efficacité attendue. De plus, ils évaluent également séparément un niveau d'assurance qui indique la confiance que l'on peut avoir dans la façon dont les mécanismes choisis implémentent les services de sécurité. Ces critères sont fréquemment remaniés et la version la plus récente en est la version 3.0.e, datée de janvier 1993 [CTCPEC 1993][74]. La description qui suit en est issue.

Dans la terminologie des *CTCPEC*, un **produit** est défini comme un ensemble de services fonctionnels. Il existe quatre grandes familles de **critères fonctionnels** : *confidentialité, intégrité, disponibilité* et *audit*. Chaque famille est divisée en **services**. Chaque service peut être implémenté par un ou plusieurs **mécanismes.**

Le résultat de l'évaluation délivre pour chaque service une mesure de l'efficacité du produit à fournir le service donné. Ce résultat est donné sous la forme de couples *xxx.zz* où *xxx* représente le service et *zz* l'efficacité. Ces doublets sont regroupés par famille de critères. L'assurance, quant à elle, est une valeur globale qui symbolise la confiance qu'a l'évaluateur en l'implémentation des mécanismes conçus pour fournir les services.

Il se peut qu'un service donné ne puisse être fourni efficacement que si un autre service est également disponible. Par exemple, un produit qui n'inclut pas de service d'identification ne peut prétendre fournir un service de gestion de politique d'autorisation, quand bien même les mécanismes de ce dernier seraient présents dans le système. Cet aspect est pris en compte dans la façon dont est évaluée l'*efficacité* d'un service. Le niveau d'efficacité de chaque service est conditionné par la vérification de **contraintes** portant sur le niveau d'efficacité d'autres services.

Alors que les *TCSEC* rangent les produits et systèmes selon sept niveaux, les *CTCPEC* n'en limitent pas le nombre. Le choix du niveau répondant le mieux à un ensemble précis d'objectifs de protection en devient d'autant plus délicat. Pour pallier ce problème et assurer une compatibilité avec les notations définies dans les *TCSEC*, les *CTCPEC* préconisent la définition de **profils**. Les profils sont des ensembles de paires (*service-efficacité*) susceptibles de répondre à un marché potentiel. En particulier, les *CTCPEC* définissent sept profils correspondant aux sept niveaux définis dans les *TCSEC*.

I.2.4 Les *ITSEC*

La Communauté Européenne a adopté provisoirement les critères "harmonisés" ou *ITSEC* [ITSEC 1991][77], résultats de l'harmonisation des travaux réalisés précédemment au sein de quatre pays européens : l'Allemagne, la France, les Pays-Bas et le Royaume-Uni

[Le Roux 1991][94]. C'est d'ailleurs l'existence, antérieure aux *CTCPEC*, de ces travaux qui explique la grande similitude entre les approches proposées dans les *ITSEC* et les *CTCPEC*.

En effet, la philosophie des deux documents est fort semblable. En particulier, la dualité fonctionnalités-assurance est également considérée ici, bien que différemment. Les *CTCPEC* attribuent un niveau d'assurance dans l'*efficacité* de chaque service et évaluent l'assurance dans la *conformité* au niveau du système/produit global. Les *ITSEC*, eux, scindent également l'évaluation de l'assurance en deux : conformité et efficacité, mais ces deux évaluations portent sur l'ensemble des fonctionnalités et résultent en une valeur finale globale. Les fonctionnalités quant à elles, sont regroupées en classes, semblables dans l'esprit aux profils des *CTCPEC*. Ces classes, complétées par un niveau d'assurance, permettent d'établir un parallèle avec les niveaux *TCSEC* [Brewer et al. 1991][28].

Les *ITSEC* ont pour but de définir des critères d'évaluation applicables tant aux systèmes qu'aux produits. Aussi utilisent-ils le terme de "cible d'évaluation" (*TOE* : Target Of Evaluation). Le contenu exigé d'une *TOE* comprend : une politique de sécurité du système, une spécification des fonctions requises dédiées à la sécurité, une définition des mécanismes de sécurité requis (optionnelle), la cotation annoncée de la résistance minimum des mécanismes, le niveau d'évaluation visé.

Il est intéressant de se pencher plus longuement sur les critères qui prévalent à l'établissement de l'assurance que l'on a de l'*efficacité* d'un système. Ils sont scindés en deux grandes classes : les critères d'efficacité portant sur la **construction** et sur l'**exploitation**. Les critères de construction comportent quatre aspects :

- *Analyse de pertinence* : les fonctions de sécurité choisies concourent-elles à la satisfaction des objectifs de protection ?
- *Analyse de cohésion* : les fonctions forment-elles un ensemble intégré[1] et efficace ?
- *Analyse de résistance des mécanismes* : quelle est la capacité des mécanismes choisis à résister à une attaque directe en tirant profit d'insuffisances dans leurs algorithmes, leurs principes ou propriétés sous-jacentes ?
- *Liste des vulnérabilités connues dans la construction* : quelle est l'influence sur les objectifs de protection de vulnérabilités de construction, telles que les moyens de désactiver, court-circuiter, altérer ou contourner des fonctions et mécanismes dédiés à la sécurité ?

Les critères d'exploitation comportent deux parties :

- *Analyse de la facilité d'emploi* : la *TOE* peut-elle être configurée, en exploitation, de manière non sûre sans qu'un administrateur ou un utilisateur final ne puisse s'en apercevoir ?
- *Liste des vulnérabilités en exploitation connues* : quelle est l'influence de vulnérabilités d'exploitation sur les objectifs de protection ?

1. Cet aspect est à mettre en relation avec la définition, dans les *CTCPEC*, des contraintes entre les services.

Ces critères, et plus particulièrement les quatre derniers, montrent que, dans l'esprit des évaluateurs, un système sûr ne répond pas nécessairement au critère de zéro-faute. En effet, ils envisagent qu'un système puisse contenir des fautes (*intentionnelles non malveillantes*) introduites dans les phases de construction ou d'exploitation. Ils exigent que, dans ce cas, soient apportés des éléments de preuve pour garantir (i) que la faute ne peut mener à défaillance pour l'objectif de protection envisagé, (ii) qu'elle n'existe pas dans la pratique ou (iii) qu'elle peut être convenablement contrée par des mesures de sécurité techniques, organisationnelles, physiques ou relatives au personnel. Un système sûr serait-il donc, dans l'esprit des évaluateurs, un système qui répond au critère de zéro-défaillance ?

Nous pourrions le croire, n'était l'existence du critère d'analyse de résistance des mécanismes[1]. Ce critère correspond à une approche de prévision de fautes et laisse apparaître que la sécurité du système peut être compromise si une attaque lui est appliquée, d'une intensité supérieure à la résistance du mécanisme incriminé. Une telle hypothèse revient à admettre qu'un système peut ne satisfaire ni au critère de zéro-faute, ni au critère de zéro-défaillance. Un aperçu de la façon de mener à bien ce type d'analyse est donné en annexe du document intitulé "*Information Technology Security Evaluation Manual*" (*ITSEM*, [ITSEM 1993][78]) qui détaille la méthodologie à mettre en œuvre pour réaliser une évaluation d'un système selon les critères *ITSEC*. Par ailleurs, la philosophie générale des *ITSEM* est expliquée dans [Klein et al. 1992][84].

I.2.5 Les Federal Criteria

En 1992, le *National Institute of Standards and Technologies* (*NIST*) et la *National Security Agency* (*NSA*) américains ont publié une version préliminaire de nouveaux critères d'évaluation appelés "critères fédéraux" [Federal Criteria 1992a][50], [Federal Criteria 1992b][51]. Ces documents sont le résultat d'une coopération entre les Etats-Unis et le Canada mais n'ont jamais fait l'objet d'une publication finale officielle. Plus qu'une fin en soi, ces documents doivent être perçus comme la contribution nord-américaine à l'effort d'harmonisation internationale qui devrait trouver son aboutissement dans la définition de *Critères Communs*.

Les Critères Fédéraux mettent en exergue la notion de **profil de protection**, relativement semblable à la notion canadienne. Un profil définit un ensemble de fonctionnalités, le niveau d'assurance requis sur le processus de développement ainsi que sur le processus d'évaluation.

Dans l'esprit de ces critères, trois phases distinctes sont à considérer pour l'évaluation d'un système : **enregistrement**, **évaluation**, et **certification**

La première phase consiste à définir les contraintes d'un produit de sécurité donné sous la forme d'un profil de protection. C'est dans cette première phase que l'on examinera, en particulier, si les objectifs de protection sont satisfaits par les fonctions choisies.

1. Cette notion de "résistance" était déjà présente dans les autres critères évoqués, *TCSEC* et *CTCPEC*, mais toujours de façon très laconique et simpliste. Les *ITSEC*, au contraire, intègrent dès la définition de la *TOE* le niveau de résistance minimum requis des divers mécanismes.

Ensuite vient la phase d'évaluation proprement dite où l'on analyse les caractéristiques d'un produit vis-à-vis des spécifications d'un profil de protection.

Enfin, la phase de certification consiste à considérer un ou plusieurs produits —déjà évalués vis-à-vis d'un profil de protection— dans un environnement opérationnel précis. Cette phase est le feu vert à recevoir pour pouvoir utiliser le système.

Cette version n'étant qu'une ébauche préliminaire, nous ne tenons pas à nous y attarder. Retenons simplement qu'ici aussi l'étude des fonctionnalités est distincte de l'évaluation de la confiance en leur efficacité et conformité. L'évaluation peut être menée à plusieurs niveaux d'intensité, délivrant différents niveaux de confiance. Enfin, tout en gardant une compatibilité totale avec les *TCSEC* (il suffit de définir les profils de protection adéquats), cette méthode offre une grande souplesse. Des exemples de profils de protection sont donnés dans [Federal Criteria 1992b][51].

I.2.6 Les autres approches

Le Japon a, lui aussi, publié en 1992 une version préliminaire de ses propres critères d'évaluation [JCSEC 1992][81]. Ce document est très incomplet. Il dresse un bref historique des travaux en cours et se positionne dans la lignée des *ITSEC*. La majeure partie du document est consacrée à l'identification de toute une série de contraintes que doivent remplir les fonctions de sécurité ; chaque contrainte est accompagnée d'une liste de vulnérabilités potentielles dont il faut veiller à se prémunir.

La communauté européenne, de son côté a mis sur pied en mai 1992 un groupe de réflexion, le *SOG-IS* (Senior Officials Group - Information Security), chargé de proposer les lignes d'action prioritaires en matière de sécurité des systèmes d'information ; un document, dénommé "**livre vert**" [Green Book 1993][59], reprend les réflexions de ce groupe. On y trouve, en particulier, l'expression du besoin pressant d'harmoniser les différentes approches évoquées ci-avant sous une forme unique.

Cette harmonisation devrait s'appuyer principalement sur les *ITSEC*/*ITSEM* et les *Federal Criteria*. Sa réalisation est en cours et une ébauche devrait être publiée à la fin de l'année 1994. A plus long terme, elle devrait aboutir à une norme *ISO*. Malheureusement, nous ne disposons pas de ce document au moment où nous écrivons ces lignes. On peut raisonnablement penser, cependant, qu'il sera semblable dans l'esprit à ceux que nous avons évoqués précédemment (*ITSEC*, *CTCPEC*, *Federal Criteria*)

Il est bon de noter qu'apparaît de plus en plus au sein de la communauté scientifique la volonté d'étudier de façon globale les problèmes de sûreté de fonctionnement, sans plus cloisonner ses différents attributs ; nous en prenons pour témoins une récente table ronde où ce problème fut évoqué ([Deswarte 1994][45], [Neumann 1994][109], [Randell 1994][113]) et les réflexions en cours pour élargir les critères d'évaluation à la disponibilité et à la fiabilité [Abrams et Toth 1994][1].

I.2.7 Conclusions

Les critères d'évaluation abordent toute une série de problèmes conceptuels primordiaux : évaluation de l'efficacité, de l'assurance, de la conformité, etc. Cependant, la mise en œuvre de ces évaluations reste souvent difficile. Ainsi en est-il, par exemple, de l'évaluation de l'adéquation du schéma d'autorisation aux objectifs de protection ; dans les *TCSEC*, les règles sont figées et l'adéquation est prouvée formellement. Ceci conduit à l'assurance de **zéro-défaillance par une méthode de prévention de fautes**. Or, il ressort des travaux plus récents qu'un objectif de zéro-faute est peu réaliste dès que les objectifs de protection deviennent plus complexes que ceux envisagés dans les *TCSEC*. Ces travaux soulignent l'existence inévitable de **vulnérabilités résiduelles** dont il faut tenir compte lors de l'évaluation de la sécurité. Dans ce but, les *ITSEM* envisagent par exemple l'étude de la résistance des mécanismes de protection et la mise en œuvre de tentatives de pénétration [ITSEM 1993][78]. Cependant, dans leur état actuel, ces approches se limitent à l'étude de chaque faute indépendamment des autres. En cela, elles s'écartent de la philosophie générale des méthodes de prévision qui envisagent l'influence de l'ensemble des fautes sur les objectifs de sécurité.

En résumé, l'analyse des différents critères montre qu'il est **nécessaire** de mettre en œuvre des méthodes de prévision des fautes mais la façon de procéder n'est pas définie. Elle reste un problème théorique ouvert. Afin de trouver des éléments de réponse à cette question, nous considérons à présent les méthodes d'analyse des risques qui, précisément, ont pour objectif d'évaluer les conséquences, sur la sécurité, de l'existence de vulnérabilités résiduelles.

I.3 Les méthodes d'analyse des risques

I.3.1 Introduction

Dans un premier temps, nous montrons la différence existant entre les trois concepts d'analyse, d'évaluation et de gestion des risques. Ensuite, nous détaillons six phases qui entrent généralement en jeu lors d'une analyse des risques d'un système. Nous raffinons ce point de vue en montrant les différentes manières, orthogonales les unes par rapport aux autres, dont nous pouvons classer les méthodes existantes. Enfin, nous mettons en exergue les principaux sujets de polémique et concluons en indiquant que le principal grief qui peut être fait aux méthodes d'analyse des risques est d'avoir privilégié une approche méthodologique au détriment d'une modélisation adéquate des systèmes étudiés.

I.3.2 Les risques : analyse, évaluation, gestion

Un risque peut être vu comme un événement redouté. On peut lui attribuer une fréquence d'occurrence et un coût. En ce sens, cette notion se rapproche de la notion de défaillance définie précédemment.

Trois expressions reviennent couramment dans la littérature : l'analyse des risques (*"risk analysis"*), l'évaluation des risques (*"risk assessment"*) et la gestion des risques (*"risk management"*). L'analyse des risques est le concept le plus large[1], englobant les deux autres. Chacun de ces termes se définit comme suit [Hoffman 1989][74], [Anastovski 1993][3] :

- *L'analyse des risques* est une discipline analytique destinée à permettre l'implémentation d'une politique de sécurité en tenant compte de rapports coûts/bénéfices dans un environnement réel et complexe. Elle est composée de deux parties : l'*évaluation des risques* et la *gestion des risques*.
- *L'évaluation des risques* consiste à identifier les risques présents au sein d'un système, leurs conséquences et à évaluer l'efficacité des parades associées.
- *La gestion des risques* consiste à sélectionner les parades qui permettent de minimiser l'exposition des biens aux risques tout en prenant en compte des contraintes pragmatiques d'ordre social, politique et financier.

La *gestion des risques* est typiquement dévolue aux preneurs de décision. Elle utilise les résultats de l'évaluation des risques qui, elle, a une forte connotation technique. La subdivision entre ces deux catégories peut sembler anodine au premier abord mais elle introduit un aspect non encore évoqué jusqu'ici : la *non-optimalité technique des parades choisies*. En effet, à son niveau de compétence, l'évaluateur va déterminer un ensemble de parades qu'il considère optimal mais cette proposition peut être remise en cause et amendée lors de la prise de décision qui, en raison de contraintes externes, peut ne pas se satisfaire d'une solution optimale du seul point de vue technique.

Cette approche est sans doute plus courante dans d'autres domaines de la sûreté de fonctionnement. Ainsi en est-il de l'industrie automobile et des systèmes visant à assurer la sécurité-innocuité des passagers. La plupart des constructeurs, par exemple, ont adopté le système du coussin d'air gonflable (*airbag*) pour protéger le conducteur en cas de collision et ont renoncé à développer d'autres approches prometteuses en raison de l'engouement du public pour ce système. Rien n'indique cependant que cette solution soit techniquement meilleure que celle qu'ils avaient initialement choisie[2]. Nous verrons plus loin que le principe d'un choix technique non-optimal existe également dans le domaine de la sécurité informatique.

I.3.3 Les caractéristiques communes

Sous la bannière "analyse des risques", existe une myriade de travaux différents, allant de la simple liste de contrôle (*check-list*) aux bases de connaissances en passant par l'étude de scénarios de défaillance. Cette profusion de méthodes aux qualités inégales entache la réputation de ce domaine de recherche et ne permet pas d'en avoir une vision claire.

1. D'autres auteurs, tel [Guarro 1987][61], envisagent la gestion de risques comme le concept principal englobant analyse et évaluation.

2. Un constructeur, notamment, a cessé d'installer dans ses voitures les colonnes de direction rétractibles alors que plusieurs modèles de la marque en étaient équipés avec succès depuis plusieurs années.

Nous essayons cependant de dégager quelques principes fondateurs ainsi que des phases communes à la plupart des méthodologies. Pour cela, notons que le résultat attendu de l'application de toute méthode d'analyse des risques est une prise de décision. En effet, ces méthodes doivent permettre de motiver un choix de moyens de protection à mettre en œuvre pour protéger des intérêts. Cela nécessite l'identification de ces intérêts, des parades possibles et des risques potentiels ; ceci explique que l'on retrouve classiquement les six étapes suivantes dans ces méthodes [Zviran et al. 1990][149] :

(1) *Identification des actifs* du système d'information : les actifs représentent les éléments que l'on désire protéger dans le système.
(2) *Identification des menaces et calcul des risques* : il est nécessaire d'identifier les menaces qui guettent chaque actif identifié à l'étape précédente. Pour chaque menace, il faut calculer le risque qu'elle représente. Ce risque est égal à la fréquence d'occurrence de la menace multipliée par la conséquence, en termes financiers, de sa réalisation.
(3) *Analyse des conséquences* : cette étape estime les conséquences de l'occurrence d'une menace sur les actifs du système d'information.
(4) *Evaluation des parades existantes* : il est nécessaire d'identifier non seulement les contre-mesures existantes mais également leur efficacité.
(5) *Evaluation du niveau de sécurité existant* : en fonction des résultats qui précédent, on estime le niveau de sécurité et on effectue un choix de parades optimales d'un point de vue coût/efficacité pour améliorer ce niveau.
(6) *Formulation d'un plan de sécurité* : le résultat final de l'étude débouche sur un compte rendu des résultats précédents ainsi que sur un plan de mise en œuvre de mesures de protection complémentaires.

Certaines méthodes s'écartent plus ou moins de ce canevas générique ; il offre toutefois une bonne idée des lignes de force qui sous-tendent le domaine. Il est important de noter dès maintenant que ces méthodes prônent l'interdisciplinarité tant dans les domaines d'application qu'elles couvrent que dans les méthodes d'investigation qu'elles utilisent. Elles définissent une méthodologie apte à prendre aussi bien en compte les risques de catastrophes naturelles (inondation, orage, tremblement de terre) que les dégradations volontaires du matériel (terrorisme, hold-up) ou même les intrusions au sein du système informatique. Nous verrons que cette volonté de tout couvrir est considérée à la fois comme un atout par les partisans de ces méthodes, et comme une faiblesse par leurs détracteurs.

I.3.4 Classification des méthodes

Le nombre de méthodes différentes est tel et leur description souvent si laconique qu'une classification rigoureuse représenterait un travail fort difficile. Cependant, afin de faire sentir au lecteur la diversité de ces méthodes, nous avons tenté de souligner quelques uns de leurs traits caractéristiques et nous présentons dans les pages qui suivent divers points de vue qui permettent de les regrouper. Nous évoquons ensuite de façon globale les avantages et inconvénients des méthodes d'analyse de risque. Il est nécessaire de noter que notre approche

constitue une synthèse réalisée à partir, notamment, d'autres classifications déjà publiées [Hellewell et al. 1992][66], [Anastovski 1993][3], [Baskerville 1993][14], [Anderson 1991][4], [Saari 1991][118]. En raison de certaines divergences entre ces documents, notre analyse ne peut être considérée comme un reflet fidèle d'aucun d'entre eux mais plutôt comme le fruit d'une interprétation personnelle, complétée par la lecture d'autres références données dans le cours du texte.

Un autre facteur qui a motivé notre décision de ne pas décrire ici telle ou telle méthode en détail est la difficulté de faire un choix représentatif des méthodes à évoquer. En particulier, le succès de certaines n'est pas tant dû à leurs qualités intrinsèques qu'à une certaine pression sociale qui pousse à les utiliser au sein de milieux professionnels précis. C'est notamment le cas de l'influence du monde de l'assurance sur l'utilisation de *MARION* en France ([Lamère 1986][86], [Grissonanche 1987][60]), de *CRAMM* au Royaume-Uni [Farquhar 1991][49], ou encore d'une méthode propre au département australien de la défense [Hellewell et al. 1992][66]. Ces méthodes étant souvent confinées au niveau d'une nation, il est difficile d'en avoir une vision critique objective : soit elles n'ont pas reçu l'attention qu'elles méritent des autres pays, soit, au contraire elles ont été critiquées pour des raisons plus politiques que techniques. Ainsi, une méthode qui, ici, sera considérée comme "la" référence, sera totalement ignorée ailleurs, et vice-versa.

I.3.5 Différents points de vue

Chaque section définit des caractéristiques communes à une famille de méthodes mais une même méthode peut apparaître dans plusieurs familles car les points de vue offerts sont orthogonaux entre eux. Ils sont au nombre de cinq :

(1) Pour quel domaine d'application la méthode a-t-elle été conçue ?
(2) Quel niveau d'abstraction la méthode utilise-t-elle ?
(3) Quels sont les éléments à protéger dans le système ?
(4) Comment les paramètres de l'évaluation peuvent-ils être exprimés ?
(5) Quelle est la logique qui guide la mise en œuvre de la méthode ?

A. Domaines d'applications

Il est classique de distinguer les méthodes issues d'organismes militaires ou gouvernementaux de celles que l'on retrouve dans les systèmes commerciaux, industriels ou financiers. Cette vision tend à s'estomper avec le temps car les techniques qu'elles manipulent sont de plus en plus semblables. En revanche, la notion de valeur des actifs y est fort différente, comme nous allons le voir.

i) Les systèmes "militaires" ou "gouvernementaux"

Nous considérons ici les méthodes d'analyse des risques dont le cadre d'application est celui des système militaires ou gouvernementaux. Elles sont utilisées pour décider, dans un environnement opérationnel donné, quel niveau de sécurité, au sens *TCSEC* du terme, doit

posséder le système étudié. Cette décision est prise en fonction des informations qu'il contient et des utilisateurs qui y ont accès. Les quelques points suivants caractérisent ces méthodes :

- Elles s'intéressent au sous-ensemble des risques liés à la sécurité logique, par opposition à la sécurité physique (feu, incendie, émanation électro-magnétique, etc.).
- Elles sont directement liées aux critères d'évaluation évoqués précédemment.
- La valeur des actifs n'y est pas considérée en des termes financiers mais en fonction des niveaux de classification de l'information.
- Le risque est directement proportionnel à la distance entre le plus bas niveau d'habilitation des utilisateurs du système et le plus haut niveau de sensibilité des informations contenues dans le système.
- Elles s'intéressent surtout à la confidentialité.

Parmi les approches qui appartiennent à cette famille, on peut citer, comme dans [Hellewell et al. 1992][66], le livre jaune, la méthode de Landwehr et Lubbes, les méthodes *DADPSWG*/88-1, Logicon ou encore *ANSSR* [Bodeau et al. 1990][24].

ii) Les systèmes "commerciaux"

Les systèmes commerciaux ne s'intéressent pas qu'à la seule confidentialité de leurs informations. Ils ne peuvent se satisfaire des méthodes précédentes centrées sur ce seul attribut et c'est pourquoi d'autres méthodes existent qui se caractérisent comme suit :

- La classe des risques couverts est souvent aussi large que possible. L'analyse de risque y est envisagée comme une démarche globale. La sécurité logique n'en est qu'un des aspects.
- Il est nécessaire d'identifier et de fixer les valeurs des actifs du système, souvent en des termes financiers, pour permettre le choix des parades.
- Elles débouchent sur un ensemble de parades à installer plutôt que sur le choix d'un niveau *TCSEC* de sécurité global à implémenter.

B. Niveaux d'abstraction

Baskerville, dans [Baskerville 1993][14], envisage le problème de la spécification de la sécurité de systèmes informatiques. Selon lui, la mise en œuvre de la sécurité est fonction de la méthode de spécification utilisée. Il dégage trois générations de méthodes de spécification et, par là, définit trois générations de méthodes de développement de la sécurité des systèmes. En réalité, son analyse assimile l'utilisation de l'analyse des risques à une mise en œuvre de la sécurité. Ceci l'amène à décrire trois familles de méthodes d'analyse des risques qui, conceptuellement, diffèrent par le niveau d'abstraction qu'elles offrent par rapport à la définition du système et de ses risques. Son point de vue n'en est pas moins intéressant pour autant.

i) Première génération

La première génération se caractérise par l'adoption de trois hypothèses :

(1) Il existe une liste de toutes les parades utiles à la protection des ressources informationnelles.
(2) Chacune de ces parades est applicable à tout système. Ceci implique l'universalité des solutions à apporter aux objectifs de protection.
(3) Le but ultime d'une méthode est de parvenir à diminuer la probabilité d'occurrence d'une attaque ou le coût résultant d'une attaque réussie.

De façon concrète, toutes les méthodes les plus anciennes basées sur des listes de contrôle appartiennent à cette famille mais on peut également citer certaines méthodes relativement récentes telles *SPAN* [Zviran et al. 1990][149] et *LRAM* [Guarro 1987][61]. En fait ces méthodes sont totalement dépendantes d'un ensemble restreint de solutions techniques qu'elles essaient d'appliquer à un système. En quelque sorte, elles répondent à une question du type *"que pouvons-nous faire?"* plutôt que *"que devons-nous faire?"*. Elles n'offrent aucun recul par rapport aux spécificités techniques du système.

ii) Deuxième génération

La deuxième génération est empreinte d'une volonté de décomposition. Face à un système complexe, ces méthodes optent pour un partitionnement du système en sous-systèmes qu'elles sont mieux à même de gérer. Elles adoptent trois hypothèses :

(1) Les éléments qui composent la sécurité d'un système sont complexes et interconnectés ; il est nécessaire de partitionner le système pour l'analyser.
(2) L'ensemble des parades est spécifique au seul système sur lequel l'analyse est menée.
(3) La sécurité dépend également de la maintenance ; elle est sensible aux modifications du système.

En pratique, la différence essentielle avec la première génération réside dans le fait que les méthodes de deuxième génération doivent être capables d'ordonner les parades à mettre en œuvre en fonction de leur efficacité. Une algèbre est donc nécessaire. De plus, ces méthodes sont censées être réutilisées dans le cycle de maintenance des systèmes. L'analyse n'est plus basée sur un ensemble de parades existantes mais sur l'analyse du système et de ses besoins, en identifiant actifs et menaces. Elle est sans doute la plus classique et celle qui cadre le mieux avec les 6 phases identifiées préalablement. *CRAMM* [Farquhar 1991][49] et *RISKPAC* [Gardner 1989][55] appartiennent à cette catégorie.

iii) Troisième génération

La troisième et dernière génération adopte un haut niveau d'abstraction par rapport au système. L'analyse des risques est partie prenante de la phase de développement du système et est

menée sur le système dans sa globalité et non sur des partitionnements de ce dernier. Cinq hypothèses caractérisent cette génération :

(1) La solution de protection idéale ne peut être trouvée qu'à partir d'une compréhension du système global ; on ne peut donc, comme le fait la deuxième génération, restreindre son attention à des morceaux isolés du système.
(2) Seul un niveau d'abstraction suffisant permet d'envisager un système dans sa globalité.
(3) Les solutions dérivées d'une telle vision sont plus flexibles et plus faciles à adapter en cas de modification du système.
(4) Il n'existe pas de solution universelle ; chaque système requiert une solution particulière.
(5) L'utilité de la notion de rapport coût/bénéfice est inversement proportionnelle au niveau d'abstraction utilisé. En effet, il est fort difficile de quantifier les coûts et bénéfices induits par les risques et les parades quand n'est disponible qu'une abstraction du système réel.

Baskerville range deux méthodes dans cette catégorie, *SSADM-CRAMM* [GBT 1993][56] et "logical control design", sur lesquelles, en raison de leur relative nouveauté, il est difficile de donner un avis critique. D'autres auteurs envisagent également d'inclure l'analyse de risques dès les premières étapes des spécifications du système ([Bradenhorst et Eloff 1990][25], [Kristiansen 1991][85]). Cependant, il est clair qu'une telle vision s'écarte du cadre classique de l'analyse des risques et dépasse les frontières du domaine. En effet, nous pourrions ranger dans cette troisième génération toutes les méthodes d'évaluation classiques développées en sûreté de fonctionnement, telles que les graphes de Markov et les réseaux de Petri stochastiques, puisqu'elles offrent une méthode de modélisation abstraite et une capacité à générer des évaluations quantifiées sur ces représentations globales. Pour ne prendre qu'un exemple concret, le logiciel *SURF*-2 ([Béounes et al. 1993][17], [Metge 1994][102]) peut calculer, outre l'évolution du coût du système en fonction du temps, toutes les autres mesures classiques en sûreté de fonctionnement (fiabilité, disponibilité, maintenabilité, etc. [Arlat et al. 1989][9]). L'outil permet également de comparer différentes configurations au vu des résultats obtenus. Pourtant, nous ne pouvons dire que *SURF*-2 soit l'implémentation d'une méthode d'analyse de risques à part entière. Plutôt, c'est un outil utile à l'étude de la sûreté de fonctionnement d'un système, à utiliser dans le cadre d'une analyse plus générale des risques d'un système.

Ceci nous conforte dans l'idée que ces trois générations ne diffèrent que par le niveau d'abstraction qu'elles adoptent dans leur approche et le moment de leur utilisation. En pratique, une analyse complète doit utiliser des méthodes issues de chacune de ces "générations" en fonction des besoins rencontrés.

C. Centres d'intérêt des méthodes.

Toutes les méthodes ne s'intéressent pas aux risques portant sur les actifs du système. Nous pouvons considérer deux familles différentes :

i) **Méthodes centrées sur les actifs** : L'approche la plus classique consiste à identifier tous les actifs que contient le système. Il est également nécessaire d'évaluer les menaces vis-à-vis de ces actifs et les vulnérabilités du système à ces menaces. On définit ensuite le taux annuel d'occurrence d'une menace —sa *Fréquence*— ainsi que le coût induit par son activation —son *Impact*. Ceci permet de calculer la perte[1] annuelle estimée selon la formule $Perte = Impact \times Frequence$ [GBT 1993][56] que l'on évalue généralement menace par menace. Cette valeur est utilisée dans un rapport coût/bénéfice afin de décider de la mise en application, ou non, de parades connues. C'est le schéma classique, en six points, discuté plus haut.

ii) **Méthodes centrées sur les fonctions** : Pour ne pas avoir à identifier tous les actifs d'un système, d'autres méthodes préconisent de s'intéresser aux fonctions organisationnelles. Elles s'attachent à trouver les relations de dépendance entre ces fonctions et à distinguer les parties du système qui sont critiques pour son fonctionnement. Les menaces et vulnérabilités doivent également être recensées mais une vulnérabilité ne sera considérée que si elle implique une partie critique du système. Cette méthode est généralement plus rapide à mettre en œuvre que la précédente mais est beaucoup plus sujette à caution ; en effet, elle dépend très fortement de la vision qu'auront les décideurs des dépendances et parties critiques de la chaîne fonctionnelle [Warren 1983][145].

D. Mode d'évaluation des paramètres

Nous avons évoqué de nombreux paramètres jusqu'ici : actifs, menaces, occurrence de réalisation, coûts associés, etc. Afin de les évaluer, deux approches sont possibles.

i) **Approche quantitative**. L'approche la plus classique consiste à attribuer des valeurs financières aux actifs ainsi qu'aux coûts associés à la réalisation d'une menace. D'autre part, les menaces sont caractérisées par leur taux d'occurrence. Cette approche est aisée pour des systèmes matériels, par exemple, pour lesquels nous disposons de données statistiques. Elle pose cependant de nombreuses questions de fond pour les domaines de la sécurité où les menaces peuvent trouver leur origine dans le comportement malveillant d'un utilisateur du système. Aucun profil statistique n'est disponible dans ce cas.

ii) **Approche qualitative**. Ici, il n'y a plus de valeurs mais des estimations floues, du type "grave", "fréquent", "cher", etc. Ces valeurs sont généralement fournies à des programmes informatiques sophistiqués qui, à l'aide d'un formalisme spécifique, parviennent à en inférer un résultat quantitatif. Par exemple, les méthodes *RISKPAC*, *RANK-IT* et *COSSAC/VES*, décrites dans [Gardner 1989][55], appartiennent à cette famille.

Donner des estimations floues peut se révéler fort utile mais il n'en reste pas moins vrai que, derrière ces qualificatifs, se cachent des valeurs —ou du moins des ensembles de valeurs. Il est en effet vital que toutes les personnes participant à l'analyse d'un système disposent de points de repère communs pour les qualificatifs qu'elles emploient, sans quoi la pertinence et la

1. Le correspondant anglais de la perte, couramment utilisé dans la littérature, se nomme *ALE* pour *Annual Loss Expected* [Saari 1991][118].

cohérence des résultats risquent d'en souffrir. C'est pourquoi le terme "approche qualitative" nous semble mal approprié. Nous l'avons utilisé en raison de la fréquence de son apparition dans la littérature mais nous aurions préféré utiliser une autre expression comme "approche quantitative lâche" par opposition à "approche quantitative stricte", par exemple.

E. Modes opératoires

Enfin, nous distinguons cinq logiques différentes dans la façon d'aborder la mise en œuvre des méthodes :

i) **Listes de contrôle :** Partant des parades possibles, la nécessité de les mettre en place est examinée. Cette approche a été discutée précédemment (page 21).

ii) **Evaluation quantitative :** La méthode la plus classique consiste à quantifier le risque à partir des actifs, des menaces et des vulnérabilités comme nous l'avons expliqué ci-devant. La démarche à suivre, dans ce cadre, est fastidieuse. La logique d'analyse part des données connues du système et se prolonge par une évaluation systématique.

iii) **Questionnaires** : Certaines méthodes étudient les risques par le biais d'entretiens avec des utilisateurs et responsables du système. Le résultat de ces questionnaires préliminaires peut déterminer la suite des travaux à mener. Les méthodes que nous avons qualifiées de "*quantitatives*" et "*centrées sur les fonctions*" procèdent de cette façon.

iv) **Scénarios** : Dès lors qu'un système devient complexe, il devient nécessaire de pouvoir définir non plus des couples vulnérabilité-impact mais des scénarios d'attaques. Ceux-ci envisagent la possibilité pour un individu de combiner plusieurs vulnérabilités, *a priori* anodines, pour menacer certains actifs. Dans ce cas de figure, l'estimation doit tenir compte non seulement de la probabilité de pouvoir utiliser avec succès chaque vulnérabilité indépendamment mais aussi de la probabilité de parvenir à exploiter une succession d'attaques. En effet, les attaques déjà réussies peuvent influencer la probabilité de continuer l'intrusion avec succès. D'une part, l'attaquant peut avoir sa tâche facilitée par l'expérience des intrusions déjà menées mais, d'autre part, elles peuvent avoir alerté les responsables du système et obliger l'attaquant à dissimuler sa présence. Les modèle *ANSSR* [Bodeau et al. 1990][24] et *MELISA* [GBT 1993][56] incluent cette approche dans les solutions qu'ils proposent. Le modèle développé par Fitch et Hoffman [Fitch III et Hoffman 1993][52] offre, lui aussi, des fonctionnalités analogues.

v) **La ligne commune :** L'histoire n'est que recommencement. C'est ce que nous pouvons nous dire en considérant cette dernière famille. Déçus par la lourdeur ou l'imprécision des méthodes développées au long de ces vingt dernières années, certains auteurs proposent une approche pragmatique. Forts de leur expérience, ils préconisent d'appliquer un ensemble de recettes bien connues qui ont fait leurs preuves par le passé dans des sociétés bien protégées. Faisant fi de toute réelle évaluation, ils arguent du fait que, *in fine*, l'ensemble des méthodes à mettre en œuvre est bien connu. Il suffit donc de se contenter de réutiliser l'expertise acquise par le développement d'autres systèmes et

calquer les parades à mettre en œuvre sur celles d'autres systèmes qui fonctionnent bien [Saari 1991][118].

I.3.6 Discussion

Des pages qui précèdent, il ressort que les méthodes d'analyse des risques sont diverses et variées. Cette prolifération est, notamment, le résultat des critiques nombreuses qui leur ont été portées : pour pallier un certain nombre d'entre elles, de nouvelles méthodes furent créées, donnant elles-mêmes naissance à de nouvelles critiques, et ainsi de suite. Ce qui suit est un essai de synthèse des reproches principaux faits à l'analyse de risques. Outre les nombreux documents qui présentent telle ou telle méthode, nous avons basé notre analyse sur [Hoffman 1986][73], [Parker 1986][111], [Anderson 1991][4], [Baskerville 1991][13] et [Parker 1986][111].

A. Mise en œuvre

Le choix d'une méthode est un problème crucial. Il n'existe pas de bancs de test qui permettent de comparer les candidates et de choisir la plus adéquate dans un environnement donné. Ce n'est qu'une fois l'analyse réalisée que l'on peut savoir si elle était utile, réalisée au meilleur moment possible et avec le niveau d'abstraction adéquat. Il n'est cependant pas possible de tirer de conclusion définitive puisque deux méthodes différentes peuvent, en raison de leur mode opératoire propre, délivrer des résultats différents.

Le problème est exacerbé par l'importance que peuvent avoir certains choix de l'évaluateur sur les résultats finaux. Son expérience peut, en effet, se révéler cruciale dans l'estimation de certains paramètres, voire même dans la manière d'appréhender le système à étudier.

B. Obtention des données

La collecte des données est problématique : soit elle se veut exhaustive et la méthode est coûteuse en temps et en effort, soit elle utilise quelque heuristique de sélection et s'en remet alors à la maîtrise de l'évaluateur.

Cependant, si la méthode est rigoureuse dans sa façon de traiter les données, elle ne peut considérer les données sociologiques ou politiques (dans l'évaluation des conséquences, par exemple) et il reviendra à l'évaluateur, *in fine*, d'interpréter les résultats et d'aider le preneur de décision en lui faisant sentir l'influence de ces paramètres qui n'ont pas pu être pris en compte précédemment.

Dans le cas précis de l'analyse des risques des systèmes d'informations, le problème est patent. En effet, il y est extrêmement difficile de cerner les conséquences financières de l'utilisation d'une vulnérabilité. A titre d'exemple, Parker [Parker 1986][111] relève vingt-cinq types de pertes induits par un acte malveillant sur un système informatique. De plus, les taux d'occurrence sont mal connus ou même inconnus puisque nous ne disposons pas de bases de données statistiques des profils d'assaillants. Dans ce contexte, cela pousse certains auteurs à comparer

les méthodes d'analyse des risques à un exercice stérile de devinettes sans réponse. Sans aller jusque là, nous reconnaissons toutefois le rôle primordial joué dans le résultat final par les estimations de l'évaluateur.

C. Modélisation du système

Nous avons indiqué que la plupart des méthodes ne se restreignaient pas à un champ d'investigation particulier. Au contraire, elles prônent l'adoption d'une démarche commune, à adapter aux spécificités des différents environnements de travail. Hormis le travail d'estimation de paramètres évoqué plus haut, l'évaluateur a donc également pour tâche de définir la façon dont la méthode va être appliquée dans chaque cas précis. Ceci requiert une connaissance approfondie de la méthode en question, de ses hypothèses sous-jacentes et de ses mécanismes. Ceci explique le fait qu'aucune méthode n'est présentée comme accessible à des néophytes en matière d'analyse de risques.

Dès lors que l'assistance d'une personne externe au système est requise, il faut pouvoir communiquer avec elle. Cette tâche est facilitée si les communautés amenées à se côtoyer ont des habitudes communes. Malheureusement, dès 1986, le manque de relations, de vocabulaires communs et de modèles partagés entre les communautés de l'analyse de risque et de la sécurité informatique a été mis en exergue. Dans un rapport interne [Hoffman 1986][73], Hoffman reprend les conclusions de trois journées de travail auxquelles ont participé vingt-quatre experts des deux disciplines. Il en ressort que "*si un modèle général de risques pouvait être développé qui puisse être utilisé par les deux communautés de l'analyse de risque et de la sécurité informatique, cela aiderait considérablement les deux communautés dans les domaines du test, de l'évaluation des méthodologies et de leur analyse de complétude*".

Huit ans après, nous ne disposons toujours pas d'un tel modèle. Ce n'est cependant pas faute de nouvelles approches puisqu'il y a pléthore en la matière. Le problème vient de ce que toutes participent de la même démarche méthodologique [Anderson 1991][4]. Chacune s'inscrit dans la lignée d'une, ou plusieurs précédentes dont elle prétend se distinguer par la résolution de critiques qui leur avaient été faites. De plus, la volonté d'approche globale reste très marquée et est un frein à l'utilisation des modèles formels de sécurité bien connus des spécialistes de la sécurité informatique.

En vérité, peut-être faut il attribuer au fossé entre ces deux communautés une cause plus philosophique. C'est en tous cas ce qu'explique Baskerville dans [Baskerville 1991][13], où il montre que l'analyse des risques n'est pas une science au sens positiviste du terme bien qu'elle en soit une si on la considère dans un cadre "*interprétatif*". Ce cadre de travail, classique en sociologie, autorise, selon lui, l'évaluateur à agir comme un filtre sur les données qu'il doit gérer. Son expertise et son savoir-faire sont des éléments indissociables de la méthode qu'il met en œuvre.

D. La méthode "idéale"

Nonobstant cette analyse de Baskerville, il n'en reste pas moins vrai que le besoin d'une "bonne" méthode apte à mettre en œuvre des moyens de prévision des fautes reste réel. Dès lors que nous désirons rester dans un cadre positiviste, nous pouvons identifier les qualités fondamentales que doit avoir une méthode idéale en nous inspirant des recommandations que font les *ITSEM* à ce propos [ITSEM 1993][78]. En particulier, ils stipulent qu'une méthode d'évaluation doit être :

- Répétable : toute autre chose restant égale, une évaluation réalisée plusieurs fois par le même évaluateur sur le même système doit livrer les mêmes résultats.
- Reproductible : toute autre chose restant égale, une évaluation réalisée par différents évaluateurs sur un même système doit livrer les mêmes résultats.
- Impartiale : l'évaluation ne peut favoriser un type de résultat particulier.
- Objective : les résultats ne peuvent être basés que sur des faits non influencés par les opinions ou sentiments des évaluateurs.

Enfin, au vu de ce qui précède nous stipulons également qu'une méthode idéale doit :

- Permettre un lien aisé avec les modèles formels de sécurité connus.
- Permettre une acquisition des données qui soit aussi économique que possible.
- Offrir un moyen de s'assurer de la plausibilité des résultats qu'elle donne.

I.4 Conclusions

Dès lors que le système comprend des fautes résiduelles, il est nécessaire de connaître leur influence sur les objectifs de sécurité. Pour cela, il faut mettre en œuvre des méthodes de prévision de fautes [Dacier 1994][41]. Les méthodes d'analyse des risques ont acquis une expertise de ce type d'approche qui est difficilement transposable à la sécurité informatique. Issues de cadres d'application aux contraintes différentes, elles tentent d'évoluer en réponse aux critiques qui leur sont faites. Soucieux d'adapter des méthodologies existantes et de justifier leur adéquation à un nouveau cadre, les concepteurs de méthodes se sont peu ou prou intéressés à modéliser les systèmes qu'ils étudiaient. Comme le note très justement Anderson, dans [Anderson 1991][4], l'état de l'art témoigne plus d'une démarche méthodologique que d'un réel travail de modélisation.

Notre but étant de réaliser une évaluation quantitative de la sécurité, nous devons disposer d'un modèle mathématique permettant une telle évaluation. Ceci n'est possible que si nous disposons, au préalable, d'un modèle de représentation du système et de ses règles de sécurité. Les méthodes d'analyse des risques n'offrent pas de tels modèles et c'est la raison pour laquelle nous nous intéressons dans le chapitre qui suit aux modèles formels développés pour la sécurité.

CHAPITRE II *Modèles formels pour la sécurité*

Préambule

Dans le chapitre précédent nous avons montré que :

(1) Un objectif de zéro-faute est illusoire pour des systèmes réels.
(2) L'évaluation de la sécurité requiert la mise en œuvre, notamment, de méthodes de prévision des fautes.
(3) La démarche adoptée par les méthodes d'analyse des risques pèche par le peu de modélisation des systèmes étudiés.

Prolongeant cette dernière remarque, nous considérons dans ce chapitre l'état de l'art en matière de modélisation formelle de la sécurité. Après un bref rappel des différentes familles de modèles, nous choisissons, pour sa facilité d'utilisation, celle basée sur les matrices de contrôle d'accès. Nous examinons les solutions apportées par ces modèles et leurs hypothèses sous-jacentes. Nous considérons plus particulièrement trois modèles, le modèle *HRU*, Take-Grant (*TG*) et enfin celui dit de matrice de contrôle d'accès typée (*TAM*).

La description du modèle *HRU* nous permet de préciser la notion de problème de protection dont la résolution se révèle être non-décidable dans le cas général. Il en résulte la nécessité de trouver un compromis entre puissance d'expression et complexité de résolution du problème de protection.

Le modèle Take-Grant (*TG*) est issu de cette réflexion. Sa simplicité nous permet d'expliquer l'hypothèse de *pire cas* adoptée par la plupart des modèles sur le comportement des utilisateurs du système étudié. Nous justifions la nécessité de nous en affranchir afin d'obtenir une modélisation réaliste et proposons une solution basée sur les réseaux de Petri.

Enfin, nous détaillons le modèle *TAM* dont la richesse d'expression est supérieure à celle du modèle *TG* tout en conservant des propriétés intéressantes au niveau de la complexité de l'algorithme de résolution. Les systèmes réels se caractérisent par des schémas d'autorisation dont les règles induisent la cession de grands ensembles de droits sur de nombreux objets. Nous étudions l'efficacité de *TAM* pour gérer de tels schémas et aboutissons à la définition d'un nouveau modèle, appelé le ***graphe des privilèges***. Il servira de base de travail lorsque nous nous intéresserons, dans le chapitre IV, au problème de l'évaluation quantitative proprement dite.

II.1 Les familles de modèles

L'étude des modèles formels de sécurité a donné lieu à un grand nombre de travaux. Afin d'en offrir une vision aussi synthétique que possible, nous présentons une classification en quatre familles, inspirée de [Landwehr 1981][89] et que nous avons réactualisée :

- Le modèle de machine à états ;
- Les modèles basés sur un treillis ;
- Les modèles spécifiques ;
- Les modèles basés sur les matrices d'accès.

Il faut noter que cette classification ne définit pas quatre ensembles disjoints de modèles. En effet, certains modèles peuvent appartenir à plusieurs familles à la fois. Cependant, cette façon de les répertorier permet d'identifier quatre points de vue spécifiques des modèles formels de sécurité.

II.1.1 Modèle de machine à états

Le modèle de machine à états est à la fois le plus général et le moins pratique. Il représente l'évolution dans le temps du système informatique comme un ensemble d'états ; il existe une fonction de transition qui, à partir d'un état courant et d'une valeur d'entrée, détermine le nouvel état du système. En fait, nous pouvons considérer que cette famille englobe toutes les autres ; chacune se distingue par la façon de définir un état, la fonction de transition ou encore l'ensemble des états qui violent la sécurité [Landwehr 1981][89].

II.1.2 Modèles basés sur un treillis

A. Notion de treillis

Dans les milieux militaires existe un système de classification de l'information et des utilisateurs par niveaux. Ces **niveaux** représentent des classes de sécurité prédéfinies et ordonnées : non-classifié, à diffusion restreinte, confidentiel, secret, très secret. A chaque utilisation et à chaque objet correspond un niveau de sécurité précis. Il existe une relation d'ordre totale sur ces niveaux.

En sus de ces niveaux, existe la notion de **compartiment** qui caractérise le type de l'information : par exemple, les informations en rapport avec les armes nucléaires appartiendront au compartiment "nucléaire". Ces compartiments définissent des ensembles d'éléments (sujets ou objets) et la relation d'inclusion définit une relation d'ordre partiel sur les compartiments : un compartiment est plus grand ou égal à un autre s'il inclut les éléments de ce dernier.

La **classification** d'un élément comprend son *niveau de sécurité* et le *compartiment* auquel il appartient. Les classifications peuvent être ordonnées selon un ordre partiel de la façon

suivante : pour tout niveau de sécurité a et b et pour tout compartiment c et d, nous avons $(a,c) \geq (b,d) \Leftrightarrow a \geq b \wedge c \supseteq d$. Si l'élément est un utilisateur, le terme d'**habilitation** est utilisé.

Un flot d'information d'un élément x vers un élément y n'est possible, dans ces modèles, que si la classification de y est supérieure ou égale à celle de x. La représentation de tous les flots d'information possibles correspond à un treillis dont la borne supérieure correspond à la classification *(très secret, union de tous les compartiments)* et la borne inférieure à *(non-classifié, aucun compartiment)*.

La notion de treillis est très générale. Elle est principalement utilisée dans les modèles de contrôle d'accès multi-niveaux et de contrôle de flot que nous décrivons ci-dessous.

B. Contrôles d'accès multi-niveaux

Le modèle de Bell-La Padula [Bell et La Padula 1976][15] utilise la notion de treillis afin d'imposer des conditions nécessaires, mais non suffisantes, sur les autorisations d'accès à certaines informations :

- A chaque sujet s_i correspond une habilitation $h(s_i)$;
- A chaque objet o_j correspond une classification $c(o_j)$;
- *Propriété simple* : aucun sujet n'a accès en lecture à un objet dont le niveau de classification est supérieur à son niveau d'habilitation.
- *Propriété étoile :* les flux d'information d'un niveau de classification donné vers un niveau de classification inférieur sont interdits.

L'intérêt du modèle de Bell-La Padula réside dans le fait qu'il offre une preuve rigoureuse que les règles de contrôle d'accès qu'il propose (c'est-à-dire *le schéma d'autorisation*) sont telles que ces propriétés (c'est-à-dire les *objectifs de protection*) sont toujours vérifiées.

Le principal inconvénient de ce modèle est lié à la dégradation par surclassification des informations. En effet, au cours de sa vie, le niveau de classification d'une information ne peut que croître : si une information non-classifiée est utilisée par un sujet habilité au secret, tout objet modifié par ce sujet avec cette information sera classifié secret. Petit à petit, les niveaux de classification des informations croissent de façon systématique, et il faut les "déclassifier", manuellement par un officier de sécurité ou par un processus de confiance (n'obéissant pas aux règles du modèle).

Le modèle de Biba [Biba 1977][18] est un modèle dual de celui de Bell-La Padula dans la mesure où il définit des propriétés qui visent à préserver l'intégrité des données, alors que le modèle de Bell-La Padula se focalise sur leur confidentialité.

C. Modèles de contrôle de flot

Dans [Lampson 1973][88], l'auteur explique pourquoi contrôler l'accès aux informations ne suffit pas à empêcher toute forme de propagation interdite. En effet, il existe différentes

techniques pour divulguer à un tiers le contenu d'un document sans qu'il lui soit pour autant possible d'accéder au document en question. Dans ce cas, le flot d'informations utilise ce qu'il est courant d'appelé un "*canal caché*".

L'accès à des ressources partagées entre différents utilisateurs (réseau, *CPU* , imprimantes, bases de données etc.) est notamment la source de tels canaux cachés. En modulant son utilisation d'une ressource partagée, un individu peut transmettre une information, sous forme codée, à un observateur. De nombreux travaux ont étudié la meilleure façon d'assurer un accès concourant à des ressources tout en éliminant les canaux cachés basés sur ce type de synchronisation entre comparses ([Sandhu et al. 1992][122], [McDermott et Jajodia 1993][100]).

D'autre part, d'autres modèles ont été développés qui utilisent la structure du treillis mais optent pour une démarche plus abstraite ne s'intéressant plus aux règles de contrôle d'accès des différents utilisateurs. Les modèles de contrôle de flot servent à vérifier des propriétés générales sur les flots d'information *entre sujets*. Ces modèles portent leur attention sur les différentes actions qui permettent de transférer de l'information plutôt qu'aux transferts des privilèges sur les objets qui contiennent l'information. Typiquement, les caractéristiques des flots engendrés par l'exécution de chaque commande d'un langage de programmation peuvent être définies et des preuves développées sur les flots causés par l'exécution d'un programme particulier.

A titre d'exemple, nous pouvons citer le travail réalisé par Bieber et Cuppens [Bieber et Cuppens 1992][19] dans lequel les auteurs spécifient les propriétés de sécurité attendues d'un système à l'aide d'un opérateur modal $\mathcal{R}_B$ qui dénote la *permission de savoir* une information.

II.1.3 Modèles spécifiques

Certaines activités professionnelles sont régies par des politiques de sécurité qui leur sont spécifiques. La transposition de ces pratiques dans le cadre des systèmes informatiques a nécessité la création de nouveaux modèles. Le modèle de Clark et Wilson, ainsi que celui de la muraille de Chine ressortent de cette mouvance.

Le modèle de Clark et Wilson [Clark et Wilson 1987][33] s'intéresse principalement à l'intégrité des données. Se basant sur les pratiques en vigueur dans les systèmes commerciaux, il introduit deux nouvelles notions : les transactions bien formées et la séparation des privilèges. La première stipule que les utilisateurs ne peuvent manipuler les informations que par le biais de méthodes précises qui préservent l'intégrité des données. La seconde préconise l'intervention de plusieurs personnes différentes pour mener à bien une tâche donnée. C'est là une forme de tolérance aux fautes humaines aussi bien intentionnelles qu'accidentelles (fraudes et erreurs).

Le modèle de la muraille de Chine [Brewer et Nash 1989][27] étudie les problèmes de confidentialité liés à l'intervention d'une même personne dans plusieurs milieux aux intérêts

concurrents. Le modèle stipule que les droits que détient un intervenant à un instant donné conditionnent ceux qu'il pourra avoir par la suite. En pratique, le modèle définit des ensembles de groupes d'informations et interdit aux utilisateurs d'avoir accès à plus d'un groupe d'informations d'un même ensemble. Par exemple, supposons que les sociétés *A* et *B* soient concurrentes. Si un consultant a accès aux informations de la société *A*, il ne pourra, par la suite, intervenir auprès de la société *B*. Cette dimension temporelle est nouvelle et justifie l'introduction d'un nouveau modèle. La muraille de Chine est une image des barrières qui se dressent au devant des utilisateurs au fur et à mesure des actions qu'ils mènent. Notons que cette politique correspond à une réglementation imposée aux agents de change britanniques car leur travail les amène à travailler pour différentes sociétés qui peuvent être en situation de conflit d'intérêts.

II.1.4 Modèles basés sur les matrices de contrôle d'accès

La notion de matrice de contrôle d'accès, introduite par Lampson [Lampson 1971][87] puis précisée par Denning et Graham ([Denning 1971][43], [Graham et Denning 1972][57]) est un modèle simple et général. Elle manipule trois concepts fondamentaux : les sujets, les objets et les droits. Les sujets représentent les agents actifs du système, les objets sont les éléments que les sujets peuvent manipuler et les droits conditionnent les actions qu'un sujet peut effectuer sur un objet. Il est important de noter que tout sujet est également un objet. Le nombre de lignes (respectivement, de colonnes) de la matrice est égal au nombre de sujets (respectivement, d'objets). Les éléments se situant dans la ligne *L*, colonne *C*, correspondent aux droits que possède le sujet *L* sur l'objet *C*. Cette matrice caractérise l'**état de protection** du système.

Cet état peut évoluer au fil du temps du fait de l'activité des différents sujets. La fonction de transition est définie sous forme de règles qui constituent le **schéma d'autorisation**. Dans le cadre de ces modèles, prouver que le schéma est correct vis-à-vis du problème de protection revient à définir les états de protection non acceptables et vérifier qu'à partir d'un état de protection initial sûr, on ne peut atteindre un état non sûr (c'est-à-dire ne respectant pas les contraintes de sécurité) par l'application des règles du schéma d'autorisation.

Ces modèles ont rencontré un vif succès en raison de leur simplicité et de leur souplesse d'utilisation. Nous détaillons dans ce qui suit leurs principaux résultats.

II.2 Le modèle *HRU*

II.2.1 Introduction

Harrison, Ruzzo et Ullman furent les premiers à proposer un cadre rigoureux pour définir les notions énoncées ci-dessus. Leur modèle, dénommé *HRU* ([Harrison et Ruzzo 1975][64],

[Harrison et Ruzzo 1976][65]), utilise une matrice d'accès classique, mais précise les commandes qui peuvent lui être appliquées. Les seules actions possibles sont données dans la table II.1.

1) **enter** r **into** $[X_s, X_o]$;	2) **create subject** X_s ;	3) **destroy subject** X_s ;
4) **delete** r **from** $[X_s, X_o]$;	5) **create object** X_o ;	6) **destroy object** X_o

Table II.1 Les opérations élémentaires du modèle *HRU*

Le format des commandes est donné dans la table II.2. Tous les sujets et objets apparaissant dans la partie conditionnelle de la commande doivent faire partie des paramètres de la commande. Plus formellement, nous avons : X_1, X_2, ..., X_k sont des paramètres de la commande, r_1, ..., r_m représentent des droits ; $s1$, $s2$, ..., sm de même que $o1$, $o2$, ..., om représentent des entiers prenant leurs valeurs entre 1 et k, les opérations op_i représentent une des six opérations élémentaires définies dans la table II.1

command $foo(X_1, X_2, \ldots, X_k)$
  if $r_1 \in [X_{s1}, X_{o1}]$ **and** $r_2 \in [X_{s2}, X_{o2}]$ **and ... and** $r_m \in [X_{sm}, X_{om}]$
    then $op_1; op_2; \ldots; op_n$
  end

Table II.2 Les commandes du modèle *HRU*

Nous appellerons **système de protection** un ensemble fini, $\mathcal{R}$, de droits et un ensemble fini, C, de commandes.

Les règles du modèle *HRU* permettent d'exprimer une très grande variété de schémas d'autorisation, mais cette richesse a un prix. En effet, le résultat fondamental délivré par le modèle *HRU*, qui va conditionner tous les travaux ultérieurs, est la preuve de la non-décidabilité de la *sécurité* d'un système de protection dans le cas général : étant donné une matrice d'accès initiale et un ensemble de commandes, les auteurs ont prouvé qu'il est impossible de savoir si aucune séquence d'applications de ces commandes n'aura pour conséquence de mettre un droit particulier à un endroit précis de la matrice où il ne se trouvait pas initialement.

Bien entendu, la non-existence d'un algorithme général n'implique pas la non-existence d'algorithmes particuliers pour des schémas d'autorisation précis. Encore faut-il définir ces schémas restreints et leurs algorithmes associés. C'est ce que firent les travaux qui suivirent en tentant de restreindre la richesse d'expression des schémas d'autorisation pour trouver une solution au *problème de protection* (tel qu'il a été défini dans le chapitre I, section I.1.4, page 8).

II.3 Le modèle *TG*

II.3.1 Introduction

Le modèle "*Take-Grant*" (*TG*, [Lipton et Snyder 1977][95], [Snyder 1977][126], [Landwehr 1981][89], [Snyder 1981a][127]) utilise un graphe et des règles de modification de ce graphe pour modéliser le contrôle d'accès sur des données partagées. Les nœuds du graphe sont de deux types : sujets ou objets. Les arcs sont étiquetés. Ces étiquettes sont des sous-ensembles d'un ensemble fini $\mathcal{R}$ de droits. Il existe quatre règles de réécriture du graphe. L'ensemble de ces quatre règles constitue le schéma d'autorisation. Mis à part cet aspect graphique des choses, c'est bel et bien un modèle basé sur une matrice d'accès. En effet, dans ce modèle, le graphe représente l'état de protection du système, jouant le rôle de la matrice, et les règles de réécriture correspondent au schéma d'autorisation, c'est-à-dire aux commandes. Il est clair que le modèle *TG* est donc une variante du modèle *HRU*, obtenu par restriction de l'expressivité des commandes de définition du schéma d'autorisation.

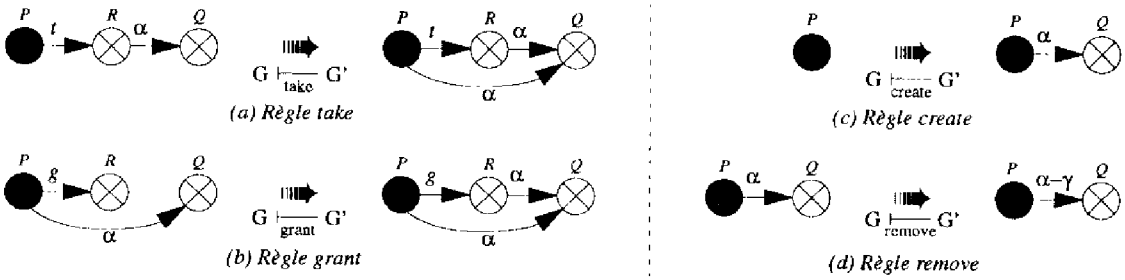


Figure II.1 Les règles de réécriture du modèle *Take-Grant*

Les quatre règles de réécriture se nomment respectivement: *take*, *grant*, *create* et *remove*. Nous en donnons une représentation graphique[1] dans la figure II.1 L'interprétation en est immédiate. Ainsi, par exemple, la règle *take* stipule qu'un arc étiqueté *t* entre un sujet *P* et un sujet (ou objet) *R* indique que *P* peut prendre tous les droits que *R* détient sur d'autres nœuds. Quant à la règle *create*, elle stipule que tout sujet peut créer de nouveaux sujets ou objets et s'accorder les droits α sur ces nouveaux éléments (en particulier, α peut représenter les droits *t* et/ou *g*).

On peut exprimer cette règle *take* de façon similaire avec la syntaxe du modèle *HRU*. C'est ce qui est fait dans la table II.3. Bien entendu, il faut définir une commande *take_*r_j pour tout droit r_j défini dans l'ensemble $\mathcal{R}$de droits du système de protection considéré.

```
command take_rj(P,R,Q)
    if take ∈ [P,R] and rj ∈ [R,Q]
        then enter rj in [P,Q]
    end
```

Table II.3 Expression de la règle *take* à l'aide de la syntaxe *HRU*

1. Dans la représentation habituelle du modèle *TG*, les cercles pleins représentent des sujets, les cercles vides des objets et les cercles barrés d'une croix des sujets ou des objets.

Il ne faut pas se laisser tromper par l'apparente simplicité de ces règles. Une fois combinées, elles peuvent mener à des résultats surprenants. Par exemple, considérons le graphe de protection représenté dans la figure II.2. Il contient deux sujets, *P* et *R*, et un objet *O*. Dans l'état de protection initial, *R* possède le droit α sur *O* et le droit *t* sur *P*. Considérons un objectif de protection stipulant que le système est déclaré non-sûr si *P* parvient à acquérir le droit α sur *O*. A première vue, l'absence d'arc entre *P* et *O* inclinerait à penser que le système est sûr.

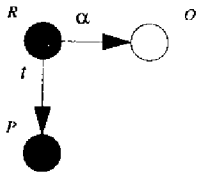


Figure II.2 Un exemple simple d'état de protection

Cependant, la séquence d'applications des règles décrites dans la figure II.3 indique qu'il ne l'est pas. Comme nous allons le voir, vérifier qu'un schéma d'autorisation satisfait des objectifs de protection précis peut se révéler assez complexe dans le cas du modèle *TG*.

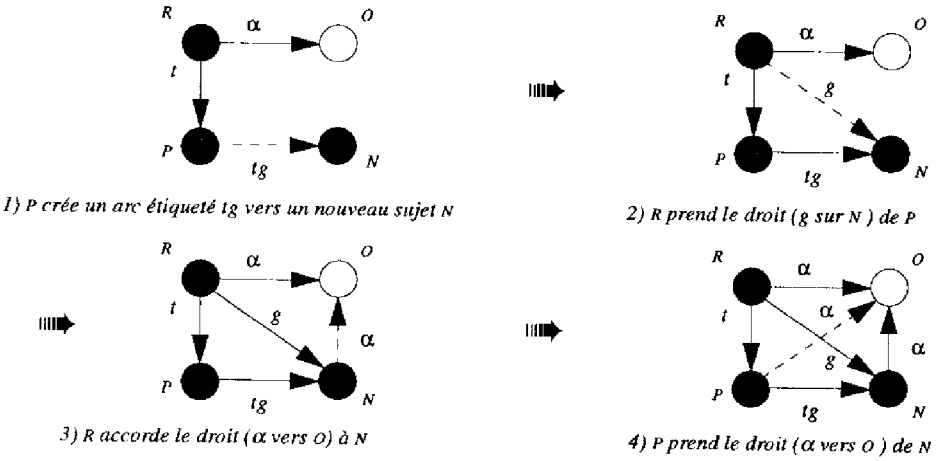


Figure II.3 Exemple d'application des règles de réécriture

II.3.2 Le prédicat "can"

Jones et al. ont étudié le *problème de protection* dans le cadre du modèle *TG* [Jones et al. 1976][82]. Ils définissent le prédicat "**can**" de la façon suivante : "*P can α Q*" est vrai si et seulement si il existe une séquence de graphes $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_n$ telle que *P* ait le droit α sur *Q* dans le graphe $\mathcal{G}_n$. L'intérêt de leur travail réside dans le fait qu'ils définissent les conditions nécessaires et suffisantes pour que le prédicat soit satisfait. Ce faisant, ils établissent également l'existence d'une solution algorithmique de complexité linéaire permettant d'établir si le prédicat est vérifié.

Quoique très important, ce résultat n'est pas d'un grand secours car il repose sur des hypothèses de travail fort peu réalistes. En effet, pour en revenir à notre exemple précédent, s'il est vrai que *P* peut effectivement parvenir à acquérir le droit α sur *O*, il ne peut le faire que

si R a collaboré avec lui (aux étapes 2 et 3 de la figure II.3). En réalité, il est difficile d'imaginer que tous les sujets, sans exception, vont collaborer afin de mettre la sécurité en péril. Une telle hypothèse est donc une hypothèse de *pire cas* sur le comportement des utilisateurs du système.

II.3.3 Le prédicat "can steal"

Certains auteurs ont entrepris d'étudier d'autres cas de figures où certains sujets ne collaboreraient pas. Snyder, par exemple, imagine une situation où toute personne en possession du droit α dans le graphe de protection initial ne va pas exécuter d'action qui aurait pour effet de diffuser ce droit [Snyder 1981][128]. Il définit un nouveau prédicat, "**can steal**" : "*P can steal* α *Q*" est vrai si et seulement si P peut acquérir le droit α vers Q sans l'aide d'aucun sujet déjà en possession du droit α sur Q dans le graphe de protection initial. Il définit également les conditions nécessaires et suffisantes à satisfaire pour que le prédicat soit vérifié. Certains utilisateurs ne collaborent donc pas, mais rien ne change en ce qui concerne tous ceux qui ne possèdent pas le droit α sur Q dans le graphe initial.

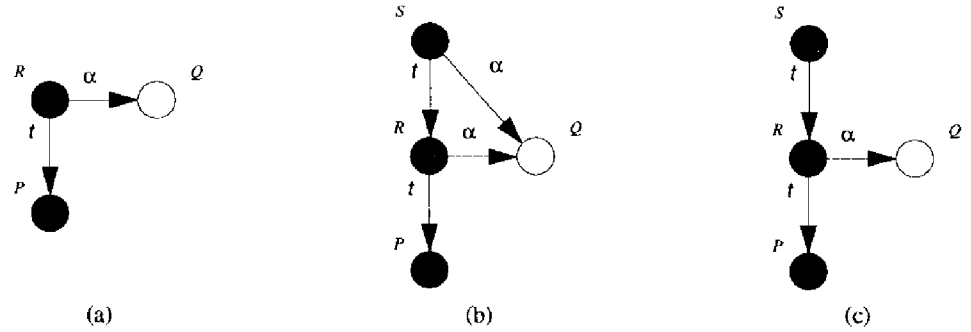


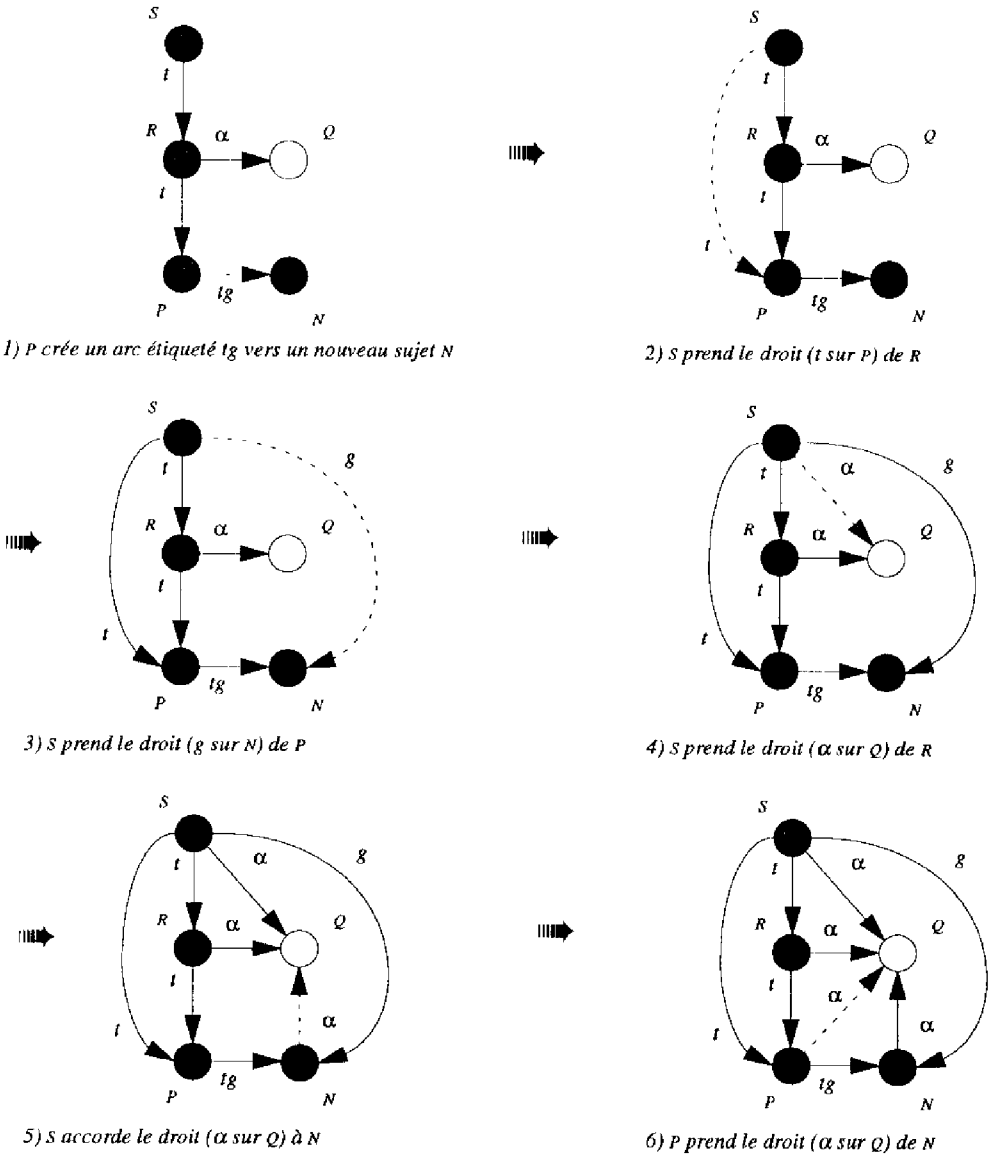
Figure II.4 (a) P can not steal α O; (b) P can not steal α O; (c) P can steal α O

Trois exemples de graphe de protection sont donnés à la figure II.4 afin d'illustrer ce prédicat. Le cas (a) est le même que celui considéré précédemment (figure II.2, page 36). Ici, le prédicat "*P can steal* α *Q*" n'est pas vérifié puisque R détient le droit α sur Q dans le graphe de protection initial et ne peut donc plus collaborer avec P, comme il le faisait aux étapes 2 et 3 de la figure II.3.

Le cas (b) livre le même résultat puisqu'il ne contient qu'un utilisateur en plus, S, et que ce dernier détient également le droit α sur Q dans le graphe de protection initial. Il ne peut donc initier de règles et, ce faisant, n'est d'aucune utilité pour P. Dans le cas (c), en revanche, S n'a pas le droit α sur Q et peut donc collaborer avec P. Pour illustrer notre propos, nous indiquons dans la figure II.5 comment P et S peuvent collaborer afin d'arriver à un état où P détient le droit convoité sur Q.

II.3.4 Extensions du modèle

Comme on le voit, la définition de ce prédicat est un premier pas vers un relâchement de l'hypothèse de pire cas émise sur le comportement des utilisateurs du système.



1) P crée un arc étiqueté tg vers un nouveau sujet N

2) S prend le droit (t sur P) de R

3) S prend le droit (g sur N) de P

4) S prend le droit (α sur Q) de R

5) S accorde le droit (α sur Q) à N

6) P prend le droit (α sur Q) de N

Figure II.5 Grâce à *s*, *P* parvient à acquérir le droit α sur *Q*

Malheureusement, l'avance est faible. Il serait plus utile de disposer d'un prédicat du type "*P can α Q avec l'aide du seul ensemble C de collaborateurs*" ; ce prédicat est vérifié si *P* peut acquérir le droit α avec la seule aide des sujets définis dans un ensemble *C*. Malheureusement, un tel prédicat n'existe pas.

Pourtant, ce n'est pas faute de l'intérêt porté à ce modèle au fil des années. Ainsi, certains ont-ils défini de nouveaux prédicats caractérisant différentes propriétés du graphe [Snyder 1977][126], d'autres l'ont enrichi de nouvelles règles [Bishop et Snyder 1979][20] afin de l'utiliser dans le cadre du problème du confinement [Lampson 1973][88] et étudier des systèmes de protection hiérarchiques ([Wu 1981][147], [Bishop 1981][21]), d'autres enfin ont exprimé le modèle sous forme de grammaires de réécriture ([Biskup 1984][23],

[von Solms et de Villiers 1988][142]) afin d'introduire dans le modèle *TG* le concept de niveaux de sécurité décrits précédemment [von Solms et Edwards 1989][143]. Tous ces travaux montrent l'intérêt potentiel de ce modèle dans différents contextes d'application. Ils font d'autant plus ressentir le besoin d'une méthode permettant de relâcher l'hypothèse de pire-cas.

Nous avons récemment proposé une solution à ce problème [Dacier 1993][39]. Elle offre une totale liberté dans la définition de l'ensemble des collaborateurs et définit un algorithme de résolution dont la complexité algorithmique reste linéaire. Elle possède les trois caractéristiques suivantes :

- Un réseau de Petri est utilisé pour représenter l'état de protection initial et modéliser la logique des règles de réécriture.
- L'ensemble des collaborateurs est défini par le marquage initial du réseau ainsi construit.
- La résolution du problème de protection consiste à effectuer une séquence de tir maximale de ce réseau de Petri.

Nous en expliquons les grandes lignes dans la section qui suit et reportons à l'Annexe A les développements et justifications mathématiques.

II.4 Levée de l'hypothèse de pire-cas pour le modèle *TG*

II.4.1 Introduction

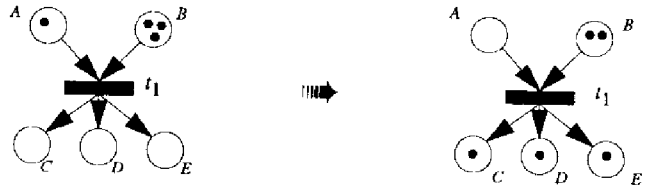
Un **réseau de Petri** [Peterson 1981][112] est un quadruplet P, T, e, s où :

- P et T sont des ensembles finis non vides disjoints de places et de transitions ;
- e est une application de P dans T qui, à chaque place de P, fait correspondre un sous-ensemble de T appelé *transitions suivantes* et définit un sous-ensemble E de $P \times T$;
- s est une application de T dans P qui, à chaque transition de T, fait correspondre un sous-ensemble de P appelé *places de sorties* et définit un sous-ensemble S de $T \times P$.

On représente généralement un réseau de Petri sous la forme d'un graphe orienté biparti. Les nœuds du graphe sont les places (matérialisées par des cercles) et les transitions (matérialisées par des traits) ; les arcs représentent les relations e et s.

On définit de plus sur un réseau de Petri une application m de P dans N (ensemble des entiers naturels) appelée **marquage.** On obtient ainsi un *Réseau de Petri marqué*. Le marquage des places est représenté par la distribution de $m(p)$ marques (aussi appelées jetons et matérialisées par des points) dans chaque place.

Une transition t est dite **tirable** si et seulement si chacune des places de E reliée à t contient au moins un jeton. Le résultat du tir d'une transition t est l'ajout d'un jeton dans chacune des places de sortie de la transition tirée et le retrait d'un jeton de chacune des places d'entrée de cette même transition. Ceci est représenté dans la figure II.6. Dans cet exemple, le réseau n'est



(a) marquage initial (b) marquage après le tir de la transition t_1

Figure II.6 Exemple de tir d'une transition d'un réseau de Petri

constitué que d'une seule transition t_1 qui n'est tirable que si les places *A* et *B* contiennent au moins un jeton. C'est le cas avec le marquage (a) ; le second, (b), est le résultat du tir de la transition : les places *C*, *D*, *E* se sont vu attribuer un jeton supplémentaire tandis que l'on en retirait un à chacune des places *A* et *B*. Le réseau, tel qu'il est représenté en (b), est dit **bloqué** car aucune transition n'y est plus tirable.

Etant donnés un réseau de Petri et un marquage initial, on peut construire le graphe des marquages associé. Chaque nœud du graphe représente un marquage. Il existe un arc d'un nœud vers un autre si, à partir du marquage représenté dans le premier nœud il est possible d'obtenir le second marquage par le tir d'une seule transition dans le réseau. Le graphe des marquages représente l'ensemble des marquages que l'on peut obtenir à partir du marquage initial.

II.4.2 Illustration du problème de protection

Le modèle *HRU* détecte la présence d'un droit particulier dans une case particulière de la matrice pour conclure qu'un système est non sûr. Le modèle *TG* détecte la présence d'un arc étiqueté d'une façon particulière entre deux nœuds particuliers pour arriver à la même conclusion. De notre côté, nous allons détecter la présence d'un jeton particulier dans une place particulière pour détecter qu'un système de protection ne répond pas à un objectif de protection défini.

Considérons l'exemple trivial représenté dans la figure II.7 et observons si *A* peut parvenir à acquérir le droit α sur *O* en ne profitant d'aucune collaboration. La séquence adéquate est très simple à trouver : *A* prend le droit (*t* sur *C*) de *B*, puis prend le droit (α sur *O*) de *C*. Le sujet *A* est donc bien le seul à avoir exécuté une action.

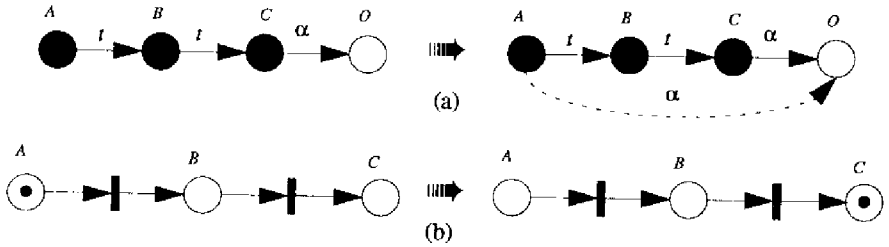


Figure II.7 Exemple d'équivalence entre le modèle *TG* et le réseau de Petri

Voyons comment nous pourrions le transposer sur un réseau de Petri. Nous pouvons parvenir à nos fin en construisant un réseau où chaque arc *take* du modèle *TG* est remplacé par une transition ayant une place d'entrée et une place de sortie ; i) la place d'entrée dans le réseau de Petri correspond au nœud de départ de l'arc dans le modèle *TG* ; ii) la place de sortie correspond au nœud d'arrivée. Ensuite, nous plaçons un jeton dans les places correspondant aux sujets impliqués dans le problème de protection, c'est-à-dire ceux dont nous voulons savoir si, *in fine*, ils peuvent obtenir un droit particulier sur un objet particulier. Nous désignerons ces sujets sous le terme d'**attaquants**.

La figure II.7 (b) montre un tel réseau construit à partir du graphe de la figure II.7 (a). Ensuite, nous identifions les places qui correspondent aux nœuds qui, dans le modèle *TG*, ont le droit convoité. Ces places seront désignées sous le terme de **cibles** par la suite. Dans notre exemple, la place *C* est la seule *cible*. Enfin, nous effectuons toutes les séquences de tir de transitions possibles. Si, au cours d'une de ces séquences, un jeton arrive dans une des places cibles, alors nous déduisons que le système de protection est non sûr. Dans notre exemple, il est trivial de constater que la seule séquence de tir possible amène effectivement le jeton de l'attaquant *A* à la cible *C*. Le système est donc non sûr, comme prévu.

Voici esquissée l'idée générale de la transposition du modèle *TG* en un réseau de Petri. Bien entendu, dans le cas général, ces règles de construction ne suffisent pas : il est nécessaire de prendre en compte les autres règles de réécriture du modèle. De plus, il faut construire le réseau de telle sorte que le tir d'une seule séquence maximale[1], quelconque, suffise pour obtenir la solution attendue. Tout ceci complique d'autant la construction du réseau de Petri, même si la logique reste la même. Nous ne voulons pas rentrer ici dans le détail des règles de transformation, données à l'annexe A, mais il nous semble important de présenter un cas non trivial où apparaît la notion de collaboration.

II.4.3 Réseau de Petri et collaboration

Le réseau de Petri est construit par la substitution des nœuds par des places et des arcs par des transitions, mais cette substitution implique la création de nombreuses places et transitions afin de rendre compte de la logique des différentes règles de réécriture du modèle *TG*. Considérons, par exemple, le graphe de protection décrit dans la figure II.8.

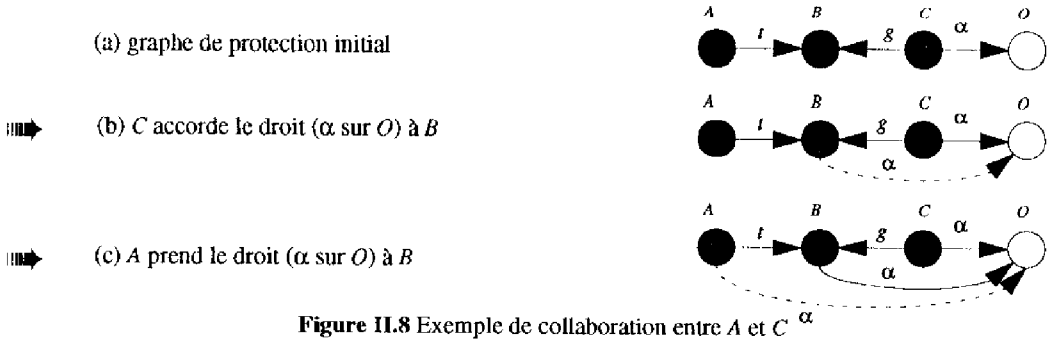


Figure II.8 Exemple de collaboration entre *A* et *C* [α]

1. Une séquence de tir est dite maximale si le réseau de Petri auquel elle aboutit est bloqué.

A, pour acquérir le droit α, a besoin de l'aide de *C*. Ceci doit apparaître également explicitement dans le réseau de Petri. Pour cela, nous créons pour chaque nœud *N* du modèle *TG* une seconde place dans le réseau de Petri, identifiée par l'indice *c* (*c* pour collaborateur). Le marquage initial du réseau de Petri comporte un jeton dans chaque place N_c si et seulement si l'utilisateur correspondant, *N*, est défini comme étant un collaborateur. Cette place est reliée par un arc aux entrées des transitions qui ne peuvent être franchies que si *N* collabore effectivement avec l'assaillant considéré. Ceci est représenté dans la figure II.9 où nous donnons la séquence de tir qui mène du marquage initial (a) jusqu'au marquage final (c) qui possède un jeton dans la place *C*.

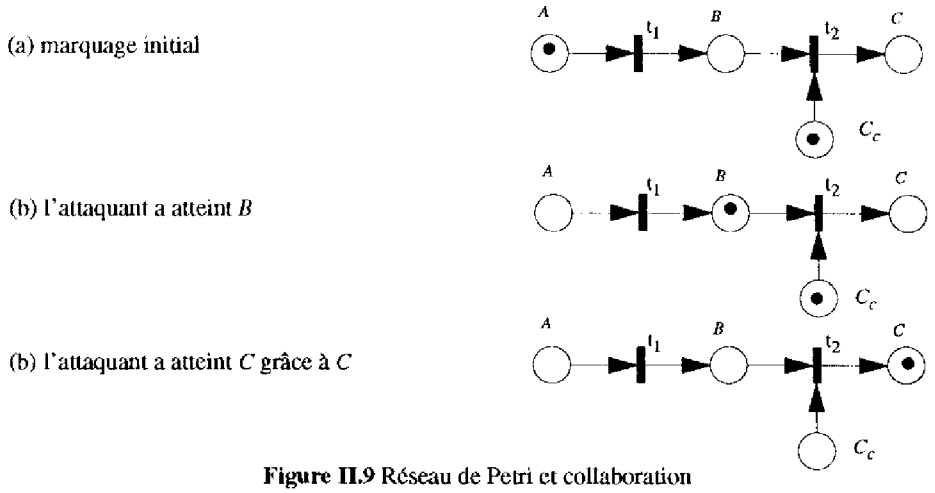


Figure II.9 Réseau de Petri et collaboration

L'attaquant peut arriver à la cible ; le système est donc non sûr. Ceci est dû au fait que *C* a été défini comme un collaborateur. Dans le cas contraire, la place C_C n'aurait pas contenu de jeton dans le marquage initial et la transition t_2 n'aurait pas pu être tirée.

Cet exemple est très simple, mais il permet de fixer les principaux avantages de notre démarche :

- Le marquage de places ad-hoc permet de définir un ensemble d'attaquants et un ensemble de collaborateurs. Cette flexibilité nous permet de nous affranchir de l'hypothèse de pire cas que doivent faire les autres modèles. La définition du marquage initial reste à la discrétion de l'utilisateur du modèle.
- La transformation du modèle *TG* en un réseau de Petri peut être automatique, basée sur des substitutions d'arcs en transitions et de nœuds en places.
- Nous avons montré que la résolution du problème de protection se ramène au tir d'une séquence maximale quelconque. Un tel algorithme de tir existe et il est de complexité linéaire [Dacier 1993][39]. Il est décrit dans l'annexe A.

Pour conclure cette partie, et pour donner une idée au lecteur de la complexité réelle de la procédure de substitution nous donnons dans la figure II.10 le motif de réseau de Petri qui remplace un arc "take" entre deux nœuds *X* et *Y*. Même si, graphiquement, cette construction a l'air complexe, le processus de substitution et d'application de la méthode ne l'est pas au sens

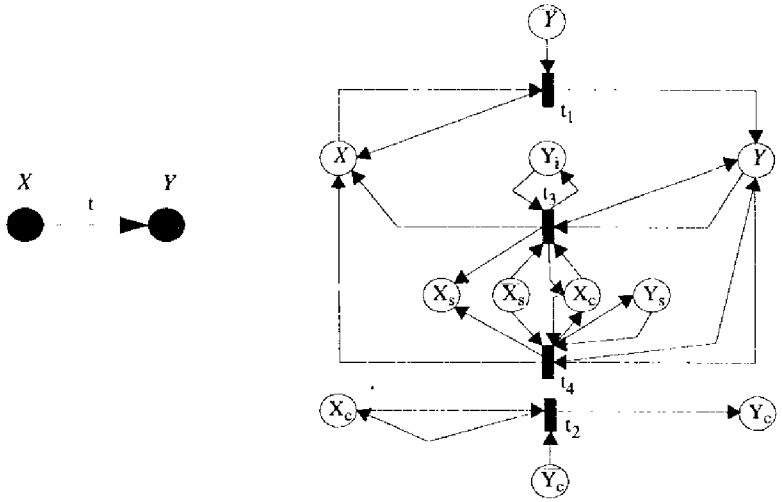


Figure II.10 Réécriture de la règle "take" en un réseau de Petri.

algorithmique du terme. Nous avons montré dans [Dacier 1993][39] que la construction du réseau de Petri est de l'ordre de L applications des règles de réécriture (L étant égal au nombre d'étiquettes t et g dans le graphe de protection initial). Le tir d'une séquence maximale requiert, quant à lui, $3T$ tirs de transition au plus (T étant égal au nombre de transitions dans le réseau de Petri construit). La justification de ces résultats ainsi que des explications détaillées sur le motif sont données dans l'annexe A.

II.4.4 Discussion

Nous avons fait un pas dans la direction d'une définition d'un modèle formel qui s'affranchisse de l'hypothèse de pire cas. Cete étape était nécessaire pour parvenir à définir un modèle qui nous aide à parvenir au but fixé, à savoir l'évaluation quantitative de la sécurité opérationnelle. Toutefois, le modèle obtenu conserve une très faible expressivité : dès le moment où un sujet décide de céder quelque chose, cela ne peut être que tous les droits qu'il détient sur tous les objets ou rien. Nous sommes bien loin de l'expressivité du modèle *HRU*. Afin de combler ce vide entre les deux modèles, Sandhu a défini dans [Sandhu 1988][119] un modèle appelé *SPM* (Schematic Protection Model) qui offre une expressivité et des propriétés algorithmiques intéressantes, mais dont il n'a pas, à ce jour, été possible de prouver l'équivalence avec le modèle *HRU* du point de vue de la puissance d'expression [Sandhu 1992a][120]. Par la suite, Sandhu a proposé un modèle appelé *Typed Access Matrix* (*TAM*) que nous présentons dans la section suivante afin de montrer dans quelle mesure il peut servir notre propos.

II.5 *TAM, ATAM*

II.5.1 Définitions

Le modèle présenté dans [Sandhu 1992a][120], appelé *Typed Access Matrix* (*TAM*), est directement inspiré du modèle *HRU*. Chaque objet appartient à un certain type qui ne peut changer. Les commandes qui déterminent le schéma d'autorisation tiennent compte de cette notion de type. Un modèle de sécurité utilisant *TAM* est composé d'un ensemble fini $\mathcal{R}$ de droits, d'un ensemble fini de types d'objets $\mathcal{T}$ et d'un ensemble fini de types de sujets $\mathcal{T}_s$ ($\mathcal{T}_s \subseteq \mathcal{T}$). Ces éléments sont utilisés pour définir l'état de protection à l'aide d'une matrice de contrôle d'accès typée. Le schéma d'autorisation, quant à lui, est constitué de $\mathcal{R}$, $\mathcal{T}$et d'une collection finie de commandes *TAM*.

Une commande *TAM* doit respecter le format indiqué dans la table II.4 où α désigne le nom de la commande ; $X_1, X_2,..., X_k$ sont des paramètres dont les types sont respectivement $t_1,t_2,...,t_k$; $r_1, r_2,..., r_m$ représentent des droits ; $s_1, s_2,..., s_m$ de même que $o_1, o_2,..., o_m$ représentent des entiers prenant leurs valeurs entre 1 et k .

command $\alpha(X_1 : t_1, X_2 : t_2, ..., X_k : t_k)$
if $r_1 \in [X_{s_1},X_{o_1}] \wedge r_2 \in [X_{s_2},X_{o_2}] \wedge r_m \in [X_{s_m},X_{o_m}]$
then $op_1; op_2; ... ; op_n$

Table II.4 Format d'une commande *TAM*

Chaque op_i est une des opérations primitives définies dans la table II.5 où $z \in \mathcal{R}$ et où s, ainsi que o, sont des entiers à valeur entre 1 et k.

enter z ***into*** $[X_s,X_o]$	***create subject*** X_s ***of type*** t_s	***destroy subject*** X_s ***of type*** t_s
delete z ***into*** $[X_s,X_o]$	***create object*** X_o ***of type*** t_o	***destroy object*** X_o ***of type*** t_o

Table II.5 Les six opérations primitives de *TAM*

Dans ce même article [Sandhu 1992a][120], Sandhu montre qu'il est possible de résoudre le problème de protection dans bon nombre de cas pratiques sans perdre de puissance d'expression. Il décrit un algorithme qui permet d'obtenir un **état maximal de protection**. Cet état se caractérise par une matrice d'accès sur laquelle on ne peut plus éxécuter de règle du schéma d'autorisation.

Il faut également noter qu'une version dite "*augmentée*" de *TAM*, appelée *ATAM*, a été proposée [Ammann et Sandhu 1992][2] afin de fournir un moyen simple de détecter l'absence de droits dans une matrice d'accès. La différence essentielle entre les deux modèles réside dans le fait que le second modèle, *ATAM*, permet l'utilisation de test du type $r_i \notin [X_{s_n},X_{o_m}]$ dans la partie conditionnelle de la commande. La façon de gérer ce type de commande et de résoudre le problème de protection est également définie. L'intérêt de cette démarche est de modéliser facilement la séparation des privilèges évoqués précédemment (section II.1.3, page 32). Cependant, il a été démontré dans [Ganta et Sandhu 1993][54] que les deux modèles sont

équivalents du point de vue de leur expressivité. Néanmoins, l'article montre également l'utilité respective des deux modèles, en fonction des schémas d'autorisation à traiter.

Dans le même ordre d'idées, nous allons à présent étudier l'efficacité de *TAM* pour gérer des schémas d'autorisation particuliers dont les règles induisent la cession de grands ensembles de droits sur de nombreux objets. L'intérêt de cette étude réside dans le fait que ces schémas sont représentatifs de situations réelles et qu'ils requièrent une puissance d'expression virtuellement située entre celle du modèle *TG* et celle du modèle *TAM*.

II.5.2 Un exemple simple

Afin de préciser notre propos, nous allons utiliser au cours des pages qui suivent un exemple simple. Pour cela nous utilisons un ensemble $\mathcal{T}$ de types tel que $\mathcal{T}$= {*sujet, obj1, obj2, obj3*}. De plus, nous définissons l'ensemble $\mathcal{R}$ des droits : $\mathcal{R}$= {*e,o,r,w*} (les lettres sont les initiales habituelles pour les droits correspondant, respectivement, à l'exécution, la possession, la lecture et l'écriture). La table II.6 donne, sous forme d'une matrice d'accès typée[1], l'état de protection initial que nous étudions par la suite.

	f: obj$_1$	*g: obj*$_2$	*h: obj*$_3$	*i: obj*$_3$
a: sujet	*e,o,r,w*			
b: sujet	*e*	*r,w*	*r*	
c: sujet		*e,o,r,w*	*o,r,w*	*r*

Table II.6 Exemple d'état de protection initial

Nous définissons un schéma d'autorisation constitué des deux règles suivantes :

Règle 1 Si un utilisateur U_1 possède un fichier F de type obj_1 et si un utilisateur U_2 peut exécuter ce fichier F, alors U_2 peut accorder à U_1 tous les droits en lecture que U_2 détient sur les fichiers de type obj_3.

Règle 2 Si un utilisateur U_1 peut écrire dans un fichier F de type obj_2 et si un utilisateur U_2 est le propriétaire de F, alors U_2 peut accorder à U_1 tous les droits en lecture et écriture que U_2 détient sur tous les fichiers de type obj_3.

Nous montrons dans le chapitre 3 que ce schéma d'autorisation n'est pas le fruit du hasard, mais qu'il est l'expression d'une réalité concrète.

Munis de cet état de protection initial et de ce schéma d'autorisation, nous allons définir les objectifs de protection sous la forme de deux propriétés que nous voulons voir respectées par le système :

P1 : l'utilisateur *a* ne peut détenir le droit en lecture (*r*) sur l'objet *i* ;
P2 : l'utilisateur *a* ne peut détenir le droit en écriture (*w*) sur l'objet *h*.

1. Par souci de concision, les lignes et colonnes vides n'ont jamais été représentées dans ce document.

II.5.3 Application directe de *TAM*

Afin d'appliquer TAM à cet exemple, il est nécessaire de formaliser le schéma d'autorisation à l'aide des trois règles suivantes :

command $R_1(U_1$: *sujet*, U_2: *sujet*, F_1: obj_1, F_2: $obj_3)$
  if $o \in [U_1,F_1] \wedge e \in [U_2,F_1] \wedge r \in [U_2,F_2]$
    then enter *r* ***into*** $[U_1, F_2]$

command $R_{2r}(U_1$: *sujet*, U_2: *sujet*, F_1: obj_2, F_2: $obj_3)$
  if $w \in [U_1,F_1] \wedge o \in [U_2,F_1] \wedge r \in [U_2,F_2]$
    then enter *r* ***into*** $[U_1, F_2]$

command $R_{2w}(U_1$: *sujet*, U_2: *sujet*, F_1: obj_2, F_2: $obj_3)$
  if $w \in [U_1,F_1] \wedge o \in [U_2,F_1] \wedge w \in [U_2,F_2]$
    then enter *w* ***into*** $[U_1, F_2]$.

La première commande est l'expression de la première règle du schéma d'autorisation ; les deux autres formalisent la seconde règle. La table II.7 représente l'état maximal de ce système de protection. Il peut être obtenu par l'application de plusieurs séquences différentes ; nous en donnons une à titre d'exemple : $R_1(a,b,f,h)$ - $R_{2r}(b,c,g,i)$ - $R_{2w}(b,c,g,h)$ - $R_1(a,b,f,i)$. Quatre applications de commandes sont donc requises. Résoudre le problème de protection est trivial dans ce cas : il suffit de rechercher dans la matrice finale la présence ou l'absence d'un droit précis. Ceci nous amène à constater que seule la seconde propriété, P2, est vérifiée.

	f: obj_1	*g*: obj_2	*h*: obj_3	*i*: obj_3
a: *sujet*	*e,o,r,w*		*r*	*r*
b: *sujet*	*e*	*r,w*	*r,w*	*r*
c: *sujet*		*e,o,r,w*	*o,r,w*	*r*

Table II.7 Exemple d'état de protection maximal

Il est important de constater que le nombre d'applications de commandes est directement proportionnel au nombre de fichiers de type obj_3. Ceci apparaît clairement si nous généralisons l'exemple précédent. Considérons un schéma d'autorisation défini par la seule règle R_1 ; considérons un état de protection composé de n utilisateurs (U_1, U_2, ..., U_n), chacun d'entre eux possède le droit r sur m objets différents de type obj_3. Supposons également l'existence de n-1 fichiers de type obj_1 ($F_1,F_2,...,F_{n-1}$) tels que : $\forall j, (1 \le j \le n-1) \quad o \in [U_j,F_j] \wedge e \in [U_{j+1},F_j]$. Dans ce cas, l'état maximal sera atteint en appliquant m fois la commande avec U_n et U_{n-1} comme paramètres, $2m$ fois avec U_{n-1} et U_{n-2},..., et finalement $(n-1)m$ fois avec U_1 et U_2. Dès lors, le nombre total d'applications de la commande peut s'exprimer : $m + (m + m) + \ldots + (m + \ldots + m) = m \times (1 + \ldots + n-1) = \frac{m \times n \times (n-1)}{2}$

Donc, dans ce cas, $O(mn^2)$ applications de commande sont nécessaires pour atteindre l'état maximal. La section suivante montre que nous pouvons tirer parti de la richesse d'expression de *TAM* afin de trouver une modélisation plus efficace en termes de complexité algorithmique.

II.5.4 Introduire des privilèges ad-hoc

Nous pouvons trouver une solution de coût constant, indépendant du nombre d'objets de type obj_3. Dans ce but, introduisons deux nouveaux droits que nous appelons "*tr*" and "*tw*"[1]. $tr \in [U_1,U_2]$ (respectivement $tw \in [U_1,U_2]$) dénote le fait que l'utilisateur U_1 possède tous les droits en lecture (respectivement, en écriture) de U_2 sur les objets de type obj_3. Ces droits représentent donc, en réalité, des **ensembles de droits**. Nous choisissons d'appeler **privilèges** de tels ensembles. Le schéma d'autorisation s'écrit alors de la façon suivante :

command $R'_1(U_1$: *sujet*, U_2: *sujet*, F_1: $obj_1)$
  if $o \in [U_1,F_1] \wedge e \in [U_2,F_1]$
    then enter *tr* ***into*** $[U_1, U_2]$

command $R'_2(U_1$: *sujet*, U_2: *sujet*, F_1: $obj_2)$
  if $w \in [U_1,F_1] \wedge o \in [U_2,F_1]$
    then enter *tr* ***into*** $[U_1, U_2]$; ***enter*** *tw* ***into*** $[U_1, U_2]$

La table II.8 indique l'état maximal obtenu après application de $R'_1(a,b,f)$ et $R'_2(b,c,g)$ à l'état de protection initial défini dans la table II.6 (page 45).

	b	*c*	*f*: obj_1	*g*: obj_2	*h*: obj_3	*i*: obj_3
a: *sujet*	*tr*		*e,o,r,w*			
b: *sujet*		*tr,tw*	*e*	*r,w*	*r*	
c: *sujet*				*e,o,r,w*	*o,r,w*	*r*

Table II.8 Etat maximal avec les privilèges *tr* et *tw*

A présent, le nombre d'applications requis est seulement égal à n-1, mais la réponse ne peut plus être trouvée par une simple inspection de la matrice car les privilèges "*tr*" et "*tw*" possèdent une sémantique précise dont nous devons tenir compte pour vérifier les propriétés P1 et P2.

En fait, ces privilèges jouent le rôle de pointeurs. Introduire ce type d'éléments dans un modèle *TAM* est un artifice qui a déjà été utilisé par Sandhu et Ganta dans [Ganta et Sandhu 1993][54] pour montrer que les modèles *TAM* et *ATAM* possèdent la même puissance d'expression. Nous définissons ensuite une fonction récursive *possède()* afin de vérifier les propriétés P1 et P2. Son algorithme[2] est donné dans la table II.9.

La fonction *possède(a,i,r)* examine d'abord si la matrice contient le droit *r* dans la case [*a,i*]. Si ce n'est pas le cas, elle cherche un pointeur vers un autre sujet qui pourrait détenir ce droit ou un pointeur vers un sujet tiers, etc. Dans le cas particulier de notre exemple, cinq inspections[3], au minimum, de la matrice sont requises pour vérifier P1 et trois pour P2. Nous constatons que

1. *tr* (respectivement *tw*) est utilisé comme l'abréviation de "*take read* " (respectivement "*take write*").
2. *possède()* renvoie TRUE si la réponse est positive, FALSE si elle est négative. En fait, ceci n'est qu'une vision simplifiée de l'algorithme. Une version complète devrait comporter des tests complémentaires pour assurer la terminaison qui pourrait ne pas se faire si des cycles apparaissent lors de la récursion.
3. *possède(a,i,r)* : $r \notin [a,i] \Rightarrow tr \in [a,b] \Rightarrow r \notin [b,i] \Rightarrow tr \in [b,c] \Rightarrow r \in [c,i] \Rightarrow$ la propriété **P1** n'est pas vérifiée ;
   possède(a,h,w) : $w \notin [a,h] \Rightarrow tw \notin [a,b] \Rightarrow tw \notin [a,c] \Rightarrow$ la propriété **P2** est vérifiée.

```
function possède (U1: sujet, F: object, R: right)
if R ∈ [U1, F]
    then return TRUE ;
    else if(R = r)
        then foreach (U: sujet such that tr ∈ [U1, U])
            { if possède(U, F, R) then return TRUE ;};
    end if;
    else if(R = w)
        then foreach (U: sujet such that tw ∈ [U1, U])
            { if possède(U, F, R) then return TRUE ;};
    end if
end if
return FALSE
```

Table II.9 Algorithme de la fonction *possède()* avec les privilèges *tr* et *tw*

la complexité de cette solution est indépendante du nombre d'objets de type obj_3 et linéaire selon le nombre d'utilisateurs. Ce résultat est à comparer à $O(mn^2)$ obtenu précédemment.

En réalité, la complexité de notre solution est directement proportionnelle au nombre de pointeurs. Si nous considérons un état de protection composé d'un grand nombre d'utilisateurs dont aucun n'a le droit de lire *i*, cette solution est évidemment moins bonne que la première. En effet, elle impose de suivre une longue liste de pointeurs pour finalement délivrer une conclusion négative qui était immédiate avec la première méthode ! Cependant, nous montrerons par la suite que notre exemple de schéma d'autorisation et d'état de protection est représentatif, bien que spécifique, de nombre de situations réelles. Dans ces cas là, la seconde solution est meilleure que la première, du point de vue la complexité algorithmique.

II.5.5 Discussion

Comme nous venons de le voir, l'introduction de privilèges ad-hoc dans un modèle *TAM* peut améliorer de façon significative sa complexité algorithmique dans certaines situations précises. Malheureusement, ceci nécessite au moins un nouveau droit pour chaque classe d'ensemble de privilèges cédés et chaque nouveau droit implique la réécriture de l'algorithme de résolution *possède()* qui doit tenir compte de leur sémantique. Une telle tâche est d'autant plus fastidieuse que l'ensemble des commandes à considérer est grand. Ainsi, bien que cette solution semble prometteuse, elle n'est pas idéale en raison de son manque de modularité.

Nous pourrions être tentés de définir une nouvelle approche en n'utilisant qu'un seul type de pointeur dirigé vers des nouveaux sujets virtuels créés à dessein au fur et à mesure de l'application des règles. Par exemple, pour représenter le fait qu'un utilisateur *b* accorde à *a* tous ses droits en lecture (*r*) sur tous les objets de type obj_3, nous pourrions créer un nouvel utilisateur β, lui donner tous les droits que *b* possède sur tous les objets de type obj_3 et introduire dans la case [*a*,β] un pointeur que l'on appellerait, par exemple, *acquiert*. Cependant, une telle solution pose de nombreux problèmes d'implémentation dans le cadre restreint d'une matrice d'accès typée. Les deux scénarios qui suivent expliquent cela :

(1) Une fois β créé, supposons que *b* acquière de nouveaux droits *r* sur un des objets de type obj_3. Les privilèges de β doivent être mis à jour. Donc, nous devons écrire de

nouvelles règles qui tiennent compte de toutes ces mises à jour nécessaires au fur et à mesure de l'application d'autres règles.

(2) Une fois β créé, supposons que *b* acquière tous les droits en lecture et écriture que *c* possède sur les objets de type obj_3. Pour cela, selon le même principe, un nouvel utilisateur γ est créé (possédant tous les droits *r* et *w* que *c* possède sur les objets de type obj_3) et le droit *acquiert* doit être inséré dans [*b*,γ]. Comment tenir compte de cette modification dans la mise à jour de β ? Si nous ajoutons le droit *acquiert* dans [β,γ] alors *a*, par transitivité, obtiendra les droits en écriture de *c*, ce qui n'est pas correct.

Ces deux scénarios montrent que les règles de mise à jour peuvent être complexes à définir. En réalité, une telle solution pèche par le même manque de modularité que la précédente. Cependant, si nous pouvions *définir* l'ensemble des privilèges de β plutôt que de les *énumérer* lors de sa création, alors nous n'aurions plus de problème de mise à jour puisque le contenu de β serait indépendant de l'évolution de celui de *b*.

De plus, si nous disposions d'une telle définition de β, nous pourrions également résoudre le problème esquissé dans le second scénario en intégrant les commandes de mise à jour dans les commandes habituelles. Nous allons expliquer la façon de réaliser ceci dans la section qui suit ; elle présente une extension de *TAM* capable de gérer ce type de définitions formelles d'ensembles de privilèges.

II.6 Le graphe de privilèges

II.6.1 Définitions

Afin de définir, plutôt que d'énumérer, les privilèges cédés, nous utilisons un graphe dirigé dont les nœuds sont des ensembles de triplets [Dacier et Deswarte 1994][42] : $(U, O, \Sigma_{\mathcal{R}})$ où U représente un sujet, O un objet et $\Sigma_{\mathcal{R}}$ un ensemble de droits $(\Sigma_{\mathcal{R}} \subseteq \mathcal{R})$. Pour chaque type $\theta, \theta \in \mathcal{T}$, nous définissons Σ_θ comme étant l'ensemble des objets de type θ. Nous définissons $\Sigma_{\mathcal{T}}$ comme une union d'ensembles, plus précisément $\Sigma_{\mathcal{T}} = \bigcup_{\theta \in \mathcal{T}} \Sigma_\theta$.

Les nœuds représentent des ensembles de privilèges sur des ensembles d'objets. Un nœud n'est pas supposé correspondre à l'une quelconque des lignes de la matrice d'accès, ni même à une partie d'une ligne. Il représente un ensemble de privilèges qui peut être cédé à un tiers.

L'existence d'un arc d'un premier ensemble de privilèges vers un second indique que la possession du premier permet d'acquérir le second.

Supposons, par exemple, qu'une règle spécifie que le sujet "*b*" peut accorder tous les droits en lecture qu'il a sur tous les objets de type "obj_3". L'application de cette règle créerait un nœud que l'on peut définir de la façon suivante : $N = \{(b,O,r) | O \in \Sigma_{file_3} \wedge r \in [b,O]\}$. Ce nœud représente un sous-ensemble des privilèges que *b* possède effectivement dans la matrice d'accès lorsque la règle est appliquée, mais ce sous-ensemble n'est pas figé. En effet, une telle définition

désigne également les droits insérés dans la matrice *après* que le nœud ait été créé. Ceci est possible puisque le contenu du nœud n'est jamais énuméré mais seulement défini.

Pour tout sujet U, nous définissons $\mathcal{M}_u$ comme l'ensemble maximal de privilèges que U pourrait obtenir. Cet ensemble correspond à la ligne de la matrice représentant l'état maximal, au sens classique *TAM* du terme, représenté dans la table II.7 (page 46). Formellement[1], nous pouvons écrire : $\forall U \quad \mathcal{M}_U = \{(U,O,\Sigma_{\mathcal{R}}) | O \in \Sigma_T \wedge \left(\Sigma_{\mathcal{R}} = \left(\mathcal{R} \cap [U,O]^*\right)\right) \wedge \Sigma_{\mathcal{R}} \neq \varnothing\}$. Il faut noter qu'aucun nœud du graphe ne répondra à cette définition, pour la simple et bonne raison que jamais nous n'aurons à calculer cet état maximal lors de la construction du graphe.

L'existence d'un arc dans le graphe dirigé d'un nœud $\mathcal{N}_1$ vers un nœud $\mathcal{N}_2$ implique que $\forall U, U \in \Sigma_{users}, \mathcal{M}_U \supseteq \mathcal{N}_1 \Rightarrow \mathcal{M}_U \supseteq \mathcal{N}_2$. En d'autres termes, l'existence d'un arc partant d'un nœud $\mathcal{N}_1$ vers un nœud $\mathcal{N}_2$ signifie que tout sujet capable d'acquérir l'ensemble de privilèges représenté par $\mathcal{N}_1$ est capable d'acquérir celui représenté par $\mathcal{N}_2$. Ceci n'est qu'une définition formelle ; elle est inutilisable en pratique puisque, comme nous l'avons dit, nous ne voulons pas calculer la définition en extension de $\mathcal{M}_u$.

En pratique, les nœuds et les arcs sont créés par application des règles qui composent le schéma d'autorisation. Pour cela, nous ajoutons deux opérations primitives aux six déjà définies dans *TAM* (dans la table II.1, page 34) : *make_edge* et *make_node*. L'opération *make_node* (respectivement *make_edge*) crée un nœud (un arc) dans le graphe pour autant que ce nœud (cet arc) n'ait pas déjà été créé.

II.6.2 Construction

Nous avons déjà mentionné que nous ne voulions, à aucun moment, calculer $\mathcal{M}_u$. Avec notre méthode, résoudre le problème de protection revient à trouver un chemin dans un graphe orienté. Le protocole qui suit explique notre façon de procéder :

(1) Pour chaque sujet U dans l'état de protection initial, nous créons un nœud défini de la façon suivante : $\mathcal{N}_U = \{(U,O,\Sigma_{\mathcal{R}}) | O \in \Sigma_T \wedge (\Sigma_{\mathcal{R}} = (\mathcal{R} \cap [U,O])) \wedge \Sigma_{\mathcal{R}} \neq \varnothing\}$. A tout moment, cette définition représentera les privilèges présents dans la matrice pour cet utilisateur[2].

(2) Nous appliquons les commandes jusqu'à atteindre un état maximal[3]. Dans ce cas, l'état maximal est caractérisé non plus par la seule matrice d'accès mais par le graphe construit et par la matrice d'accès associée. Les deux sont nécessaires pour caractériser l'état de protection final et résoudre le problème de protection posé.

(3) Nous reformulons le problème de protection en termes de deux ensembles de nœuds conflictuels et nous cherchons si un chemin existe entre ces deux ensembles.

1. Nous indiquons par la notation $[U,O]^*$, au lieu de $[U,O]$, que nous faisons référence à la matrice d'accès obtenue, une fois atteint l'état maximal de protection.

2. $\mathcal{N}_U$ peut être identifié avec $\mathcal{M}_U$ si et seulement si la matrice décrivant l'état maximal au sens classique *TAM* et celle issue de notre méthode sont identiques. De façon générale ceci n'est pas vrai et nous avons plutôt $\mathcal{N}_U \subseteq \mathcal{M}_U$.

3. Construire le graphe est une tâche finie car le nombre de nœuds dans le graphe sera au plus une combinaison linéaire du nombre de cases dans la matrice décrivant l'état de protection maximal dans le modèle *TAM* classique. La taille de cette matrice étant finie, le graphe est lui aussi de taille finie.

Afin de fixer les idées, nous allons voir comment mettre ces principes en œuvre dans l'exemple utilisé précédemment. Tout d'abord, définissons les commandes qui vont caractériser notre schéma d'autorisation. La première est donnée dans la table II.10. Si on la compare à R'_1, on voit que cette nouvelle commande R''_1 possède un troisième test dans sa partie conditionnelle :

(1) $o \in [U_1,F_1] \wedge e \in [U_2,F_1]$ traite du schéma d'autorisation lui-même ;

(2) $\{(U_1,O,r) | O \in \Sigma_{file_3} \wedge r \in [U_1,O]\} \subseteq \mathcal{N}_1$ identifie dans le graphe le nœud auquel l'opération *make_edge* devra être appliquée. Cette règle sera appliquée à tout nœud qui satisfait cette définition. Ainsi, pour un triplet donné (U_1, U_2, F_1), la règle ne créera jamais qu'un seul nœud $\mathcal{N}_2$, mais, le cas échéant, plusieurs arcs pourraient y arriver, venant de différents nœuds $\mathcal{N}_1$ vers $\mathcal{N}_2$.

command $R''_1(U_1$: *sujet*, U_1: *sujet*, F_1: *obj*$_1$, $\mathcal{N}_1$: *nœud*, $\mathcal{N}_2$: *nœud*)
  if $o \in [U_1,F_1] \wedge e \in [U_2,F_1] \wedge \{(U_1,O,r) | O \in \Sigma_{file_3} \wedge r \in [U_1,O]\} \subseteq \mathcal{N}_1$
    then ***make_node*** $\mathcal{N}_2 = \{(U_2,O,r) | O \in \Sigma_{file_3} \wedge r \in [U_2,O]\}$
      make_edge from $\mathcal{N}_1$ ***to*** $\mathcal{N}_2$
  end if

Table II.10 Première règle exprimée pour le graphe de privilèges.

Les mêmes principes mènent à la définition des règles R''_{2r} et R''_{2w} dans la table II.11.

command $R''_{2r}(U_1$: *sujet*, U_2: *sujet*, F_1: *obj*$_2$, $\mathcal{N}_1$: *nœud*, $\mathcal{N}_2$: *nœud*)
  if $w \in [U_1,F_1] \wedge o \in [U_2,F_1] \wedge \{(U_1,O,r) | O \in \Sigma_{obj_3} \wedge r \in [U_1,O]\} \subseteq \mathcal{N}_1$
    then
    make_node $\mathcal{N}_2 = \{(U_2,O,r) | O \in \Sigma_{obj_3} \wedge r \in [U_2,O]\}$
    make_edge from $\mathcal{N}_1$ ***to*** $\mathcal{N}_2$
  end if

command $R''_{2w}(U_1$: *sujet*, U_2: *sujet*, F_1: *obj*$_2$, $\mathcal{N}_1$: *nœud*, $\mathcal{N}_2$: *nœud*)
  if $w \in [U_1,F_1] \wedge o \in [U_2,F_1] \wedge \{(U_1,O,w) | O \in \Sigma_{obj_3} \wedge w \in [U_1,O]\} \subseteq \mathcal{N}_1$
    then
    make_node $\mathcal{N}_2 = \{(U_2,O,w) | O \in \Sigma_{obj_3} \wedge w \in [U_2,O]\}$
    make_edge from $\mathcal{N}_1$ ***to*** $\mathcal{N}_2$
  end if

Table II.11 Deuxième règle exprimée pour le graphe de privilèges.

Une fois ces commandes définies, nous appliquons les trois étapes décrites plus haut. Le graphe des privilèges qui en résulte est représenté dans la figure II.11. Durant la première étape, nous créons les nœuds ❶, ❷, et ❸. Ensuite, nous appliquons les commandes. Une séquence possible d'applications peut être : $R''_1(a,b,f,❶,❹)$ - $R''_{2r}(b,c,g,❷,❺)$ - $R''_{2w}(b,c,g,❷,❻)$ - $R''_{2r}(b,c,g,❹,❻)$. C'est une des séquences valides qui mènent au graphe représenté dans la figure II.11. La troisième et dernière étape est décrite dans la section qui suit.

II.6.3 Résolution du problème de protection

Pour vérifier la propriété P1, nous suivons les étapes suivantes :

(1) Vérifier dans la matrice si a a accès en lecture à i ; si la réponse est non, aller à l'étape 2.

(2) Identifier dans la matrice les sujets qui ont accès en lecture à i ; dans notre exemple, le seul sujet répondant à ce critère est c.

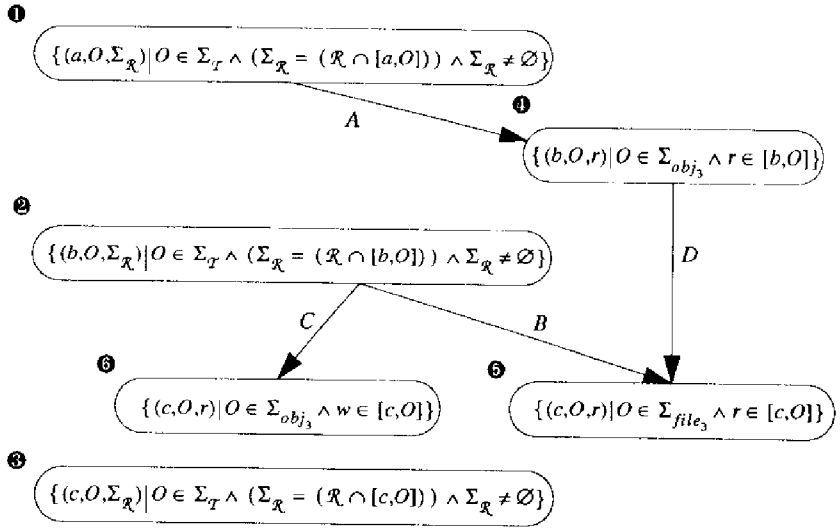


Figure II.11 Exemple d'un graphe de privilèges

(3) Identifier dans le graphe tous les nœuds qui contiennent le triplet (c,i,r) ; cela donne les nœuds ❸ et ❺.

(4) Si un chemin existe entre le nœud représentant l'ensemble de tous les privilèges de a (❶) et au moins un des nœuds identifiés dans l'étape 3, alors nous savons que a peut lire i ; dans notre cas, l'existence d'un chemin entre les nœuds ❶ et ❺ (arcs A et D) implique que la première propriété n'est pas vérifiée.

Quant à la propriété P2, elle est vérifiée car il n'existe de chemin ni entre ❶ et ❸, ni entre ❶ et ❻.

II.6.4 Discussion

Selon les schémas d'autorisations, cette méthode sera plus ou moins efficace du point de vue de la complexité algorithmique. Nous ne pouvons, bien entendu, pas calculer cette complexité de façon générale pour *tous* les schémas possibles. Cependant, nous pouvons dégager des conditions nécessaires pour que notre méthode soit efficace. Nous les reprenons ci-dessous :

(1) Les ensembles de privilèges cédés ne doivent pas contenir de droits d'accès sur des objets qui servent également dans la partie conditionnelle de l'une ou l'autre commande. Ceci nous assure que ces parties conditionnelles peuvent être vérifiées au seul vu de la matrice, sans inspecter le graphe (abstraction faite, bien sûr de la partie conditionnelle qui consiste à identifier un nœud). Ainsi, dans notre exemple, nous accordons des privilèges sur les objets de type obj_3, mais les conditions ne portent que sur des objets de type obj_1 ou obj_2. Si tel n'était pas le cas, il faudrait rechercher l'existence de nœuds précis dans certains chemins du graphe ce qui, en raison de la complexité impliquée, nuirait gravement à l'efficacité du modèle.

(2) Les ensembles de privilèges cédés doivent avoir des intersections aussi réduites que possible entre eux. Le meilleur cas est celui où ils n'en possèdent pas, ce qui minimise le nombre de nœuds et d'arcs créés.

L'exemple qui suit met en exergue le gain potentiel en complexité que l'on peut attendre de cette méthode par rapport à *TAM* lorsque ces deux conditions sont satisfaites. Considérons la matrice d'accès initiale représentée dans la table II.12 ainsi que la commande R_e, à la *TAM*, décrite dans la table II.13. Supposons également l'existence d'une règle R_r (respectivement R_w) où le droit "*e*" a été remplacé par le droit "*r*" ("*w*") .

	x: obj$_1$	*y: obj*$_1$	*z: obj*$_1$	*l: obj*$_2$	*m: obj*$_2$	*n: obj*$_2$
a: sujet	*e,o,r,w*			*e,o,r,w*		
b: sujet		*e,o,r,w*			*e,o,r,w*	
c: sujet			*e,o,r,w*	*o,r,w*		*e,o,r,w*

Table II.12 Nouvel exemple de matrice d'accès initiale

command $R_e(U_1$*: sujet,* U_2*: sujet,* F_1*: obj*$_1$*,* F_2*: obj*$_2)$
  if $o \in [U_1,F_1] \wedge e \in [U_1,F_2]$
    then enter *e* ***into*** $[U_2, F_2]$

Table II.13 Définition de la règle R_e

Dans ce cas précis, les deux contraintes sont satisfaites : i) les objets concernés dans les conditions sont distincts de ceux invoqués dans les cessions et ii) les ensembles de privilèges cédés sont distincts. La commande R'_e, donnée dans la table II.14, est la transcription de la règle R_e dans le cadre du graphe des privilèges.

command $R'_e(U_1$*: sujet,* U_2*: sujet,* F_1*: obj*$_1$*,* $\mathcal{N}_1$*: nœud,* $\mathcal{N}_2$*: nœud)*
  if $o \in [U_1,F_1] \wedge \{(U_2,O,\Sigma_{\mathcal{R}}) | O \in \Sigma_{obj_2} \wedge (\Sigma_{\mathcal{R}} = (\{e\} \cap [U_2,O])) \wedge \Sigma_{\mathcal{R}} \neq \varnothing\} \subseteq \mathcal{N}_1$
    then
    make_node $\mathcal{N}_2 = \{(U_1,O,\Sigma_{\mathcal{R}}) | O \in \Sigma_{obj_2} \wedge (\Sigma_{\mathcal{R}} = (\{e,r,w\} \cap [U_1,O])) \wedge \Sigma_{\mathcal{R}} \neq \varnothing\}$
    make_edge from $\mathcal{N}_1$ ***to*** $\mathcal{N}_2$
  end if

Table II.14 Définition de la règle R'_e

Supposons à présent qu'il existe n utilisateurs, chaque utilisateur possèdant le droit "*o*" sur au moins un objet de type *obj*1. De plus, chaque utilisateur détient k droits qu'il peut céder sur m objets de type obj_2. Dans ce cas, *TAM* nécessitera $n \times k \times m \times (n-1)$ applications de règles pour atteindre l'état maximal. Avec le graphe de privilèges, il en faudra seulement $n \times k \times (n-1+n-1)$. Donc, si les contraintes sont satisfaites, nous observons une réduction de la complexité d'un facteur de $m/2$. Ceci est un gain d'autant plus important, qu'en général nous aurons des valeurs de m largement supérieures à n $(m \gg n)$. Bien entendu, il ne s'agit ici que d'un exemple qui n'a pour seule intention que de mieux illustrer le cadre typique d'application utile de notre méthode.

Si les contraintes ne sont pas satisfaites, il est nécessaire de calculer le meilleur compromis entre le graphe des privilèges et un modèle utilisant directement *TAM*. Ceci ne peut se faire qu'en évaluant la complexité, mais il faut détenir, pour cela, la connaissance complète et

précise du schéma d'autorisation et de la matrice de protection initiale, ou du moins l'adoption d'hypothèses de travail sur leurs caractéristiques principales.

II.7 Conclusions

Dans ce chapitre, nous avons étudié l'état de l'art en matière de modèles formels de sécurité. Nous avons souligné les hypothèses classiquement utilisées sur le comportement des utilisateurs. Nous avons expliqué pourquoi il était souhaitable de s'en libérer pour obtenir un modèle réaliste et comment y parvenir dans le cas d'un modèle relativement simple.

Ensuite, nous avons proposé une extension à un modèle plus expressif, *TAM*, afin d'augmenter son efficacité dans le cas de schémas d'autorisation particuliers. Nous montrons dans le prochain chapitre que ces schémas sont typiques de ceux utilisés dans le monde réel.

Ces deux résultats (levée de l'hypothèse de pire cas et définition d'un nouveau modèle) nous permettront au chapitre IV de définir un modèle formel d'évaluation quantitative de la sécurité, applicable à des systèmes réels et affranchi de l'hypothèse de pire cas.

CHAPITRE III *Application du modèle du graphe de privilèges*

Préambule

Dans les chapitres qui précèdent, nous avons montré que :

(1) L'évaluation de la sécurité d'un système requiert, notamment, la mise en œuvre de méthodes de prévision des fautes.
(2) Les méthodes d'analyse des risques gagneraient à mieux modéliser les systèmes étudiés.
(3) Les modèles formels de sécurité adoptent une hypothèse de pire cas sur le comportement des utilisateurs.
(4) Il est possible de relâcher cette hypothèse pour un modèle simple comme le modèle *TG*.
(5) Un nouveau modèle, appelé graphe des privilèges, se révèle être un candidat intéressant pour une modélisation efficace de certains systèmes réels.

Dans ce chapitre, nous appliquons le modèle du graphe des privilèges dans le cadre des systèmes d'exploitation UNIX. Nous offrons de la sorte une justification pragmatique des hypothèses émises au chapitre précédent. L'utilisation du modèle nous amène ensuite à reprendre la discussion, amorcée au premier chapitre, sur l'utilité des méthodes de prévision de fautes.

Nous commençons par un bref rappel des mécanismes de protection des systèmes UNIX et nous présentons ce que désignent les nœuds et les arcs du graphe de privilèges dans ce cadre d'application. Nous expliquons quelques méthodes de cession de privilèges, en illustrant chacune d'entre elles par un exemple.

Nous montrons ensuite comment utiliser le modèle du graphe des privilèges dans ce cadre concret. Un exemple simple est présenté et son utilisation discutée. Il ressort de cette discussion que l'élimination de fautes intentionnelles non malveillantes se révèle délicate, voire même impossible, à mettre en œuvre dans ce contexte. Nous proposons alors intuitivement une manière d'enrichir le graphe de privilèges pour évaluer les conséquences, sur les objectifs de sécurité, de l'existence de fautes résiduelles qu'il n'est pas possible —ou pas souhaitable— d'éliminer.

III.1 Cadre d'application

Nous avons choisi d'appliquer le graphe de privilèges à la famille des systèmes d'exploitation UNIX[1]. Notre référence de travail sera le système SunOS4.1, développé par Sun Microsystems. Cependant, à quelques détails d'implémentation près, les explications données ci-après s'appliquent également aux systèmes développés par d'autres constructeurs (ULTRIX, SINIX, AIX, AUX, HP-UX, etc.).

Dans le "monde" UNIX, chaque utilisateur est identifié par un numéro, appelé UId (*User Identifier*). Il est possible de définir des groupes d'utilisateurs. Chaque groupe est identifié par un numéro, appelé GId (*Group Identifier*). Chaque utilisateur fait au moins partie d'un groupe et il peut aussi appartenir à plusieurs groupes.

Tous les fichiers UNIX sont stockés dans des répertoires qui peuvent également contenir d'autres répertoires. La structure du système de fichiers est souvent assimilée à un arbre[2] dont la racine est un répertoire et dont les feuilles sont les fichiers. Chaque objet (fichier ou répertoire) appartient à un utilisateur et à un groupe. Tout utilisateur peut gérer trois types de droits sur les objets dont il est le propriétaire : lecture, écriture, exécution. Il peut accorder ou réfuter chacun de ces droits à trois ensembles distincts d'utilisateurs : lui-même ; les membres du groupe propriétaire de l'objet (lui-même excepté) ; tous les autres utilisateurs.

Lorsqu'un utilisateur u veut exercer un droit r sur un objet o, le protocole de vérification décrit dans la table III.1 est utilisé.

```
Si (u est propriétaire de o)
    alors { si (le droit r sur o est accordé au propriétaire de o)
            alors { succès }
            sinon {échec}
            }
    sinon {si (u appartient au groupe propriétaire de o)
            alors { si ( le droit r sur o est accordé au groupe propriétaire de o)
                    alors { succès }
                    sinon {échec}
                    }
            sinon { si ( le droit r sur o est accordé aux autres utilisateurs)
                    alors { succès}
                    sinon { échec}
                    }
            }
```

Table III.1 Protocole de contrôle d'accès

Ce protocole masque toutefois trois notions importantes :

(1) Créer ou détruire un fichier sont des actions qui portent sur le répertoire contenant ce fichier et non sur le fichier lui-même. Pour créer ou détruire un fichier, il suffit d'avoir le droit d'écrire dans le répertoire qui contient ce fichier. Cette façon de procéder

1. Ces travaux ont fait l'objet de plusieurs publications : [Dacier et Rutsaert 1991a][36], [Dacier et Rutsaert 1991b][37] et [Dacier 1992][38].

2. La réalité est plus complexe, mais nous pouvons nous contenter de cette vision simplifiée des choses car elle est adéquate pour traiter du problème de protection.

découle du fait qu'un répertoire est lui-même considéré comme un fichier contenant des fichiers.

(2) Pour pouvoir exercer ses droits sur un objet, il faut avoir le droit de passer dans les différents répertoires qui composent, dans l'arbre du système de fichiers, le chemin aboutissant à ce fichier depuis la racine de l'arbre. On peut traverser un répertoire si on possède le droit d'exécution sur celui-ci.

(3) Certaines implémentations permettent la définition de droits vis-à-vis de plusieurs groupes d'utilisateurs distincts (*HP-UX* [HP 1988][75], par exemple, à l'aide des *Access Control Lists* — ACL's [Strack 1990][137]). Dans ce cas, l'algorithme s'en trouve compliqué et des règles de priorité entre groupes doivent être instaurées.

UNIX est conçu pour être utilisé dans le cadre d'une politique discrétionnaire : chaque propriétaire d'un objet est libre de partager ou non de l'information avec les autres. De plus, il est libre de céder tous ses privilèges à autrui, ou une partie d'entre eux seulement. Comme nous allons le voir, cette particularité peut se révéler, à la fois, favorable et néfaste pour la sécurité.

III.2 Nœuds et arcs du graphe de privilèges

III.2.1 Les nœuds

Dans ce contexte précis d'application, les nœuds du graphe de privilèges peuvent être regroupés en *deux familles* principales :

(1) Dans la première, un nœud représente l'ensemble de tous les privilèges que possède un utilisateur donné dans la matrice initiale de protection. Il existe un nœud de ce type pour chaque utilisateur. Formellement, cette famille reprend l'ensemble des nœuds $\mathcal{N}_u$ définis au chapitre précédent.

(2) Dans la seconde, chaque nœud représente un sous-ensemble de privilèges communs à un groupe d'utilisateurs. Ce groupe peut, éventuellement, se réduire à un singleton.

Nous utilisons également deux nœuds particuliers, suffisamment spécifiques pour leur donner un nom : les nœuds *interne* et *externe*.

Le nœud *interne* représente l'ensemble des privilèges communs à tous les utilisateurs enregistrés du système. Un de ces privilèges est, par exemple, le droit de lire le fichier contenant le nom de tous les utilisateurs ainsi que leur mot de passe chiffré. Cet ensemble peut être vide.

Le nœud *externe* représente l'ensemble des privilèges dont dispose un individu qui est inconnu du système. Cette définition ne peut se comprendre qu'en utilisant le mot privilège dans son acception la plus large. Par exemple, si le système étudié est connecté au réseau Internet, tous les utilisateurs de ce réseau disposent du privilège de tenter une connexion avec lui. Ils peuvent

également profiter de certains services mis à leur disposition tels que le transfert de fichiers, le courrier électronique, les serveurs *WWW*, etc.

III.2.2 Les arcs

Tout comme les nœuds, les arcs peuvent être regroupés en deux familles. Un arc d'un nœud $\mathcal{N}_1$ vers un nœud $\mathcal{N}_2$ peut signifier que :

(1) Soit l'ensemble de privilèges défini dans $\mathcal{N}_2$ est un sous-ensemble de celui défini dans $\mathcal{N}_1$.
(2) Soit il existe une méthode qui permet à un utilisateur disposant des seuls privilèges définis dans $\mathcal{N}_1$ d'acquérir ceux définis dans $\mathcal{N}_2$.

Dans le cadre des systèmes UNIX, il existe de nombreuses méthodes qui permettent d'acquérir les privilèges d'un tiers. Dans la section qui suit, nous reprenons quelques-unes des méthodes les plus connues. Pour chacune d'entre elles, nous expliquons son fonctionnement, en quoi son utilisation peut profiter à la sécurité, comment un intrus peut en abuser et, enfin, nous donnons un exemple concret de son utilisation ainsi que sa représentation en termes de graphe de privilèges.

III.3 Quelques méthodes de cession de privilèges

Notre mémoire n'a pas pour ambition de donner une liste exhaustive des différentes méthodes d'intrusion au sein des systèmes UNIX. Nous encourageons le lecteur intéressé par cet aspect des choses à parcourir la nombreuse littérature existant sur le sujet[1]. Notre ambition se borne ici à donner des exemples de ce que nous appelions, au premier chapitre, des fautes intentionnelles non malveillantes. La compréhension de ces méthodes facilite l'introduction des notions développées dans le prochain chapitre.

III.3.1 Les fichiers *.rhosts*

A. Description de la méthode

Pour pouvoir travailler sur une machine Unix, il faut au préalable fournir un nom d'utilisateur et un mot de passe. Ce sont les phases d'identification et d'authentification. Cependant, il est possible aux utilisateurs qui travaillent déjà sur une machine distante de se connecter sur une autre machine en ne répétant que la seule phase d'identification. Ceci est possible grâce au mécanisme des fichiers *.rhosts*.

1. Par exemple : [Grampp et Morris 1984][58], [Morris 1985][105], [Ritchie 1986][116], [Spafford 1986][129], [Wood et Kochan 1987][146], [Duff 1989][46], [McIlroy 1989][101], [Brand 1990][26], [Curry 1990][35], [Archimbaud 1992][7], [CERT 1992][31], [Millet et Vu 1992][103], [Farmer et Venema 1993][48].

Considérons deux machines, M_1 et M_2 connectées entre elles par un réseau ; supposons que l'utilisateur u_1 travaille sur la machine M_1 tandis que u_2 travaille sur M_2. L'utilisateur u_2 peut se connecter sur la machine M_1, sous l'identité de u_1, sans connaître le mot de passe de ce dernier[1], si les deux conditions suivantes sont réunies : (i) u_2 est connecté à la machine M_2 ; (ii) u_1 a créé un fichier *.rhosts* dans son répertoire personnel et il y a indiqué le nom de u_2 ainsi que la machine depuis laquelle il pouvait se connecter, M_2.

Mettre u_2 dans son ".rhosts" revient pour u_1 à lui céder la *quasi* totalité de ses privilèges. L'ensemble cédé ne sera cependant pas identique à l'ensemble des privilèges de u_1 car certaines actions seront interdites à u_2. Changer le mot de passe de u_1 en est un exemple. D'autres actions qui nécessitent d'être physiquement face à la console de M_1 lui sont également impossibles.

B. Avantages et inconvénients

L'usage du *.rhosts* peut se révéler particulièrement intéressante dans certains cas de figure :

- Un utilisateur dispose de plusieurs comptes sur différentes machines : l'utilisation de fichiers *.rhosts* est une des solutions qui lui permettent de minimiser le nombre journalier de phases d'authentification. L'avantage d'utiliser un fichier *.rhosts* est double dans la mesure où cette phase est fastidieuse et dangereuse puisque, à chaque fois, son mot de passe transite, en clair, sur les réseaux qui relient les machines entre elles.
- Un compte est partagé par un groupe de personnes : ici aussi, il est plus simple d'inclure une liste de noms dans un fichier que de partager le secret du mot de passe. De même, il est plus facile de retirer un nom de la liste que de demander à tous de changer de mot de passe. Enfin, cette façon de faire permet d'identifier l'individu du groupe qui abuserait des droits de ce compte commun. Si aucune identification préalable n'était requise, il bénéficierait au contraire de l'anonymat qui lui garantirait une impunité totale.

C. Exemple d'utilisation

Soit un système comprenant les utilisateurs *A* et *B* qui doivent collaborer dans le cadre d'un projet, *PROJ1*, de développement de logiciel. Pour se simplifier la tâche, harmoniser leurs environnements et disposer d'une version commune de test toujours à jour, ils demandent la création d'un utilisateur virtuel P_1 et le nom de chacun des membres du projet est ajouté au fichier *.rhosts* de ce nouvel utilisateur.

Nous pouvons donc dire que l'utilisateur "virtuel" P_1 cède l'ensemble[2] de ses privilèges aux utilisateurs *A* et *B*. Cette cession de privilèges est représentée en termes de graphes de privilèges dans la figure III.1. Les arcs y sont étiquetés de façon à rappeler la raison pour laquelle ils ont été créés. Dans la légende, la convention d'écriture "*X action Y*" signifie qu'il

1. A l'aide de la commande "*rlogin* M_1 *-l* u_1".

2. Comme cela a été expliqué précédemment, ceci n'est pas vrai en toute rigueur puisque P_1, par exemple, ne cède pas le droit de changer son mot de passe. Par souci de concision, nous admettrons qu'il cède effectivement l'ensemble de ses privilèges. Cet abus d'écriture simplifie la représentation et, par là, la compréhension sans pour autant modifier les concepts fondamentaux de notre démarche.

est possible, à partir de l'ensemble de privilèges *X*, d'acquérir l'ensemble *Y* grâce à une certaine *action*.

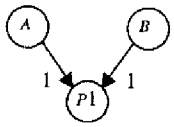


Figure III.1 Collaboration au sein d'un projet

III.3.2 Les fichiers de configuration

A. Description de la méthode

La plupart des logiciels, lorsqu'ils sont exécutés, commencent par lire un fichier, propre à chaque application et à chaque utilisateur, où sont définies des variables de configuration et des procédures d'initialisation. La définition préalable de ces fichiers peut se révéler fastidieuse pour un néophyte. Ce dernier peut opter pour l'utilisation d'un fichier déjà défini par quelqu'un d'autre, plus compétent en la matière, en remplaçant son propre fichier par un lien symbolique[1] vers le fichier de l'expert. En conséquence, à chaque fois que le novice utilisera le programme concerné, c'est le fichier de l'expert qui sera préalablement exécuté. Le novice délègue donc ses pouvoirs pour l'initialisation de l'application à l'expert.

B. Avantages et inconvénients

Certains fichiers, s'ils sont mal configurés, peuvent représenter un danger pour la sécurité de leur propriétaire. Définir une bonne configuration, du point de vue de la sécurité, n'est pas chose aisée et, dès lors, utiliser le fichier de quelqu'un qui est au fait des problèmes peut être une bonne méthode de prévention de fautes.

Bien entendu, puisque le novice va utiliser la configuration d'un tiers, il va exécuter les commandes que ce dernier aura définies. Durant tout le temps de l'exécution, l'expert va, virtuellement, agir en lieu et place du novice. C'est une cession, *de facto*, des privilèges du novice à l'expert. L'ensemble des privilèges cédés, dans ce cas, est fonction de l'application qui est lancée et de ce que l'on peut effectivement faire faire par le fichier de configuration.

C. Exemple d'utilisation

Toute personne désirant utiliser le gestionnaire de fenêtres *X-window* doit exécuter un programme appelé *xinit*. Celui-ci va utiliser le fichier *.xinitrc*, situé dans le répertoire personnel de l'exécutant. Ce fichier contient, entre autres choses, des commandes à lancer au démarrage du gestionnaire et il peut ne pas appartenir à la personne qui démarre *X-window*. Supposons l'existence de deux personnes *B* et *C* qui toutes deux utilisent *X-window*. La première connaît très bien les richesses du fichier *.xinitrc* tandis que la seconde ne s'est jamais penchée sur son

1. Le néophyte pourrait également réaliser une simple copie du fichier de l'expert, mais il ne bénéficierait alors pas des mises à jour que l'expert pourrait faire sur son propre fichier.

contenu. De plus, *C* connaît *B* et lui fait confiance. Afin de profiter de son expérience, *C* choisit d'utiliser le fichier de configuration de *B*. En termes de graphe de privilèges, cette cession est représentée dans la figure III.2

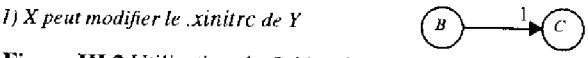


Figure III.2 Utilisation du fichier de configuration d'un expert

III.3.3 Le chemin d'accès aux répertoires de commandes

A. Description de la méthode

L'environnement de travail des utilisateurs est défini, notamment, par un ensemble de variables. L'une d'entre elles porte le nom de *$PATH*. Elle contient une liste de répertoires. A chaque fois qu'un utilisateur veut lancer un programme sans spécifier le chemin d'accès complet vers l'exécutable, le système va inspecter, successivement, le contenu de chacun des répertoires définis dans *$PATH* afin de trouver le programme désigné.

B. Avantages et inconvénients

Cette variable offre un très grand confort d'utilisation à l'utilisateur qui ne doit pas mémoriser les endroits où se trouvent les logiciels mais seulement leurs noms. Cependant, toute personne qui possède le droit d'écrire dans l'un des répertoires de la liste a la possibilité d'y déposer un cheval de Troie[1] [Thompson 1984][141]. Nous donnons deux exemples où *B* peut faire exécuter par *A*, à son insu, les commandes qu'il désire. Durant tout le temps de l'exécution du cheval de Troie, *B* possède les privilèges de *A* :

(1) Soit un programme $prog_1$, situé dans un répertoire dir_n situé en $n^{ème}$ position dans la liste définie dans la variable *$PATH* de l'utilisateur *A*. Soit un utilisateur *B* qui a le droit d'écrire dans un répertoire dir_i ($i<n$), défini en $i^{ème}$ position de la variable *$PATH* de l'utilisateur *A*. Supposons que *B* crée un fichier exécutable ayant pour nom $prog_1$ dans le répertoire dir_i. C'est ce programme, et non l'autre, qui sera exécuté par *A* lorsqu'il tapera la commande $prog_1$ puisque le contenu de la variable *$PATH* est examiné séquentiellement.

(2) Même si *B* n'a accès qu'au dernier répertoire de la liste, il peut en tirer profit en y créant des fichiers exécutables portant les noms de commandes courantes, mais comprenant une coquille typographique (*sl* pour *ls*, *xhosts* pour *xhost*, etc.).

C. Exemple d'utilisation

La plupart des systèmes UNIX sont gérés par un ensemble d'administrateurs. Ceux-ci appartiennent à différents groupes et possèdent différents privilèges, selon leurs compétences et leurs fonctions. Supposons que *F* et *G* soient les deux administrateurs chargés de gérer les logiciels en rapport avec *X-window*. Eux seuls détiennent les privilèges nécessaires pour

1. Un cheval de Troie est un programme effectuant une fonction illicite tout en donnant l'apparence d'effectuer une fonction légitime . Son fonctionnement requiert deux étapes : (i) *installation* du cheval de Troie ; (ii) *activation* du programme par un utilisateur innocent.

modifier le contenu des répertoires en rapport avec ce logiciel. Une façon simple de procéder est de créer un groupe X_{admin} dont ils sont les uniques membres et d'accorder à ce seul groupe les droits d'accès en écriture aux répertoires correspondants. Dans la figure III.3, nous avons représenté cette cession de privilège par un arc depuis F et G vers X_{admin} qui montre que X_{admin} est un ensemble de privilèges communs à F et G. La détention des privilèges de X_{admin} permet, moyennant l'insertion d'un cheval de Troie, d'acquérir les privilèges de tous les utilisateurs de ce logiciel (A, B, C, F et G). Ceci est également représenté sur la figure III.3.

1) X est un sur-ensemble des privilèges Y;
2) X peut modifier un répertoire présent dans le $PATH de Y

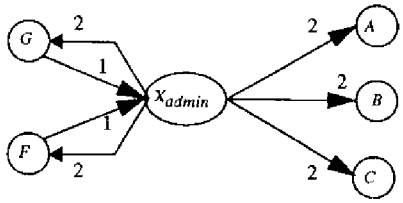


Figure III.3 Abus de pouvoir des administrateurs

III.3.4 Le bit de protection *SetUid.*

A. *Description de la méthode*

Il est possible de spécifier qu'un programme, lors de son exécution, possède les droits de son propriétaire plutôt que ceux de la personne qui l'utilise réellement. Un bit de protection particulier, le bit *setuid*, doit être positionné pour que le programme ait cette propriété [Irvine et al. 1989][76].

B. *Avantages et inconvénients*

L'utilisation du bit *setuid* permet d'envisager un contrôle d'accès qui ne porte plus sur les objets, mais bien sur les transactions. Par exemple, les utilisateurs ne peuvent pas avoir accès en écriture sur le fichier qui contient les mots de passe, mais il doit leur être permis d'initier une transaction qui modifie leur propre mot de passe dans ce fichier. Cette apparente contradiction est résolue grâce à un programme, *passwd*, appartenant au super-utilisateur[1] et ayant son bit *setuid* positionné. Cette fonctionnalité permet de céder des privilèges importants tout en contrôlant l'utilisation qui en est faite.

Si une personne autre que le propriétaire peut modifier les instructions qui doivent être exécutées, alors existe la possibilité pour celle-ci de s'accaparer les privilèges du propriétaire pendant l'exécution du programme modifié.

C. *Exemple d'utilisation*

Un stagiaire, D, est intégré à l'équipe du projet *Proj*1 cité plus haut. Il développe certaines parties du code qui seront insérées dans le logiciel complet. Pour tester son travail, il a besoin de pouvoir exécuter son programme avec les privilèges de P_1. Comme il n'en a besoin que de façon sporadique et pour une tâche bien précise, la solution du bit *setuid* a été préférée à celle

1. Le super-utilisateur, aussi appelé *root*, est le seul utilisateur à avoir accès en écriture au fichier des mots de passe.

du fichier *.rhosts*. Les autres membres du projet ont donc créé un programme appartenant à P_1 et ayant le bit *setuid* positionné. Ce programme utilise les routines sur lesquelles D travaille et ce dernier peut l'exécuter à sa guise. Il pourrait, s'il était mal intentionné, utiliser ce programme afin d'acquérir les privilèges de P_1. Ceci est représenté dans la figure III.4 en termes de graphes de privilèges.

1) X peut modifier l'exécution d'un programme dont le setuid est positionné et qui appartient à Y

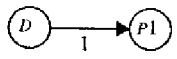


Figure III.4 Utilisation du bit *setuid*

III.3.5 Conclusions

Nous pourrions continuer à dresser la liste des méthodes de cession de privilèges, mais cela n'apporterait plus grand-chose à notre propos. Par ces quelques exemples, nous avons seulement voulu fixer les idées du lecteur sur quelques méthodes bien réelles. En particulier, nous réfutons l'idée selon laquelle les cessions de privilèges trouvent leur origine dans la distraction des utilisateurs, voire dans leurs mépris des plus élémentaires consignes de sécurité. Au contraire, leur utilisation a pour but de trouver le meilleur compromis entre le confort des utilisateurs et les impératifs de sécurité. Isolées les unes des autres, elles semblent salutaires, en revanche, leurs interactions sont pernicieuses. En effet, un utilisateur mal intentionné peut, en les utilisant successivement, parvenir à accaparer des privilèges que personne, au départ, ne désirait lui céder. Nous illustrons ce point par la construction complète du graphe d'un système ; ceci nous permet de montrer un exemple concret d'utilisation du modèle du graphe des privilèges décrit au chapitre précédent.

III.4 Construction du graphe

Hormis les méthodes de cession explicites évoquées à la section précédente, il est important de garder à l'esprit qu'existent également des techniques qui permettent d'acquérir des privilèges sans le consentement de leurs détenteurs. Ces méthodes, comme les premières, ont une influence sur la sécurité globale. Il faut donc en tenir compte lors de la construction du graphe. Enfin, il faut également modéliser les privilèges que deux attaquants pourraient décider de mettre en commun pour mener à bien une intrusion. Ceci nous amène à distinguer trois types de transfert de privilèges pour étudier soit ce qui est cédé, soit ce qui peut être pris, soit ce qui pourrait être cédé.

III.4.1 Les cessions explicites

Nous avons envisagé ces méthodes dans la section précédente. Elles se caractérisent par le fait que leur mise en œuvre est visible dans le système car elle implique la modification du contenu d'un fichier particulier ou d'un répertoire. Il suffit donc, pour construire le graphe, d'inspecter

systématiquement les différents fichiers et répertoires afin d'identifier les différentes méthodes utilisées.

A chaque technique de cession de privilèges connue correspond une règle de création de nœuds et d'arcs. Ces règles sont simples mais propres à chaque système d'exploitation utilisé. Elles devront subir des ajouts ou modifications en fonction de logiciels spécifiques à l'environnement considéré.

L'inspection systématique du contenu des fichiers *.rhosts*, par exemple, générera un ensemble d'arcs. Pour chaque utilisateur, l'identification des propriétaires de tous ses fichiers de configuration en donnera d'autres, et ainsi de suite. Nous avons réalisé un prototype qui implémente ces règles et permet la construction automatique du graphe. Nous le décrirons brièvement dans le chapitre V, consacré à l'outil réalisé pour valider nos travaux.

III.4.2 Les méthodes utilisées lors d'une attaque

Nous désignons par *attaque* la séquence d'actions qu'un utilisateur malveillant effectue pour acquérir de nouveaux privilèges. En effet, nous avons jusqu'ici tenté d'identifier les privilèges effectivement cédés mais il est tout aussi important de détecter ceux qui *pourraient* être pris. Par exemple, nous avons expliqué que l'utilisation du *.xinitrc* d'un tiers était une cession explicite de privilèges. La plupart du temps, ce sera un acte conscient et volontaire. Imaginons maintenant que ce fichier appartienne à *A*, mais soit accessible en écriture à d'autres utilisateurs. Vraisemblablement, il s'agit là d'une distraction de la part de *A*, *mais* le principe de la cession de privilèges est le même : toute personne pouvant écrire dans ce fichier *pourrait* accaparer les privilèges de *A*.

Si les techniques sous-jacentes sont les mêmes que précédemment, les méthodes de détection sont différentes ainsi que, vraisemblablement, les mesures à prendre. En particulier, il semble raisonnable de vouloir éliminer les fautes détectées dans ce cas-ci plutôt que de prévoir leurs conséquences. Ce point de vue se défend, mais n'est pas toujours possible. En effet, il existe des attaques dont on connaît l'existence et dont on ne peut se défaire. Un cas concret, mais trivial, est l'attaque par les mots de passe. L'espace de définition des mots de passe étant fini, il existe une probabilité non nulle de deviner le mot de passe de tout utilisateur. L'attaque est donc possible, mais d'autant moins probable que le mot de passe aura été "bien" choisi et que les mots de passe seront "bien" protégés. Nous reviendrons sur ce point lors du problème de l'évaluation, dans le chapitre IV.

Sans entrer dans le détail de l'implémentation, il est bon de constater que le travail de construction du graphe, pour cette deuxième famille, peut être grandement facilité par l'utilisation d'outils tels que COPS [Farmer et Spafford 1990][47] ou KUANG [Baldwin 1987][12]. Cependant, ces outils servent essentiellement à permettre la détection de fautes au sein de systèmes UNIX. Ils détectent ce qu'ils considèrent être de mauvaises configurations et avertissent l'administrateur afin qu'il prenne les mesures correctives nécessaires. Ils n'envisagent pas la notion de fautes intentionnelles non malveillantes et ne tentent donc pas

d'étudier leur influence sur les objectifs de sécurité. C'est cette étape supplémentaire dont nous soulignons l'importance et pour laquelle la construction du graphe de privilèges est nécessaire.

III.4.3 Les collaborations

Tout utilisateur est libre de gérer à sa guise les droits des objets dont il est propriétaire. Il peut donc céder ce qu'il veut, à qui il veut. Transcrire cela en termes d'arcs et de nœuds dans le graphe de privilèges n'est pas envisageable. Toutefois, ce n'est nécessaire que si nous faisons l'hypothèse que chacun va partager avec tous tout ce qu'il peut partager. C'est bien là l'hypothèse de pire-cas dont nous avons cherché à nous affranchir, dans le cadre du modèle *TG*. Il est donc raisonnable, ici aussi, de vouloir s'en libérer.

Dans un premier temps, nous ne représentons donc pas dans le graphe ce qu'un utilisateur pourrait *délibérément* céder mais uniquement ce qu'il cède *explicitement* ainsi que ce qu'il pourrait céder *malgré lui*. Par la suite, nous montrons comment enrichir le graphe afin de modéliser la collusion éventuelle de plusieurs utilisateurs désireux d'accaparer les privilèges d'autrui. Ceci sera fait en utilisant une technique de représentation à l'aide d'un réseau de Petri, tout comme nous l'avions fait au chapitre II pour le modèle *TG*.

III.5 Utilisation concrète du graphe

Nous allons examiner comment, à l'aide du graphe de privilèges, il est possible d'obtenir une image aussi précise que possible de la sécurité globale d'un système. De plus, nous verrons comment utiliser les résultats obtenus afin d'améliorer cette sécurité. Nous identifions trois phases différentes dans ce processus. Tout d'abord, il est nécessaire de construire le graphe, ensuite il faut vérifier que les cessions de privilèges sont licites, bénéfiques et désirées. Enfin, il faut confronter le graphe obtenu aux objectifs de sécurité du système, dans son ensemble. Ces trois phases sont examinées successivement ci-dessous.

III.5.1 Obtention du graphe

Dans ce qui suit, nous nous basons sur un exemple précis afin de livrer les résultats des trois procédures distinctes qui permettent d'aboutir à la construction du graphe.

Dans un premier temps, nous recensons la liste de tous les utilisateurs du système en inspectant le fichier */etc/passwd*. Cette analyse révèle l'existence de neuf utilisateurs dont les noms sont : *A*, *B*, *C*, *D*, *E*, *F*, *G*, P_1 et P_2. Neuf nœuds, correspondant aux droits de ces neuf utilisateurs, sont construits dans le graphe des privilèges. Un dixième nœud, *interne*, correspond au sous-ensemble de privilèges communs à tous ces utilisateurs est également créé. Ce système n'étant pas relié à un réseau, nous ne définissons pas de nœud *externe*.

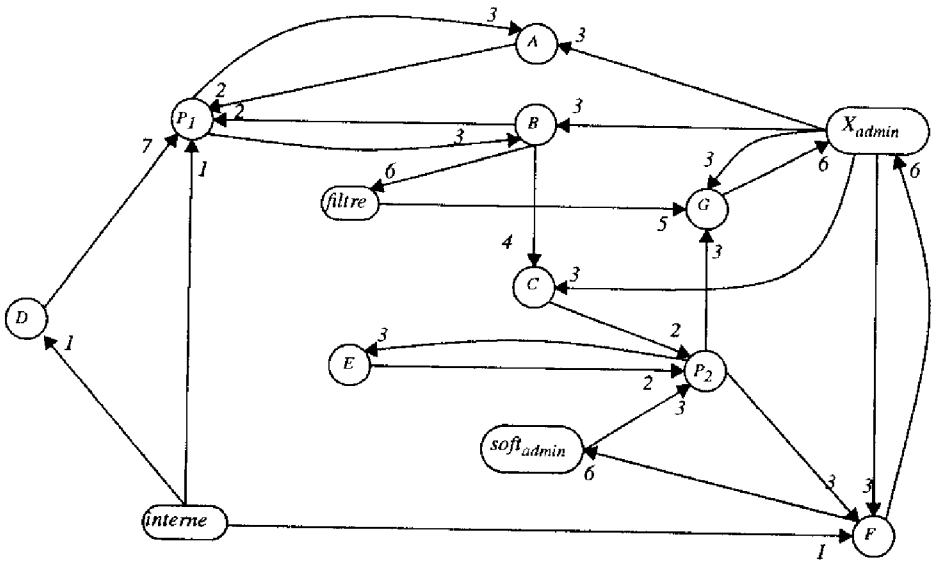
Dans un deuxième temps, nous recensons la liste de tous les groupes d'utilisateurs ainsi que leurs membres en inspectant le fichier */etc/group*. Cette analyse révèle l'existence de trois groupes : X_{admin}, $soft_{admin}$ et *normal*. Les utilisateurs *F* et *G* appartiennent au groupe X_{admin}. Un nœud est donc créé pour représenter les privilèges communs aux membres de ce groupe et un arc tracé vers lui à partir de chacun de ses membres. Nous faisons de même pour le groupe $soft_{admin}$ dont le seul membre est *F*. Enfin, il apparaît que tous les utilisateurs, sans exception, appartiennent au groupe *normal*. Il est donc inutile de créer un nœud pour ce groupe car il ferait double emploi avec le nœud *interne*, déjà créé.

Dans un troisième temps, nous exécutons un programme d'inspection du système de fichiers qui va identifier toutes les cessions de privilèges réalisées ou réalisables. L'utilisation des résultats mène au graphe de privilèges représenté dans la figure III.5. Les arcs y sont étiquetés de façon à rappeler l'étape de l'analyse du système de fichiers qui a conduit à leur création. Nous reprenons ces différentes étapes ci-dessous :

(1) Trois utilisateurs (*D*, *F*, P_1) ont mal choisi leurs mots de passe. Tout utilisateur pourrait les deviner. Cela se traduit par un arc depuis le nœud INTERNE vers ces utilisateurs.
(2) *A* et *B* sont dans le fichier *.rhosts* de P_1 ; *E* et *C* sont dans celui de P_2.
(3) Pour chaque utilisateur, nous déterminons la liste de répertoires qu'il utilise dans sa variable *$PATH*. Les droits d'accès sur ces répertoires sont examinés afin d'identifier à partir de quels ensembles de privilèges on peut menacer l'utilisateur concerné. Tous les arcs portant l'étiquette "3" dans la figure III.5 matérialisent le résultat de cette recherche.
(4) L'étude des différents fichiers de configuration propres à chaque utilisateur nous révèle que *C* utilise le fichier *.xinitrc* de *B*.
(5) Nous constatons que *G* utilise un programme de filtre automatique de son courrier et ce logiciel ne lui appartient pas. Toute personne pouvant modifier ce programme peut utiliser les privilèges de *g* par le simple biais d'un message. Un nœud *filtre*, désignant les privilèges nécessaires à la modification de ce programme, est créé et relié à *G*.
(6) Ensuite, nous constatons que seul le propriétaire du programme de filtre, à savoir *B*, possède les droits d'accès nécessaires pour le modifier. Un arc est donc relié de *B* vers *filtre*. Cet arc indique que l'ensemble des privilèges de *B* inclut celui désigné par le nœud *filtre*. En effet, ce dernier ne contient que les droits nécessaires à la modification du programme installé par *B*.
(7) L'étude minutieuse de tous les fichiers exécutables ayant leur bit *setuid* positionné révèle qu'un tel programme, appartenant à P_1, utilise des exécutables que *D* peut modifier.

III.5.2 Détection d'arcs illicites

Une fois le graphe construit, la première chose à faire consiste à vérifier que les cessions de privilèges que l'on y trouve sont "normales". C'est là un problème d'interprétation laissé à l'appréciation du responsable du système. Selon les contextes d'application, on pourra, ou



1) "peut deviner le mot de passe de" ; 2) "est présent dans le .rhosts" ; 3) "peut modifier un répertoire présent dans la liste $PATH de" ; 4) "peut modifier le .xinitrc de" ; 5) "peut réaliser une attaque par le biais du courrier électronique" ; 6) "est un sur ensemble des privilèges de" ; 7) "peut modifier l'exécution d'un programme dont le setuid bit est positionné appartenant à".

Figure III.5 Exemple de graphe des privilèges

non, admettre certaines cessions de privilèges. Par exemple, certaines organisations pourraient, de manière autoritaire, décréter que l'utilisation des fichiers *.rhosts* est interdite alors que d'autres y seront favorables. Selon la philosophie d'entreprise, les actions à envisager après la construction du graphe diffèrent.

Pour en revenir à notre exemple, l'administrateur système a peu de raisons d'interdire les cessions de privilèges qu'il a découvertes. En effet :

- Les utilisateurs *A*, *B*, *D* et P_1 collaborent à un même projet et sont conscients des privilèges qu'ils s'accordent mutuellement. Il en va de même pour *E*, *C* et P_2.
- Les utilisateurs *F* et *G* sont responsables des différents logiciels nécessaires aux utilisateurs.
- Il a eu confirmation de la part de *G* que c'est bien consciemment que ce dernier utilise un logiciel dont *b* est propriétaire.
- L'utilisateur *C*, lui aussi préfère utiliser le fichier de configuration de *B*.
- Il a averti *D* et *F* que leurs mots de passe étaient mal choisis et que n'importe qui pouvait les deviner. Tous deux n'y accordèrent pas d'importance. Il se trouve confronté à la même réaction de la part des membres du projet qui utilisent le compte P_1.

Il nous faut ouvrir ici une parenthèse. En effet, on pourrait critiquer l'attitude de cet administrateur et penser qu'il devrait être plus ferme dans sa façon d'interdire les cessions de privilèges. Une telle position, plus stricte, n'est-elle pas meilleure ? Nous pensons qu'il ne nous appartient pas de prendre position pour telle ou telle vision de la sécurité. Cela nous entraînerait dans un débat philosophique trop éloigné de notre propos. Notons simplement,

comme l'ont fait certains auteurs que, dans certains cas, il est illusoire d'espérer améliorer la sécurité en privant les utilisateurs de fonctionnalités potentiellement dangereuses, mais utiles à leur travail [Brunnstein 1991][29]. Face à une contrainte qu'il ne comprend pas, ou n'admet pas, l'utilisateur tentera de la contourner et, par là même, peut exposer le système à des dangers plus grands. L'anecdote qui suit illustre notre propos.

Une organisation obligeait ses utilisateurs à changer de mot de passe chaque mois. De plus, l'utilisateur était lié par un certain nombre de contraintes dans son choix : le mot de passe choisi devait être long d'au minimum quatre caractères, dont un caractère non-alphabétique, et ne pouvait avoir été utilisé durant les six derniers mois. L'étude, *a posteriori,* des mots de passe révéla que plusieurs utilisateurs avaient simplement choisi le nom du mois en cours comme mot de passe, suivi d'un chiffre, le plus souvent 1 ou 0. Les contraintes étaient respectées, mais le résultat n'était pas meilleur pour la sécurité. On pourrait arguer du fait que les contraintes étaient mauvaises et qu'un choix plus judicieux aurait fait son effet. C'est possible, mais il faut noter que les nombreux travaux visant à aider l'utilisateur à "bien" choisir son mot de passe ne livrent pas toujours de solutions satisfaisantes [Neugent 1987][108], [Haga et Zviran 1991][63], [Spafford 1992][133].

L'important, de notre point de vue, n'est pas tant de prôner telle ou telle approche, mais, plutôt, de pouvoir vérifier concrètement qu'une approche choisie est cohérente vis-à-vis d'un objectif de sécurité. Refermons donc notre parenthèse.

III.5.3 Vérification des objectifs de sécurité

Vérifier que chaque arc a une raison d'être n'est pas suffisant, il faut également s'assurer que son existence ne peut avoir d'incidence néfaste sur les objectifs de sécurité. Par exemple, dans le cas de A, il se sait potentiellement menacé par les membres du groupe X_{admin} ainsi que par P_1. Puisque A fait confiance aussi bien à X_{admin} qu'à P_1, il s'accommode de cette situation. Cependant, si nous examinons, non plus les arcs, mais les chemins qui mènent à ce même nœud, nous constatons que tous les utilisateurs peuvent acquérir les privilèges de A. En particulier, il existe huit chemins élémentaires[1] différents[2] entre le nœud *interne* et A ! C'est-à-dire qu'il y a huit façons de propager les privilèges de A jusqu'au nœud *interne*. Rien ne permet de penser pourtant que la confiance que A possède en P_1 et X_{admin} se propage également à tout utilisateur !

Bien sûr, il existe plusieurs façons de protéger A sans l'empêcher d'accorder sa confiance à qui bon lui semble. On pourrait par exemple, supprimer les nœuds arrivant en P_1, F et G, mais nous avons expliqué que ces arcs, eux aussi, ont une raison d'être. Nous sommes donc face à une situation typique où les utilisateurs veulent bénéficier des avantages des cessions de privilèges sans en subir les conséquences néfastes. Comment résoudre ce problème ? Comment décider

1. Un chemin est dit élémentaire s'il ne passe pas deux fois par le même nœud.
2. *interne*-D-P_1-A, *interne*-D-P_1-B-C-P_2-G-X_{admin}-A, *interne*-D-P_1-B-C-P_2-F-X_{admin}-A, *interne*-D-P_1-B-*filtre*-G-X_{admin}-A, *interne*-F-*soft*$_{admin}$-P_2-G-X_{admin}-A, *interne*-F-X_{admin}-A, *interne*-F-X_{admin}-B-P_1-A, *interne*-F-*soft*$_{admin}$-P_2-G-X_{admin}-B-$P1$-A

des arcs à supprimer ? Comment convaincre[1] un utilisateur qu'il ne doit pas céder un ensemble de privilèges car cela peut porter préjudice à un tiers ?

Les réponses à ces questions passent par la nécessité de s'interroger sur la réalité du danger qui menace *A*. En effet, si les chemins *interne-*P_1*-A* et *interne-F-*X_{admin}*-A* représentent indubitablement une menace pour *A* en raison des arcs qui les composent, pouvons-nous en dire autant du chemin passant par les nœuds *interne-soft*$_{admin}$*-*P_2*-G-*X_{admin}*-B-*P_1*-A* ? Si un chemin représente une menace très faible pour *A*, il peut ne pas être nécessaire de le supprimer. *A contrario*, si un chemin est très menaçant, il faudra vraisemblablement tenter de faire porter les modifications sur un ou plusieurs de ses arcs. Si nous étudions tous les objectifs de sécurité au vu du graphe, l'identification des chemins les plus menaçants devrait nous permettre de trouver le meilleur compromis entre les modifications à apporter au graphe et les contraintes imposées aux utilisateurs.

Bien entendu, ceci implique que nous puissions mesurer le "danger" qui pèse sur *A*. Mais de quelle variable disposons-nous pour caractériser le "danger" ? Comment obtenir les différents paramètres qu'une telle évaluation va nécessiter ? Quel modèle mathématique choisir ? Nous apportons des éléments de réponse à ces questions primordiales au chapitre IV. En attendant, nous aimerions présenter une vision intuitive des résultats attendus d'une telle évaluation de la sécurité.

III.6 Discussion

III.6.1 Analogie du labyrinthe

Comparons le graphe obtenu à un labyrinthe. Les arcs y correspondent à des portes qui relient entre elles des chambres, images des nœuds du graphe. Les portes ne peuvent s'ouvrir que dans un seul sens, mais, une fois ouvertes, elles le restent. Chacune est munie d'une serrure et est transparente, découvrant la chambre avec laquelle elle communique. Les mécanismes de fermeture sont de différents types, caractérisés par des taux de défaillance variables selon les modèles.

Une des chambres recèle une cible et nous plaçons un attaquant dans une autre chambre. Ce dernier ne connaît pas la topologie du labyrinthe et doit tenter d'atteindre la cible. Il dispose, en nombre illimité, de mécanismes conçus pour ouvrir les serrures. Il n'a qu'à en appliquer un sur chaque serrure à ouvrir puis attendre que leur mécanisme de fermeture cède. Nous ne faisons pas d'hypothèse sur le comportement de cet attaquant sinon que :

- Il est doué de mémoire : il se souvient des différentes chambres où il est passé ainsi que des communications qu'elles offraient.

1. Rappelons la nécessité de convaincre l'utilisateur sans quoi nous nous exposons à d'autres problèmes que le graphe pourrait ne plus être à même d'identifier !

- Il est doué de bon sens : il ne tentera pas de forcer une serrure pour aller dans une chambre qu'il a déjà atteinte par un autre chemin.

Nous aimerions déterminer les facteurs qui feront que la cible sera plus ou moins bien protégée de cet attaquant, c'est-à-dire s'il lui sera plus ou moins facile de l'atteindre.

III.6.2 Chemin rectiligne

Supposons que l'attaquant soit dans la pièce mitoyenne de la cible et que, de plus, il n'y ait qu'une seule porte dans cette pièce. Il est clair que la protection de la cible, ou de façon duale, l'effort de l'attaquant est directement proportionnel à la qualité de la serrure.

Si l'attaquant doit traverser plusieurs pièces et que chacune ne possède qu'une seule porte de sortie, l'effort total pour atteindre la cible sera directement proportionnel au nombre de portes et à la qualité de chacune des serrures. Nous appellerons ce type de chemin un chemin rectiligne. Dans ce cas, l'évaluation peut se résumer à l'addition, par exemple, des difficultés associées à l'ouverture de chacune des serrures ; la difficulté s'assimile alors à la notion classique de longueur de chemin.

III.6.3 Chemins multiples

Rajoutons maintenant au chemin rectiligne décrit ci-dessus un deuxième chemin rectiligne, ce nouveau chemin n'améliore certainement pas la protection de la cible ! En effet, dans ce cas, l'attaquant pourra utiliser conjointement les outils qu'il a à sa disposition sur plusieurs mécanismes. Son taux de réussite est alors une fonction du taux cumulé de défaillance des deux serrures sur lesquelles s'échinent les machines démoniaques qu'il a installées pour accomplir son forfait. Son effort est inversement proportionnel au nombre de chemins qui s'offrent à lui.

III.6.4 Prépondérance du plus court chemin

Bien entendu, il est fort probable que la serrure au plus fort taux de défaillance sera celle qui fera passer l'attaquant le plus vite, et par extension, qu'il arrivera à la cible par le plus court chemin. Ceci est d'autant plus vrai que le second chemin aura une longueur significativement plus longue que l'autre. Il faut donc imaginer un système de pondération qui soit proportionnel à la longueur des chemins. De plus, cette remarque doit être appliquée de façon récursive à tous les embranchements que trouvera l'attaquant sur ces chemins.

Intuitivement, nous en concluons que la valeur finale de l'effort à dépenser doit être de l'ordre de grandeur du plus court chemin, divisée par le nombre de chemins ayant cette longueur. Les autres chemins doivent la diminuer, tout en ayant une importance moindre.

III.6.5 Conclusions

Cette analogie nous a permis d'illustrer le processus que nous désirons modéliser à l'aide du graphe des privilèges. Elle offre une vision intuitive des propriétés attendues des mesures que nous pourrions obtenir à l'aide du graphe :

(1) La sécurité augmente avec la "longueur du chemin" vers la cible.
(2) La sécurité diminue avec le nombre de chemins vers la cible.
(3) La sécurité dépend principalement du plus court chemin vers la cible.

Il est sans doute possible de définir une formule de calcul qui respecterait les propriétés énoncées ci-dessus et qui nous permettrait donc de mener à bien les tâches d'évaluation désirées. Ce n'est pas l'approche que nous avons choisie car nous avons préféré prendre du recul par rapport à la tâche d'évaluation proprement dite et adopter un formalisme mathématique qui nous permette de modéliser la logique du processus que nous avons décrit.

Le choix de ce formalisme a été guidé par la nécessité de pouvoir l'utiliser, ensuite, pour réaliser des mesures intéressantes. Il est présenté dans le chapitre qui suit et ses relations avec le graphe des privilèges sont expliquées. Nous montrons comment il permet de calculer l'effort que doit dépenser un attaquant pour atteindre une cible donnée. Le problème de l'interprétation des résultats est, quant à lui, envisagé dans le chapitre V.

Lorsque nous calculons la difficulté pour un attaquant de violer un objectif de protection, nous faisons abstraction du processus qui l'a amené à choisir ce système en particulier. S'interroger sur la probabilité que le système étudié soit soumis à une attaque est, certes, une question intéressante, mais qui sort de notre propos. Il appartient au responsable du système, au vu des mesures de sécurité, d'initier les actions qui s'imposent en tenant compte de l'hostilité supposée de l'environnement dans lequel il se trouve.

III.7 Conclusions

Dans ce chapitre nous avons présenté un exemple d'application du graphe des privilèges. Nous avons ainsi pu illustrer la notion de fautes intentionnelles non malveillantes et la nécessité de recourir à une méthode de prévision de fautes pour gérer à la fois des impératifs de sécurité et de confort d'utilisation.

Nous avons montré que le graphe était une image du système réel dans la mesure où il pouvait en être extrait automatiquement, sans l'intervention d'évaluateurs externes. Nous avons expliqué les limites des méthodes de détection de fautes et mis en lumière intuitivement les propriétés attendues du mécanisme d'évaluation requis pour réaliser la prévision de fautes [Dacier 1994][41].

Dans le chapitre suivant, nous nous attachons à expliquer les variables sur lesquelles va porter l'évaluation, comment les obtenir et quel formalisme mathématique utiliser pour obtenir les mesures désirées.

CHAPITRE IV *Une méthode d'évaluation quantitative de la sécurité*

Préambule

Dans les chapitres qui précèdent nous avons vu que :

- La tendance en matière de critères internationaux met en exergue la nécessité de développer des méthodes de prévision de fautes.
- Les travaux en analyse des risques doivent être enrichis de l'expérience acquise en matière de modélisation formelle de la sécurité.
- Parmi ceux-ci, un nouveau modèle, le graphe des privilèges se révèle à la fois expressif et efficace pour étudier des systèmes réels.
- Un exemple d'application de ce modèle souligne la nécessité de disposer d'une méthode de prévision de fautes pour satisfaire au mieux les objectifs de sécurité.

Dans ce chapitre, nous définissons une méthode de prévision de fautes basée sur le graphe des privilèges. Dans un premier temps, nous explicitons les variables utilisées, nous justifions leur choix et précisons leur sens.

Ensuite, nous présentons le formalisme choisi pour représenter le processus mené par un attaquant éventuel. Nous montrons comment transformer le graphe des privilèges en un réseau de Petri, en tenant compte d'hypothèses sur le comportement des attaquants : la mémoire de l'attaquant, l'expérience acquise et la coalition avec d'autres assaillants sont envisagées.

Nous nous intéressons ensuite à la façon dont ce modèle peut être utilisé pour obtenir des mesures caractérisant l'évolution de la sécurité d'un système dans son environnement opérationnel. Nous passons en revue les différents résultats intéressants ainsi que la manière de les obtenir. Nous montrons qu'ils respectent bien les propriétés intuitives identifiées au chapitre précédent.

Enfin, nous situons les limites et avantages de cette méthode. Nous reprenons les hypothèses qui mènent à l'obtention des résultats et le cadre précis dans lequel il faut les interpréter. Nous explorons également les domaines d'application de la méthode.

IV.1 Les variables utilisées

IV.1.1 Introduction

Avant d'expliquer *comment* nous évaluons, il est nécessaire de préciser *ce que* nous évaluons. C'est pourquoi nous présentons dans les pages qui suivent les deux variables choisies pour évaluer la sécurité des systèmes informatiques. Ces deux variables sont le **temps** et l'**effort**. En effet, certaines attaques demandent peu d'effort de la part d'un attaquant, mais ne peuvent être réalisées avec succès que moyennant une très longue période de temps ; c'est le cas pour la plupart des chevaux de Troie. Inversement, certaines attaques sont très complexes, mais ont un effet immédiat. Toutes les autres combinaisons sont également possibles. Il semble donc, *a priori*, difficile de regrouper ces deux paramètres en une seule variable sans perdre de l'information pertinente.

IV.1.2 Définitions

A. Le temps

La sécurité d'un système est directement proportionnelle au temps nécessaire pour mener à bien une attaque. Ceci repose sur les considérations suivantes :

(1) La probabilité que la présence d'un intrus soit détectée et son attaque interrompue est d'autant plus grande qu'il reste longtemps au sein du système.
(2) La probabilité qu'une attaque soit choisie par un assaillant est d'autant plus faible que le délai supposé nécessaire à la réussite de cette attaque est important.

Cette variable "temps" est particulièrement adéquate pour évaluer l'influence du risque induit par les chevaux de Troie. Pour le voir, reprenons l'exemple de la méthode d'attaque *via* le fichier *.xinitrc* décrite au chapitre précédent (section III.3.2, page 60). Supposons que *a* puisse écrire dans le fichier *.xinitrc* de *b*. Ce dernier est donc menacé par *a*. Cependant, si *b* n'utilise jamais *X-window* (parce qu'il ne dispose pas d'un écran graphique, par exemple), l'attaque n'a aucune chance d'aboutir. Le risque est cependant présent, mais la réalisation fructueuse de la méthode demandera un temps infini. La probabilité de réussir l'attaque dans un intervalle de temps fixé est directement proportionnelle au taux d'utilisation du programme.

D'autres types d'attaque reposent également sur la notion de temps. Ainsi en est-il des attaques par le biais des mots de passe qui obligent l'attaquant à examiner un nombre très important de combinaisons afin de parvenir à ses fins. L'espace de définition auquel appartient le mot de passe, c'est-à-dire les caractères qui le composent, détermine la probabilité de le deviner dans un espace de temps donné [Riddle et al. 1989][115], [Spafford et Garfinkel 1991][132].

Il existe encore d'autres attaques, comme l'étude des données qui transitent sur le réseau, l'analyse du contenu de la mémoire, l'inspection régulière des écrans de tiers, etc. qui ne requièrent que la seule intervention de l'attaquant, mais qui ne se traduisent pas pour autant par

un succès immédiat. L'influence que joue le temps dans ces différentes classes d'attaque explique que nous ayons tenu à l'étudier de manière autonome. Cependant, cette seule variable ne suffit pas à caractériser la sécurité d'un système donné.

Notons que la sécurité croît avec le temps que *doit* mettre l'assaillant pour parvenir à ses fins, mais, que de façon duale, l'insécurité croît avec le temps que *peut* passer l'assaillant à essayer diverses attaques.

B. L'effort

La sécurité est directement proportionnelle à l'effort que demande à l'assaillant la mise en œuvre d'une attaque.

En effet, certaines attaques exigent des connaissances très pointues sur le système, que peu d'attaquants possèdent. Par exemple, modifier le contenu de certaines cases mémoire peut permettre de réaliser des intrusions, mais les détails de cette attaque sont peu connus et, de plus, spécifiques à chaque système d'exploitation visé (par exemple, l'attaque des systèmes Unix via le *daemon fingerd* décrite dans [Spafford 1988][130]).

D'autres attaques ne peuvent être réalisées que moyennant la mise en œuvre d'un matériel sophistiqué, si complexe ou si coûteux qu'il n'est pas à la portée du premier venu. Par exemple, l'étude des rayonnements électromagnétiques issus des écrans et claviers est une façon de détecter des mots de passe, mais elle exige un équipement qu'il n'est pas facile de se procurer.

D'autres, enfin, nécessitent un travail fastidieux de mise en œuvre qui en découragera plus d'un. Créer, dans un répertoire judicieusement choisi, un très grand nombre de fichiers portant le nom de commandes usuelles mal tapées fait partie, par exemple, de cette catégorie.

IV.1.3 Discussion

Idéalement, le choix d'un ensemble de variables significatives devrait se baser sur l'analyse d'intrusions connues —réussies ou non— et des paramètres qui ont influencé leurs déroulements. Malheureusement, de telles données n'existent pas ou du moins, si elles existent, ne sont pas à notre disposition.

Il est toutefois possible de s'inspirer des attaques qui eurent le plus de retentissement au sein de la communauté internationale. Il en existe quelques-unes décrites dans la littérature, mais leur rareté ne permet pas de considérer qu'elles représentent un échantillon représentatif : [Reid 1986][114], [Stanley 1986][134], [Stoll 1988][136], [Rochlis et Eichin 1989][117], [Seeley 1989][124], [Spafford 1988][130], [Cheswick 1991][32], [Stanley 1991][135], [Bellovin 1992][16].

Plus intéressants pour notre propos sont les articles où diverses personnalités, considérées comme expertes dans le domaine de la sécurité, donnent leur opinion sur les techniques généralement utilisées par les intrus et tentent de formuler un portrait type des assaillants à

redouter [Denning 1990][44]. Selon elles, la plupart des attaquants utilisent des méthodes bien connues. De plus, une tendance se dessine actuellement, dans certains milieux, en faveur de la diffusion la plus large possible des techniques d'intrusion afin de sensibiliser les responsables de sécurité. Il en résulte que "pirater" aujourd'hui revient plus à essayer un ensemble de recettes toutes prêtes qu'à véritablement inventer de nouvelles techniques. Certes, il ne faut pas nier l'existence de quelques érudits, mais ils ne représentent qu'une faible minorité de la population des attaquants potentiels.

Cette vision des choses se trouve confirmée par une étude menée conjointement par une équipe de l'université de Londres (Royaume-Uni) et de Chalmers (Suède), à laquelle nous avions été brièvement associés dans le cadre du projet européen PDCS-2. Ce travail, suggéré dans [Littlewood et al. 1991][96] et dont la mise en œuvre est décrite dans [Olovsson et al. 1993][110], avait pour but d'obtenir une estimation de l'effort moyen à dépenser par des attaquants pour pénétrer dans un système. Pour cela, reprenant l'idée des *Tiger Teams* ([Baird et al. 1987][41], [Frenkel 1987][53], [Herschberg 1988][67]) les responsables du projet demandèrent à un ensemble d'étudiants, triés sur le volet, de réaliser des intrusions dans un système précis et de rendre compte de leurs tentatives (temps passé, moyens mis en œuvre, etc.). Tous ces pirates improvisés, étudiants en informatique, faisaient de surcroît partie d'un club d'informatique. Leur intérêt pour les ordinateurs et leurs connaissances étaient donc incontestablement au-dessus de la moyenne. Cependant, au vu des résultats, il s'avère qu'aucun n'a eu recours à des méthodes d'attaque qui ne fussent connues au départ par les examinateurs. La plupart d'entre eux avaient trouvé leurs sources d'inspiration aux mêmes endroits, à savoir dans les livres les plus populaires sur la sécurité UNIX ([Wood et Kochan 1987][146], [Curry 1990][35], [Spafford et Garfinkel 1991][132]), certains forums de discussion (*comp.unix.security*, etc.) ou encore des serveurs de données [Archimbaud 1993][8]. De plus, dans cette manne d'attaques à leur disposition, ils avaient délaissé les plus compliquées au profit des simples, les lentes au profit des rapides.

En plus d'une justification à mettre en œuvre une méthode de prévision de fautes basée sur des attaques connues, nous trouvons dans ce travail une justification à notre choix de distinguer deux variables pour classer de façon orthogonale les critères de difficulté et de rapidité. En effet, l'idée des initiateurs de l'expérience était d'analyser le comportement des attaquants de façon globale et d'en déduire les caractéristiques de l'effort à mettre en œuvre pour venir à bout d'un système informatique. Cette approche fit long feu, en raison non seulement de la faible diversité des attaques observées, mais également en raison de la difficulté d'inclure sous un même terme des notions aussi différentes que la difficulté à découvrir, comprendre et mettre en œuvre une attaque, d'une part et le temps nécessaire pour qu'elle soit couronnée de succès, d'autre part.

Il faut également noter que notre approche s'appuie avant tout sur la connaissance des fautes opérationnelles connues présentes au sein du système. Ce fut également la démarche adoptée par les apprentis pirates durant l'expérience citée. Nous pourrions rechercher des fautes de conception du système (des bogues) encore inconnues en incluant dans notre méthode

certaines techniques de pénétration systématique [Gupta et Gligor 1992][62], mais nous en sommes abstenus jusqu'ici car nous avons estimé que le gain marginal était trop faible, à ce stade de notre travail.

IV.1.4 Obtention des valeurs

A. Remarque préliminaire

Toutes ces remarques nous confirment dans l'idée selon laquelle il est possible d'identifier la plupart des attaques que vont utiliser les assaillants. En effet, il est possible à un spécialiste en sécurité, et *a fortiori* à une équipe, non seulement de recenser les attaques connues sur un système, mais également d'estimer la publicité qu'elles ont reçues dans la communauté internationale.

C'est la voie que nous considérons la plus prometteuse pour le succès de la méthode que nous proposons. L'ensemble de règles que nous avons constitué aujourd'hui sur base de notre expérience personnelle est, certes, appréciable, mais demande à être complété. Il serait présomptueux de prétendre connaître l'ensemble des règles d'attaque des systèmes UNIX. Cependant, la réalisation d'un ensemble complet peut être effectuée en réunissant l'expertise de spécialistes du domaine. Une telle démarche avait d'ailleurs déjà été envisagée en 1992 afin de définir une nouvelle version du logiciel *COPS* [Farmer et Spafford 1990][47]. A l'époque, un groupe de discussion, dont nous faisions partie, avait été créé afin de définir les spécifications d'un logiciel qui profiterait de l'expertise des différents membres de la liste. Ce logiciel, comme *COPS*, aurait eu comme but d'identifier toutes les attaques possibles au sein d'un système UNIX classique. Les aléas professionnels de chacun en décidèrent autrement. *SATAN*, puisque c'est son nom, devrait cependant voir le jour d'ici la fin de l'année prochaine, grâce aux efforts combinés de Dan Farmer (concepteur de *COPS*) et de Wietse Venema [Farmer et Venema 1993][48].

Il n'en demeure pas moins qu'il est possible de définir un ensemble quasi complet de règles de construction du graphe des privilèges. Il nous faut à présent expliquer comment nous comptons évaluer les variables "temps" et "effort" caractérisant chaque attaque.

B. Le temps

Le temps joue un rôle important dans toutes les techniques d'attaque utilisant des chevaux de Troie. Il peut sembler fort difficile, voire même impossible, de déterminer le délai que l'attaquant doit attendre avant de voir une de ses victimes tomber dans un piège. Cependant, si nous connaissons les commandes les plus utilisées par chaque utilisateur, nous pouvons estimer la *probabilité* qu'il active un cheval de Troie dans un certain intervalle de temps. En effet, il est alors possible de déterminer un taux de succès de l'attaque, fonction du comportement habituel de l'utilisateur cible. Par exemple, le taux de succès d'une attaque basée sur le fichier *.xinitrc* d'un utilisateur est directement proportionnel au taux d'utilisation du logiciel *X-window* par celui-ci.

Il existe des logiciels qui permettent de déterminer des profils statistiques des utilisateurs, fonctions de leur comportement et de leurs interactions avec le système. Ils ont été développés dans le cadre des techniques de détection d'intrusion. Une intrusion y est définie comme une déviation entre le profil statistique d'un utilisateur et le profil de sa session en cours. L'expérience acquise par ces systèmes, et, en particulier, par *IDES*, le plus connu d'entre eux, montre qu'il est possible de réaliser un tel profil pour chaque utilisateur ([Lunt et al. 1990][98], [Javitz et Valdez 1991][79]).

IDES a depuis peu un successeur, *NIDES* [Anderson et al. 1993][5], dont le champ d'application est précisément celui des stations de travail UNIX utilisées dans un réseau. Il est donc envisageable d'utiliser les profils statistiques fournis par *NIDES* [Javitz et Valdez 1994][80] pour en déduire les taux de réussite des attaques présentes dans le graphe. Une telle approche nous permet de quantifier nos variables de temps de la façon la plus objective possible, sans devoir faire intervenir l'expertise d'un quelconque évaluateur. De plus, les profils statistiques évoluent au fil du temps et offrent une vision dynamique du comportement des utilisateurs et, par suite, de la sécurité du système. Ils nous permettent de travailler avec des valeurs toujours actualisées et non pas issues d'entretiens épisodiques avec les utilisateurs.

Quant aux autres techniques d'attaque qui requièrent un temps important pour réussir, elles nécessitent soit l'attente d'un événement rare, soit l'examen d'un grand ensemble de solutions. Dans le premier cas, *NIDES* peut à nouveau offrir une réponse à nos attentes. Pour le second, en revanche, la probabilité de réussite doit être calculée en fonction de la dimension de l'espace de recherche, qui lui aussi est fini et connu (recherche d'un mot de passe, par exemple).

C. L'effort

L'estimation de l'effort est plus délicate et *NIDES* ne nous est d'aucun secours. La distribution de probabilité, en fonction de l'effort fourni, d'accaparer un ensemble de privilèges doit refléter l'estimation que nous avons de ce que cette attaque peut être mise en œuvre par une population plus ou moins grande d'attaquants. Ceci suppose que nous soyons capables de définir une relation d'ordre entre les différentes attaques et que nous puissions dériver des taux de réussite pour chaque attaque, en fonction de l'effort dépensé. Un taux de réussite important correspondrait à une attaque "facile" ; une valeur faible à une attaque "difficile".

Une telle relation n'existe pas, mais, cependant, l'expérience montre que les attaques les plus connues sont, apparemment, les plus utilisées. De plus, parmi celles-ci, les plus faciles à comprendre et à mettre en œuvre, en termes de travail pour l'attaquant, sont prioritairement choisies. Si nous tentons d'établir un ensemble restreint de classes d'attaques —trois ou quatre niveaux nous semblent un maximum— il n'est pas du tout déraisonnable de trouver, au sein d'un groupe d'experts, un consensus sur les attaques à y placer. Nous proposons, à titre d'exemple, une classification à quatre niveaux :

(1) Au niveau le plus bas, nous mettons l'ensemble des attaques connues et fréquemment utilisées par le passé au cours d'intrusions répertoriées.

(2) Au deuxième niveau on retrouverait les attaques connues, mais dont l'utilisation n'a été constatée que rarement.
(3) Le troisième niveau reprend les attaques très sophistiquées dont on connaît l'existence théorique, mais dont aucune utilisation n'a été recensée.
(4) Le quatrième et dernier niveau reprend les attaques sophistiquées, non utilisées et qui, de plus, exigent de l'attaquant d'être géographiquement situé aux abords du système.

Le fait de n'avoir que peu de niveaux n'est pas du tout en contradiction avec notre méthode. Au contraire, comme nous allons le voir, l'évaluation a pour principal but d'offrir des ordres de grandeur. Dans ce sens, un faible nombre de classes d'attaques ne peut que faciliter l'interprétation des résultats.

D. Résumé

Nous attribuons donc à chaque arc, c'est-à-dire à chaque attaque possible, deux taux de transition. Le premier représente le temps moyen nécessaire à la réussite de l'attaque ; il est fonction de l'activité de la cible et du type d'attaque. Le second taux caractérise l'effort moyen à dépenser pour mener à bien l'attaque ; il est basé sur la classification en quatre niveaux, fonction de l'attaque, établie dans la section ci-avant.

IV.2 Le modèle de représentation

IV.2.1 Introduction

Jusqu'à présent, nous sommes resté évasif sur la définition précise d'un chemin dans le graphe de privilèges. Nous allons combler cette lacune en deux temps. Tout d'abord, nous allons identifier, sur un exemple simple extrait du chapitre précédent, tous les chemins qu'un attaquant peut suivre pour atteindre une cible. Ceci nous permet de mettre clairement en évidence les hypothèses que nous faisons sur le processus d'attaque qui l'anime. Ensuite, nous montrons comment il est possible de modéliser la logique de ce processus en transformant le graphe en un réseau de Petri [Dacier et al. 1993][40]. Nous donnons les règles de transformation et indiquons les avantages importants de cette façon de procéder.

IV.2.2 Identification des scénarios d'attaques

A. Hypothèses d'attaques

La figure IV.1 est extraite du graphe des privilèges présenté dans le chapitre précédent (figure III.5, page 67). Nous avons changé la numérotation des arcs afin que chacun d'entre eux soit identifié par un numéro unique auquel nous puissions faire référence dans le corps du texte.

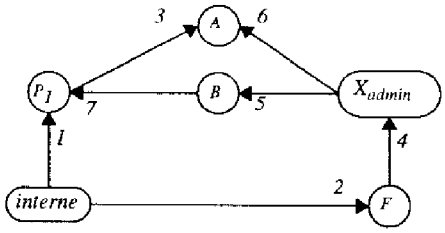


Figure IV.1 Exemple de graphe de privilèges

Pour dérouler tous les scénarios que l'utilisateur *interne* peut suivre pour arriver jusqu'à l'utilisateur *A*, nous devons faire des hypothèses sur le comportement de cet attaquant. Face à l'inexistence de profil d'attaque, indiquée précédemment, nos hypothèses ne peuvent que reposer sur le bon sens. En l'occurrence, nous supposons que :

(1) Les attaquants n'ont pas la connaissance, *a priori*, du graphe complet de privilèges. Ils voient toutes les attaques qu'ils peuvent mettre directement en œuvre et uniquement celles-là.
(2) Les attaquants sont doués de mémoire. Ils se souviennent des ensembles de privilèges qu'ils ont déjà accumulés au cours de leur attaque.
(3) Les attaquants sont doués de raison. Ils ne mettent pas en œuvre une attaque qui leur rapporterait un ensemble de privilèges dont ils disposent déjà.

Les deux dernières hypothèses ne devraient pas poser de problèmes au lecteur. La première, quant à elle, est justifiée par le fait que la construction du graphe complet nécessite la détention de l'union de tous les ensembles de privilèges. Si l'attaquant dispose de ces privilèges, il n'a aucune raison de construire le graphe ! Enfin, mentionnons également qu'un scénario d'attaque prend fin dès qu'une cible est atteinte.

Examinons les différents scénarios que peut suivre l'utilisateur *interne* pour obtenir les privilèges de *A* :

- Au départ, il voit les attaques 1 et 2.
- Supposons qu'il opte pour l'attaque 1 ; à l'étape suivante, il peut réaliser l'attaque 3 ou l'attaque 2 ; en effet, en raison de la première hypothèse, l'attaquant est doué de mémoire, il se souvient des attaques qu'il pouvait faire et qu'il n'a pas réalisées.
- Si, à ce stade, il choisit l'attaque 3, il arrive à la cible et le processus d'attaque cesse.
- En revanche, s'il choisit l'attaque 2 après l'attaque 1, il se retrouve dans la même situation que s'il avait d'abord choisi l'attaque 2, puis la 1. De là, deux choix s'offrent lui : l'attaque 4 ou l'attaque 3, et ainsi de suite ...

B. Impact sur le graphe

Nous avons représenté dans la figure IV.2 l'ensemble des états[1] qui caractérisent le processus d'attaque. Le processus change d'état lorsqu'un nouvel ensemble de privilèges a été acquis. Les arcs qui relient les états entre eux peuvent donc être étiquetés par des numéros qui font

1. Par souci de concision, X_{admin} y est représenté par *X* et *interne* par *I*.

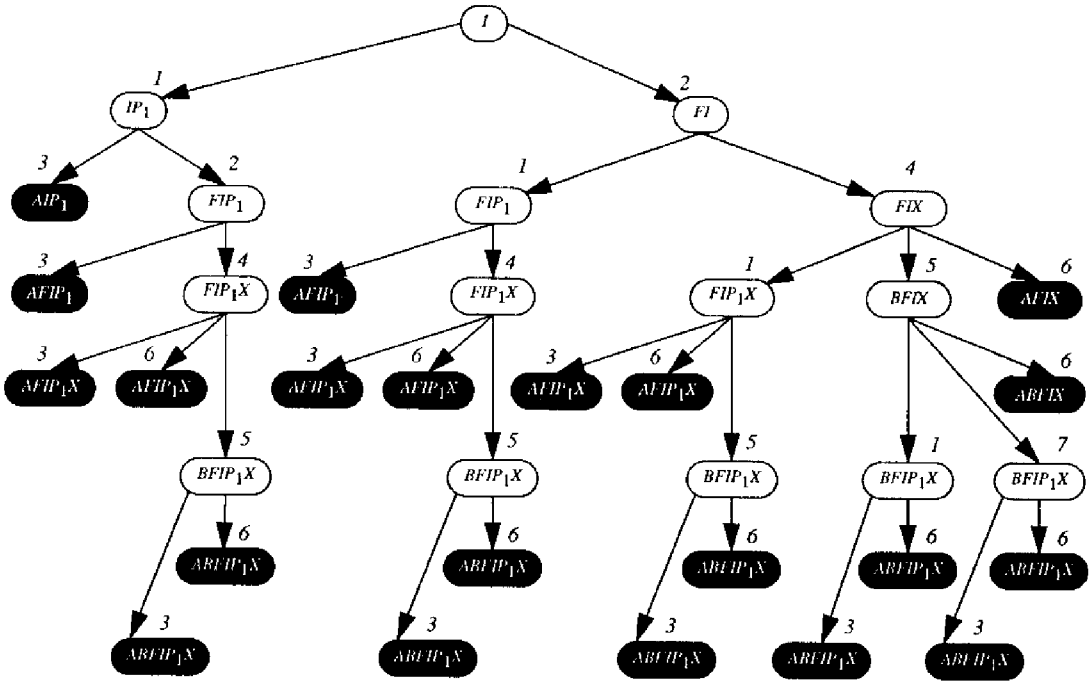


Figure IV.2 Les différents états et scénarios du processus d'attaque

référence aux méthodes mises en œuvre pour passer d'un état à l'autre. Lorsque le processus passe d'un état à un autre, l'ensemble total de privilèges qu'il détient après la transition est toujours un sur-ensemble de celui qu'il avait auparavant. Les boîtes noires représentent les états dans lesquels la cible a été atteinte.

Cette représentation n'est pas la plus concise qui soit puisque certains états y apparaissent plusieurs fois en raison de l'hypothèse de mémoire évoquée auparavant (page 80). Ainsi en est-il par exemple de l'état FIP_1 qui peut être atteint aussi bien par la séquence d'attaques 1-2 que par la séquence 2-1.

Fort de cette remarque, nous pouvons regrouper tous les états identiques car ils correspondent aux mêmes ensembles de privilèges. Cela nous donne le graphe représenté dans la figure IV.3. Notons que lorsque deux transitions différentes de la figure IV.2 relient deux mêmes nœuds, nous n'en avons plus représenté qu'un seul, tout en indiquant dans son étiquette les différentes transitions possibles, mutuellement exclusives("3 ou 6", par exemple). L'attaquant atteint sa cible s'il atteint l'un quelconque des ensembles de privilèges réunis dans la boîte grisée au bas du graphe de la figure IV.3. Cette boîte reprend l'ensemble des états caractérisant la défaillance du service de sécurité.

Cette représentation plus synthétique permet de mieux voir que, en fonction du chemin parcouru par l'attaquant, certaines transitions peuvent être inhibées. Nous le constatons en examinant les transitions qui sortent de l'état $BFIP_1X$. Seules y sont mentionnées les attaques 3 et 6 qui mènent à la cible. L'attaque 7 n'est pas envisagée car elle mène à P_1 dont l'attaquant a

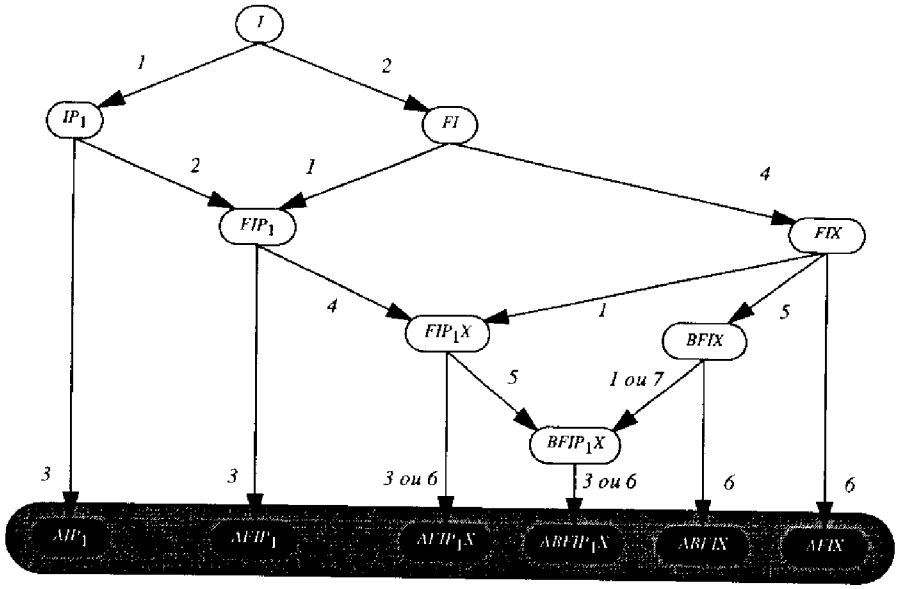


Figure IV.3 Représentation agrégée des scénarios d'attaque.

déjà acquis les privilèges. C'est la conséquence de notre deuxième hypothèse où nous stipulions que l'attaquant était doué de raison.

C. Synthèse

En résumé, reprenons les trois hypothèses et la façon dont elles apparaissent sur ce graphe d'états :

(1) Dès que l'attaquant arrive à la cible, le processus s'arrête. Il a atteint un état terminal. Ces états sont représentés par des ovales au fond sombre dans la figure IV.3.

(2) L'attaquant étant doué de mémoire, certains scénarios ne diffèrent que par l'ordre des attaques envisagées. Ceci se traduit sur le graphe par certains états qui peuvent être atteints à partir de plusieurs autres. De plus, puisqu'il se souvient des ensembles acquis, chaque transition ne peut le mener que dans un état où il possédera un sur-ensemble des privilèges qu'il possède dans l'état où il est. Aucun cycle ne peut donc exister dans le graphe des états.

(3) L'attaquant est doué de bon sens. Aucun chemin ne passe deux fois par le même nœud du graphe de privilèges ce qui implique qu'aucune transition dans le graphe d'états ne quitte un état pour revenir dans le même.

IV.2.3 Transformation en un réseau de Petri

Les hypothèses énoncées dans le point *A* de la section IV.2.2 décrivent la logique du processus d'attaque. Le graphe d'états en est la matérialisation. C'est sur ce graphe que nous allons pouvoir appliquer un modèle mathématique d'évaluation. La mise en œuvre de ce modèle n'est possible que si nous avons un moyen simple de construire le graphe d'états. L'utilisation des réseaux de Petri permet d'atteindre un tel objectif. Dans ce but, nous transformons le

graphe des privilèges en un réseau de Petri qui inclut la logique du processus d'attaque. Nous montrons que le graphe des marquages de ce réseau est équivalent au graphe d'états décrivant les scénarios d'attaque.

A. Le canevas de base

L'idée qui nous mène est la même que celle qui nous a permis de proposer une solution au problème de la collaboration dans le modèle *TG* (section II.4.3, page 41). Dans cette section, nous définissons une méthode pour transformer le graphe des privilèges en un réseau de Petri ayant les propriétés suivantes :

(1) La propagation des jetons dans le réseau représente l'évolution de l'attaquant le long d'un chemin.
(2) Le graphe des marquages du réseau doit être identique au graphe d'états qui a permis d'identifier les scénarios induits par le graphe des privilèges.

La règle de transformation principale est représentée dans la figure IV.4. A tout nœud Y du graphe correspondent deux places Y et Y_g dans le réseau de Petri. La première est appelée *place principale* et la seconde *place de garde*. Chaque arc du graphe des privilèges est remplacé par une transition ayant les caractéristiques suivantes :

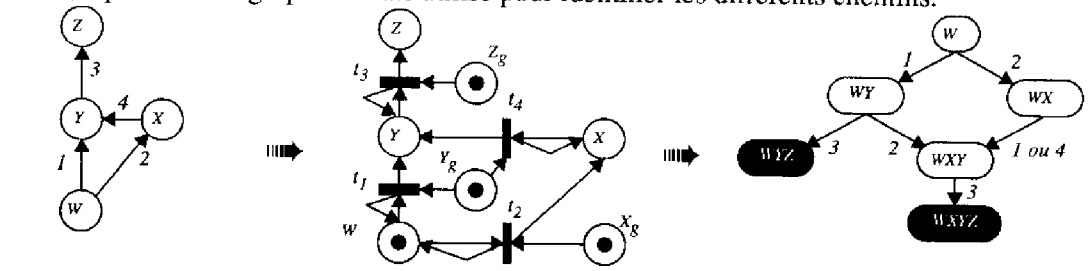
- La place X, correspondant au nœud de départ de l'arc, est reliée à la transition en entrée et en sortie.
- La place Y correspondant au nœud d'arrivée de l'arc n'est reliée à la transition qu'en sortie.
- La place Y_g n'est reliée à la transition qu'en entrée. Cette place possède un jeton dans le marquage initial.

Figure IV.4 Le canevas de base

Une fois tous les arcs transformés de la sorte, nous mettons un jeton dans les places correspondant aux nœuds des attaquants et identifions les places qui correspondent aux cibles. Ensuite, nous calculons le graphe des marquages en veillant à ce que tout état dans lequel le marquage d'une des places cibles est supérieur à zéro soit considéré comme un état terminal. Ceci nous permettra de tenir compte de l'hypothèse selon laquelle le processus d'attaque s'arrête dès que la cible est atteinte. Quant aux autres hypothèses, nous allons montrer sur quelques exemples comment cette simple transformation en tient compte.

B. Mémoire et logique de l'assaillant

Dans l'exemple décrit dans la figure IV.5, nous considérons que *W* est l'*attaquant* et *Z la cible*. L'identification des chemins que peut suivre l'assaillant, selon nos hypothèses de travail, donne quatre séquences différentes : 1-3, 2-4-3, 1-2-3 et 2-1-3. Sur la même figure, nous avons représenté le réseau de Petri, correspondant à ce graphe des privilèges, obtenu par l'application du canevas décrit ci-dessus. Nous avons également représenté le graphe des marquages de ce réseau, équivalent au graphe d'états utilisé pour identifier les différents chemins.



(a) graphe de privilèges *(b) réseau de Petri* *(c) graphe d'états*

Figure IV.5 Mémoire de l'assaillant

i) Mémoire

L'existence des chemins 1-2-3 et 2-1-3 est due à la mémoire de l'attaquant. Au fur et à mesure de sa progression, il se souvient des attaques qu'il n'a pas encore exploitées. Au niveau du réseau de Petri, cette mémoire se matérialise très simplement par le fait que toutes les places principales qui sont des places d'entrée de transition sont également des places de sortie. Il en résulte que si un jeton se trouve dans une place principale à un instant *t*, il y sera toujours à l'instant *t+dt*, pour tout *dt*. En d'autres termes, le jeton reste disponible pour tirer toute autre transition à laquelle serait reliée cette place principale.

ii) Bon sens

Nous pouvons remarquer l'absence des chemins 1-2-4-3 et 2-4-1-3. Ceci est dû à l'hypothèse de bon sens sur le comportement de notre attaquant. Puisque nous supposons qu'il ne tentera pas d'atteindre une cible par deux attaques différentes, les transitions 1 et 4 sont mutuellement exclusives.

Ceci se transcrit dans le réseau de Petri par l'utilisation des *places de garde*. A toute place principale[1] correspond une place de garde contenant initialement un jeton. Cette place sert de place d'entrée pour *toutes* les transitions pour lesquelles la place principale — à laquelle elle correspond — est une place de sortie : ainsi pouvons-nous observer sur la figure IV.5 qu'une même place de garde (Y_g)sert aux transitions t_1 et t_4. Cette place nous assure que nous ne pourrons tirer dans une même séquence t_1 et t_4.

1. Dans le graphe des privilèges, si le nœud auquel correspond la place principale ne possède que des arcs sortants, alors la place de garde, dans le réseau de Petri, est inutile.Dans notre exemple, c'est le cas pour le nœud *W*.

C. *Hypothèses complémentaires*

Les réseaux de Petri offrent un formalisme simple et adéquat qui permet d'enrichir le modèle si, dans un contexte d'application particulier, il s'avérait nécessaire d'adopter d'autres hypothèses. Nous en relevons deux : (i) sélection des attaques envisageables et (ii) expression de règles de construction plus complexes.

i) Sélection des attaques envisageables

Une fois le réseau de Petri construit, nous pouvons imaginer vouloir l'utiliser en n'envisageant qu'un sous-ensemble de toutes les attaques possibles. Il peut être utile, par exemple, de retirer du graphe toutes les attaques qui exigent des moyens matériels importants comme, par exemple, l'étude des rayonnements électromagnétiques. Il est certes possible d'extraire du réseau toutes les transitions, places et arcs impliqués par ce type d'attaque, mais une solution beaucoup plus simple existe qui illustre la souplesse d'utilisation des réseaux de Petri.

Pour cela, nous définissons une place A_C pour chaque classe d'attaques C que nous pourrions éventuellement supprimer du réseau. Cette place est la seule place d'entrée d'une transition (t_c) qui a autant de places de sorties qu'il y a de transitions dus à cette classe d'attaques dans le réseau. La figure IV.6 illustre cette construction, nous y supposons que les attaques de X vers Y et de Y vers Z appartiennent à la même classe. Si nous mettons un jeton dans la place A_C la classe d'attaques C est prise en considération dans le réseau ; dans le cas contraire, toutes les attaques de ce type sont virtuellement retirées du réseau.

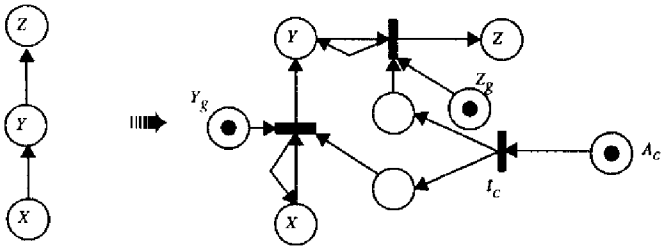


Figure IV.6 Sélection des classes d'attaques

ii) Expression de règles plus complexes

Il est intéressant de remarquer que l'utilisation des réseaux de Petri permet de relâcher certaines hypothèses que nous avions formulées lors de la construction du graphe des privilèges. Par exemple, nous pouvons maintenant envisager le cas d'intersections non vides entre les ensembles de privilèges cédés et ceux intervenant dans les conditions d'activation des règles. Dans la figure IV.7 , nous avons représenté le fait que W doit acquérir à la fois les

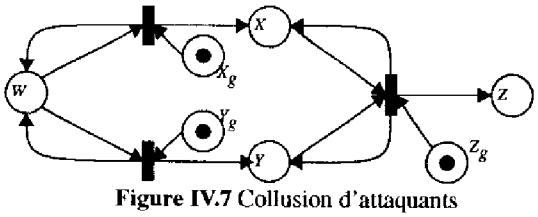


Figure IV.7 Collusion d'attaquants

ensembles X et Y pour pouvoir acquérir l'ensemble Z. Cette représentation indique également que les détenteurs des privilèges de l'ensemble X et de l'ensemble Y ne peuvent obtenir l'ensemble Z qu'en collaborant. Cependant, comme nous l'avons dit précédemment, nous n'avons pas ressenti le besoin d'exprimer de telles règles dans le cadre d'application que nous avons choisi. Aussi ne traiterons-nous plus, par la suite, de telles représentations.

IV.2.4 Conclusions

Le choix de la représentation par réseau de Petri offre un moyen simple d'identifier les différents chemins du graphe de privilèges. De plus, ce modèle permet d'enrichir le modèle initial d'hypothèses complémentaires et, si le besoin s'en faisait sentir, d'inclure de nouvelles règles difficilement exprimables au niveau du graphe de privilèges.

IV.3 Le modèle mathématique pour l'évaluation

IV.3.1 Introduction

Nous avons précisé la notion de scénarios (section IV.2.2) et montré comment leur identification était possible grâce à l'utilisation d'un réseau de Petri (section IV.2.3), nous pouvons nous interroger sur la façon de réaliser l'évaluation elle-même. Nous avons abordé le problème des taux de transition liés à chaque arc du graphe (section IV.1.2) et expliqué ainsi comment estimer la probabilité de franchir une transition dans un intervalle de temps donné (section IV.1.4). A présent, nous souhaiterions en déduire le temps et l'effort moyen nécessaires à un attaquant pour atteindre une cible.

IV.3.2 Les distributions choisies

Le problème du choix des distributions de probabilité va, bien entendu, conditionner les résultats obtenus. Deux aspects importants entrent en ligne de compte. Tout d'abord, il faut trouver les distributions qui sont cohérentes avec notre objectif et dont les paramètres peuvent être aisément définis. Ensuite, puisque nous avons toujours eu à cœur de définir une méthode qui soit pragmatique, il faut que ces distributions se prêtent aux tâches d'évaluation auxquelles nous les destinons. Ces réflexions nous ont conduit, dans l'état actuel de nos travaux, à utiliser des distributions exponentielles décroissantes à taux constant ; ceci revient à supposer que le processus d'attaque est markovien. Nous justifions ici, brièvement, les raisons de ce choix en utilisant la variable "temps". Le même raisonnement est, bien sûr, applicable à la variable "effort".

Dans notre modèle, la probabilité de franchir une transition en fonction du temps s'exprime de la façon suivante : $P(t) = 1 - exp(-\lambda \times t)$. La probabilité de passer la transition est nulle à l'instant zéro et tend vers un à l'infini. Ce choix induit les conséquences suivantes sur notre modèle :

(1) Si un chemin existe vers la cible, l'attaquant finira toujours par le trouver (peut-être après un temps proche de l'infini).
(2) Le taux de réussite d'une attaque est constant dans le temps[1].
(3) Le temps moyen nécessaire pour franchir une transition est égale à $1/\lambda$.

Cette dernière remarque est particulièrement intéressante dans la mesure où elle indique que la connaissance d'un temps, ou d'un effort moyen suffit pour définir complètement la distribution de probabilité. Or, nous avons expliqué que nous étions capable de déterminer de telles valeurs moyennes tant pour le temps que pour l'effort. L'adoption d'une telle distribution est donc tout à fait cohérente avec ce que nous sommes à même de réaliser.

De plus, cette distribution a un autre avantage, d'ordre pratique celui-là. Si nous utilisons des taux de transition constants dans notre réseau de Petri, nous retombons dans le cadre bien connu des réseaux de Petri stochastiques[2] dont les graphe des marquages sont des graphes de Markov ([Natkin 1980][107], [Molloy 1982][104]).

Les graphes de Markov sont couramment utilisés dans le domaine de la fiabilité. En effet, ils sont relativement simples à mettre en œuvre pour modéliser des systèmes matériels et pour évaluer différentes mesures de sûreté de fonctionnement [Arlat et al. 1989][9]. Confrontés à des problèmes d'évaluation similaires, il est raisonnable de vouloir profiter des propriétés mathématiques intéressantes de ce type de distribution.

La justification du choix d'un modèle mathématique ne doit pas tant reposer sur sa capacité à représenter parfaitement le processus étudié qu'à délivrer des résultats qui soient plausibles et utiles. En effet, nous savons que tous les modèles mathématiques sont, plus ou moins, imparfaits. Cependant, ils sont acceptables s'il est possible de montrer qu'ils offrent une approximation satisfaisante des aspects intéressants de la réalité. Dès lors, il est nécessaire de prouver que les résultats que nous pouvons obtenir dans le cadre markovien respectent bien les propriétés désirées du modèle, telles que présentées dans la section III.6.5 (page 71).

IV.3.3 Propriétés mathématiques

Nous formalisons à présent, sur base du graphe des privilèges, les notions introduites avec l'analogie du labyrinthe (section III.6.5, page 71). Nous supposons qu'un attaquant possède un ensemble de privilèges et tente d'acquérir un autre. Il doit trouver un chemin dans le graphe des privilèges entre l'ensemble qu'il possède et celui qu'il désire. Le terme de chemin fait donc

1. La probabilité de réussir une attaque dans l'intervalle [0,dt] est égale à la probabilité de réussir la même attaque dans l'intervalle [t,t+dt], pour tout t, sachant qu'elle n'a pas encore réussi avant l'instant t.
2. Les réseaux de Petri stochastiques sont une extension des réseaux de Petri classiques pour lesquels les transitions ne sont plus tirables de façon instantanée, mais en fonction de taux de transition constants.

référence au graphe de privilèges ; les taux de transition que nous avons définis pour les arcs modélisent les taux de défaillance des serrures des portes du labyrinthe.

Nous tentons dans ce qui suit de trouver un juste compromis entre explication intuitive et rigueur mathématique. Nous ne nous intéressons pas tant aux détails des mathématiques qui mènent à l'obtention des résultats qu'à la façon dont ces valeurs évoluent en fonction de différentes configurations. De nombreuses valeurs peuvent être évaluées sur base des graphes de Markov. Nous avons choisi de nous pencher sur l'une d'entre elles en particulier : le *MTTF* (*Mean Time To Failure*). Cette valeur représente le temps moyen pour qu'un attaquant atteigne une cible. Ce choix est motivé par la facilité d'interprétation physique de cette valeur, mais aussi par la simplicité de son obtention qui permet de construire des preuves plus rigoureuses des propriétés que nous désirons démontrer. Bien entendu, temps et effort apparaissant de façon similaire dans notre modélisation, tous les résultats obtenus pour l'un sont applicables à l'autre.

A. Expression du MTTF

Dans le cas de graphes de Markov acycliques —comme le sont ceux que nous traitons—, le *MTTF* est égal à la somme des temps de séjour dans les différents états qui mènent à la cible, pondérés par la probabilité de se trouver dans ces états. Le temps de séjour dans un état est égal à l'inverse de la somme des taux de transition des arcs qui quittent cet état. La probabilité de quitter un état par un arc donné est égale au taux de transition de cet arc multiplié par le temps de séjour dans l'état quitté. Formellement, nous pouvons écrire :

$$MTTF_k = T_k + \sum_{l \in out(k)} P_{kl} \times MTTF_l \;;\; T_k = 1/\left(\sum_{l \in out(k)} \lambda_{kl}\right) \; et \; P_{kl} = \lambda_{kl} \times T_k$$

Dans ces équations, la variable $MTTF_k$ représente le *MTTF* calculé en considérant l'état k comme l'état initial ; T_k représente le temps de séjour en l'état k ; *out(k)* représente l'ensemble des états que l'on peut atteindre en une transition à partir de l'état k ; P_{kl} représente la probabilité de quitter l'état k pour l'état l ; λ_{kl} représente le taux de transition de l'état k vers l'état l.

B. Chemin rectiligne

i) Explication physique

Dans l'analogie du labyrinthe, nous avons indiqué que la longueur d'un chemin rectiligne devait croître avec la qualité des serrures et le nombre de portes à franchir (section III.6.2, page 70). Ce cas de figure donne lieu à un graphe de Markov identique au graphe des privilèges où tout état, exception faite du dernier, ne peut être quitté que par une seule transition. Dans ce cas très simple, le graphe de Markov est rectiligne, tout comme le graphe de privilèges. Toute autre chose restant égale, il est évident que si nous rajoutons une étape à la chaîne ou si nous baissons la valeur d'un taux de transition, le *MTTF* de ce nouveau graphe sera supérieur à celui du précédent.

Nous avons représenté un tel chemin dans la figure IV.8. Les λ_i symbolisent les taux de transition moyens connus pour chaque transition. Dans ce cas, l'application des équations

définies précédemment est très simple et le temps moyen pour atteindre la cible Z est égal à $\frac{1}{\lambda_1} + \frac{1}{\lambda_2}$.

Figure IV.8 Exemple de chemin rectiligne

Ce résultat peut se paraphraser de la façon suivante :

- Le premier terme $\left(\frac{1}{\lambda_1}\right)$ représente le temps moyen que va attendre l'attaquant avant de franchir la transition vers Y. En termes de graphe de Markov, c'est le temps de séjour dans l'état initial.
- Le second terme $\left(\frac{1}{\lambda_2}\right)$ représente le temps de séjour moyen dans le second état.
- Le temps moyen total pour arriver à la cible est la somme des temps de séjour moyens qui mènent à l'état final.

De cette expression simple, il ressort bien que la longueur sera d'autant plus grande que les λ_i seront petits et que le nombre d'arcs sera grand.

C. Influence nulle des impasses

i) Explication physique

Dans notre analogie du labyrinthe, nous n'avons pas mentionné l'existence d'impasses, c'est-à-dire de séquences de chemins qui ne mènent pas à la cible. Dans l'absolu, elles ont un impact positif sur la sécurité puisqu'elles compliquent la tâche de l'assaillant. Cependant, d'un point de vue philosophique, nous concevons difficilement d'améliorer la sécurité en laissant un assaillant mener plus d'attaques. C'est pourquoi, dans notre modèle, les valeurs des *MTTF* délivrées par le modèle sont indépendantes de l'existence éventuelle d'impasses dans le graphe des privilèges. Même si elles sont propices à égarer l'assaillant, nous ne désirons pas qu'elles puissent influencer les résultats de l'évaluation et risquer de donner un faux sentiment de sécurité. Pour simplifier, nous pourrions dire qu'il est "évident" que ces impasses n'interviennent pas dans le calcul puisque la probabilité d'arriver à la cible en y passant est nulle. C'est donc, "comme si" elles n'existaient pas. Ce raisonnement n'est qu'intuitif et appelle une explication plus rigoureuse.

ii) Calcul du MTTF

Les impasses sont également présentes dans le réseau de Petri sous la forme de séquences de transitions qui ne mènent pas à la cible. Elles sont tirables et apparaissent donc dans le graphe de Markov. Cependant, il est possible de montrer qu'elles n'influencent pas les résultats. Afin de le prouver, nous faisons appel au théorème démontré par Kemeny et Snell [Kemeny et Snell 1959][83] qui définit la notion d'états équivalents et indique les conditions de réductibilité d'un graphe de Markov :

Soit $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_s\}$ *une partition de l'espace d'états* E *et* $s < m = |E|$ *; de plus, soit* $\lambda_{l,\mathcal{P}_j} = \sum_{k \in \mathcal{P}_j} \lambda_{lk}$ *où* λ_{lk} *représente le taux de transition de l'état* l *vers l'état* k

de $\mathcal{P}_j$. Le modèle est réductible par rapport à $\mathcal{P}$ c'est-à-dire que l'on peut substituer un seul état à chaque partition $\mathcal{P}_m$, si pour toute paire $(\mathcal{P}_i, \mathcal{P}_j)$, les $\lambda_{l,\mathcal{P}_j}$ sont identiques pour tout état l de $\mathcal{P}_i$. Ces valeurs communes sont les taux de transition de la chaîne réduite.

L'intérêt de ce théorème dans le cadre de notre travail est double. Tout d'abord, il nous permet de baser nos raisonnements et démonstrations sur des graphes de Markov dont aucun état, mis à part l'état terminal, n'est atteint par plusieurs transitions différentes. En effet, il est toujours possible de construire un tel graphe à partir d'un graphe quelconque comme l'illustre la figure IV.9. Ce dédoublement de nœuds est, d'ailleurs, un artifice couramment utilisé dans les démonstrations de propriétés de graphe. L'application du théorème prouve que les deux graphes sont équivalents.

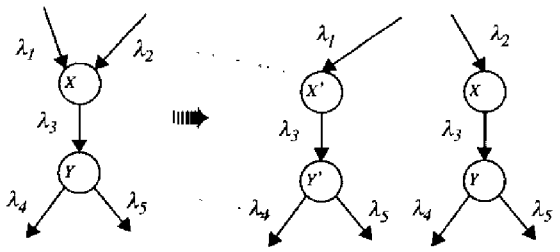


Figure IV.9 Graphe des marquages sans intersection

D'autre part, la structure très particulière de notre graphe des marquages nous permet de tirer parti de ce théorème pour démontrer l'influence nulle des impasses. En effet, considérons un graphe de privilèges G_1. Créons un graphe G_2 à partir de G_1 en introduisant un nouvel ensemble de privilèges et en établissant, vers cet ensemble, une transition T partant de l'ensemble de privilèges que détient initialement l'attaquant. Les deux graphes sont équivalents à la différence près que G_2 possède une impasse que n'a pas G_1. Montrons que leurs graphes de marquages sans intersection $\mu(G_i)$ sont équivalents.

Puisque la transition T mène à une impasse, elle ne peut aboutir à l'état final F. T peut avoir été tirée, au plus tard, pour atteindre un état précédant directement l'état final. Soit E_f un état de $\mu(G_1)$ dont toutes les transitions de sortie mènent à F (un tel état existe toujours). Dans $\mu(G_2)$, l'état correspondant à E_f, E'_f, possède une transition qui ne mène plus à l'état final : T permet de passer de E'_f à E'_{f+1}. T mise à part, toutes les transitions en sortie de E'_f et E'_{f+1} sont identiques et mènent à l'état final. Donc, au vu du théorème précédent, le modèle est réductible et nous pouvons remplacer E'_f et E'_{f+1} par un seul état, E''_f, en faisant disparaître T. Ceci est représenté dans la figure IV.10.

Soit à présent l'état E'_{f-1} qui précède E'_f dans $\mu(G_2)$. Il est identique à l'état E_{f-1} qui précède E_f dans $\mu(G_1)$ à la différence près que de E'_{f-1} part la transition T vers un état qui possède les mêmes transitions en sortie que E'_{f-1}. Donc, nous pouvons à nouveau réduire le graphe. En itérant le processus, il est clair que nous allons parvenir à réduire le graphe jusqu'à obtenir un graphe $\mu(G_2)$ identique au graphe $\mu(G_1)$.

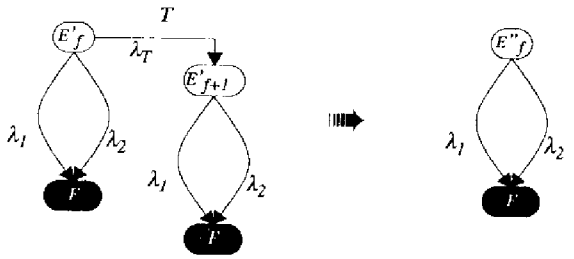


Figure IV.10 Exemple d'états équivalents

D. Chemins multiples

i) Explication physique

Nous étions également parvenu à la conclusion selon laquelle la mesure de la sécurité doit être d'autant plus faible que le nombre de chemins différents aboutissant à la cible augmente. Cette propriété est également vérifiée dans notre modèle car nous avons modélisé explicitement la mémoire de l'attaquant. En effet, rajouter des chemins revient à offrir plus de possibilités de sortir d'un état donné. Cela signifie que l'assaillant peut quitter, en moyenne, cet état plus rapidement. Bien sûr, il se peut qu'un chemin rajouté soit particulièrement rapide au départ, puis très lent. Dans ce cas, puisqu'il est rapide, la probabilité qu'il soit "choisi" au départ est forte. Cependant, à chaque étape le long de ce chemin, l'assaillant a souvenir de l'existence d'autres voies. Si le chemin devient "lent", il aura donc d'autant plus tendance à le quitter que d'autres existent et l'influence du chemin "lent" sera d'autant plus faible. A la limite, pour un chemin infiniment "lent" —une impasse—, l'influence est nulle. En moyenne, c'est le gain en temps de séjour dans les états qui l'emporte sur la longueur du chemin rajouté. Nous allons le montrer de façon plus formelle.

ii) Calcul du MTTF

A titre d'exemple, nous avons représenté dans la figure IV.11 deux chemins différents qui mènent à la cible Z. Nous y avons également représenté le réseau de Petri obtenu par transformation et le graphe de Markov dérivé. Dans ce graphe des marquages, les triplets indiquent le nombre de jetons dans les places principales X, Y et Z du réseau de Petri initial.

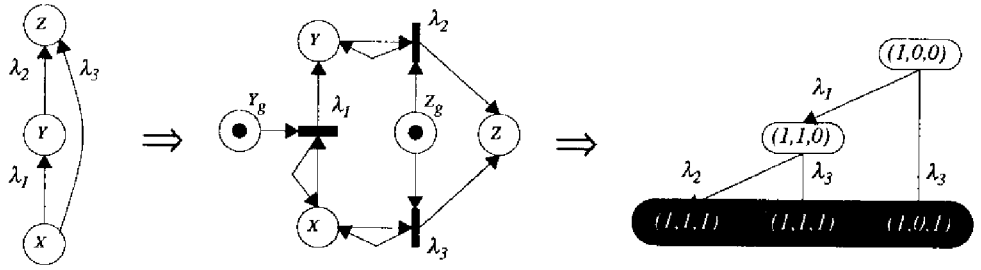


Figure IV.11 Exemple de chemins multiples

Trois états différents correspondent à l'arrivée de *X* en *Z*. Ils sont représentés par les triplets (1,1,1), (1,1,1) et (1,0,1). Appelons l'état correspondant au marquage (1,0,0), l'état 1, et celui correspondant au marquage (1,1,0), l'état 2. Appliquons les formules décrites à la page 88 afin d'obtenir le *MTTF* :

(1) $T_1 = \frac{1}{\lambda_1 + \lambda_3}$; $T_2 = \frac{1}{\lambda_2 + \lambda_3}$; $P_{12} = \frac{\lambda_1}{\lambda_1 + \lambda_3}$.

(2) $MTTF_1 = T_1 + P_{12} \times MTTF_2 = T_1 + P_{12} \times T_2 = \frac{1}{\lambda_1 + \lambda_3} + \frac{\lambda_1}{\lambda_1 + \lambda_3} \times \frac{1}{\lambda_2 + \lambda_3}$

Il nous faut prouver que cette expression est toujours inférieure à $\frac{1}{\lambda_1} + \frac{1}{\lambda_2}$ aussi bien qu'à $\frac{1}{\lambda_3}$ et ce pour toute valeur de λ_1, λ_2 et λ_3. Ceci est toujours vrai, en effet :

(1) Vis-à-vis de la somme $\frac{1}{\lambda_1} + \frac{1}{\lambda_2}$, chacun des termes est supérieur à son correspondant dans l'autre somme. Elle est donc bien supérieure.

(2) Vis-à-vis de $\frac{1}{\lambda_3}$, la vérification de l'inégalité est tout aussi rapide. Pour montrer que $\frac{1}{\lambda_1 + \lambda_3} + \frac{\lambda_1}{\lambda_1 + \lambda_3} \times \frac{1}{\lambda_2 + \lambda_3} \leq \frac{1}{\lambda_3}$ il suffit de faire passer le premier terme à droite de l'inégalité et réduire au même dénominateur, cela donne $\lambda_1 \times \frac{1}{\lambda_2 + \lambda_3} \leq \lambda_1 \times \frac{1}{\lambda_3}$.

Cet exemple simple permet de mieux comprendre les raisons pour lesquelles toute suppression de chemin dans le graphe des privilèges ne peut qu'augmenter la valeur du *MTTF* du nouveau graphe. Il faut à présent généraliser ce résultat.

Considérons un graphe de privilèges quelconque et dérivons-en le graphe des marquages sans intersection. Supprimer un des chemins peut se faire, par exemple, en considérant qu'un des taux de transition du chemin est nul. Nous savons que le choix de l'arc enlevé n'a pas d'importance puisque les impasses n'ont pas d'influence sur la valeur finale. Choisissons donc d'enlever le dernier arc du chemin ; soit λ_z le taux du dernier arc. Nous savons que cet arc n'apparaît nulle part ailleurs dans le graphe des marquages. La contribution de λ_z ne se fait donc que dans les dénominateurs de quelques T_i et P_{ij}. Donc, si $\lambda_z = 0$, tous ces T_i et P_{ij} augmenteront de valeur, puisque $\forall X, Y, \delta > 0 \quad \frac{X}{Y} > \frac{X}{Y + \delta}$. En conséquence, la somme à laquelle ils participent ne peut qu'augmenter elle aussi. Donc, si nous supprimons un chemin, le *MTTF* augmente, ce qui signifie que la sécurité augmente.

E. Prépondérance du plus court chemin

Des formules qui précèdent, nous déduisons que le temps pour arriver à la cible est toujours plus faible que le temps mis par le plus court chemin. Cependant, en raison du système de pondération utilisé, la valeur finale est principalement influencée par les chemins les plus courts. A titre d'exemple, nous calculons le temps moyen du graphe donné à la figure IV.11 en prenant les valeurs suivantes : $\lambda_1 = \lambda_2 = \lambda_3/10 = \lambda$. Ceci revient à imaginer que le chemin direct de *X* vers *Z* est beaucoup plus rapide que celui passant par *Y*. Dans ce cas, le temps moyen pour atteindre *Z* vaut, comme expliqué précédemment,

$$\frac{1}{\lambda_1 + \lambda_3} + \frac{\lambda_1}{\lambda_1 + \lambda_3} \times \frac{1}{\lambda_2 + \lambda_3} = \frac{1}{\lambda + \lambda \times 10} + \frac{\lambda}{\lambda + \lambda \times 10} \times \frac{1}{\lambda + \lambda \times 10} = \frac{12}{121\lambda} < \left(\frac{1}{10\lambda} = \frac{1}{\lambda_3} \right)$$

L'influence du chemin lent, passant par *Y*, est donc très faible par rapport à celle du chemin direct.

F. Collusion de plusieurs attaquants

Au vu de ce qui précède, il est facile de comprendre l'effet que peut avoir sur le *MTTF* l'action de plusieurs assaillants en même temps. Au niveau du réseau de Petri, cela revient à mettre un jeton dans les nœuds correspondant aux privilèges des différents utilisateurs qui sont censés faire partie de l'ensemble des assaillants. Du point de vue du graphe des marquages, en revanche, la différence entre un ou plusieurs assaillants disparaît. Il n'y subsiste que la notion d'état. Selon qu'il y a un ou plusieurs assaillants, l'état initial va varier dans le sens où le marquage va différer. Cependant, conceptuellement parlant, l'ajout d'un nouvel attaquant est totalement identique à l'ajout d'un nouveau chemin puisque toutes les attaques qui étaient possibles précédemment subsistent. Au pire, de nouvelles attaques apparaissent, au mieux rien ne se passe.

Toutes les remarques effectuées précédemment restent vraies et, en particulier, nous observons les résultats intéressants suivants :

(1) Le temps moyen mis par une population d'attaquants pour atteindre une cible est toujours inférieur au plus petit temps mis par chacun des assaillants.
(2) Le temps moyen mis par une population d'attaquants sera principalement influencé par les temps les plus courts mis par chacun des assaillants pris isolément.
(3) L'ajout d'un nouvel attaquant ne peut que faire décroître la sécurité.

Il est légitime de s'interroger sur le sort à réserver aux attaques qui relient entre eux deux ensembles de privilèges appartenant à des membres d'un même groupe d'attaquants. Puisque nous faisons l'hypothèse d'une collusion de plusieurs personnes, il semble normal de penser qu'ils ne vont pas dépenser du temps et de l'énergie à s'attaquer mutuellement. Nous devrions donc commencer par retirer du graphe toutes les attaques qui aboutissent à des assaillants. En réalité, les propriétés mathématiques du modèle nous en dispensent.

En effet, considérons un graphe où aucun assaillant ne peut voler les privilèges d'un autre et dérivons en le graphe des marquages. A présent, ajoutons un arc entre deux assaillants A et B et observons le nouveau graphe des marquages. La modification introduite dans ce nouveau graphe est exactement la même que dans le cas de l'ajout d'une impasse puisque cette transition, comme l'impasse, ne peut apparaître à la fin d'un chemin. Donc, l'ajout de cet arc ne modifie en rien les valeurs calculées.

G. Expertise de l'attaquant

Nous avons déjà évoqué le cas d'attaques qui demandaient un effort de mise en œuvre important. Nous avions montré qu'il était possible d'utiliser un canevas particulier de réseau de Petri pour décider si, oui ou non, nous pensions que l'attaquant allait utiliser cette attaque. Avec les réseaux de Petri stochastiques, nous pouvons maintenant généraliser l'usage de ce canevas aux attaques qui ne demandent d'effort ou de temps que la première fois qu'elles sont utilisées. En effet, il suffit d'utiliser le canevas décrit dans la figure IV.6 (page 85) et d'adjoindre à la transition t_c un taux représentant l'effort de mise en œuvre initiale. Une fois

cette transition franchie, toutes celles du graphe qui utilisent l'attaque seront franchissables sans dépenser à nouveau cet effort.

IV.4 Limites et avantages

IV.4.1 Limites

A. Discussion des hypothèses

Nous avions supposé que les attaquants se souviennent, au fur et à mesure de leur progression, des attaques qu'ils n'ont pas encore utilisées. Ils se souviennent également des privilèges qu'ils ont acquis. Aucune hypothèse n'est faite sur leur stratégie d'attaque si ce n'est que la probabilité de parcourir un chemin est fonction de la facilité relative de celui-ci par rapport aux autres. La logique du processus d'attaque est prise en compte par la construction ad-hoc du réseau de Petri. S'il s'avérait possible de mieux caractériser le processus d'attaque, il faudrait alors modifier le canevas de base défini dans la section IV.2.3 (page 83) en fonction des nouvelles hypothèses à adopter.

Le modèle d'évaluation proposé étudie la difficulté pour un intrus de violer les objectifs de protection. Il ne cherche pas à évaluer la probabilité qu'un intrus s'intéresse au système. Il serait intéressant d'incorporer cette notion, mais, dans l'état actuel des connaissances (discuté dans la section IV.1.3, page 75), il nous a semblé déraisonnable de mêler ces deux processus que sont le choix d'un système à attaquer et le choix d'une méthode pour y arriver.

Nous avons adopté un cadre markovien pour mener à bien nos évaluations. Nous avons montré que ce cadre, quoique imparfait, était acceptable puisque nous pouvions être assuré de la plausibilité des résultats qu'il donnait. Le choix d'autres distributions de probabilité nécessiterait de vérifier qu'elles conservent les propriétés intuitives prouvées dans la section IV.3.3 et compliquerait la phase de calcul proprement dite, mais ne remettrait pas en cause la méthode générale proposée dans ce document.

B. Valeur relative des résultats

A la différence des critères d'évaluation décrits au premier chapitre, notre méthode n'a pas la prétention de classer les systèmes entre eux en fonction des valeurs qu'elle délivre. En effet, nous ne pouvons comparer des systèmes aux objectifs de protection différents. En revanche, notre méthode se révèle particulièrement utile pour étudier l'évaluation de la sécurité d'un système dans son environnement opérationnel.

IV.4.2 Avantages

A. Identification des chemins critiques

Nous avons montré que les *MTTF* étaient toujours inférieurs aux plus courts chemins et que ceux-ci avaient une importance prépondérante dans le calcul. Les résultats permettent donc de mettre en exergue les chemins critiques tout en restant sensibles au nombre de chemins existant.

De plus, nous avons montré que l'existence de chemins qui ne conduisent pas à la cible n'influence pas le résultat final. Certains peuvent considérer que, en pratique, la multiplicité de chemins menant à des impasses est "bonne" pour la sécurité puisque l'attaquant pourrait s'y égarer et se lasser. Une telle vision de la sécurité nous semble pour le moins critiquable et dangereuse.

B. Suivi de la sécurité opérationnelle

Le premier champ d'application de notre méthode nous semble être le suivi de la sécurité opérationnelle des systèmes. Nous avons expliqué que le graphe des privilèges peut évoluer en fonction du comportement au jour le jour des utilisateurs. Il est nécessaire de vérifier que cette évolution ne va pas porter préjudice au système. Puisque notre méthode, depuis la construction du graphe jusqu'à l'obtention des valeurs, peut être réalisée automatiquement, sans intervention humaine, il est tout à fait raisonnable d'imaginer obtenir des valeurs initiales dans un état de référence que l'on considère comme satisfaisant et vérifier par un calcul journalier qu'aucun résultat ne s'en écarte significativement. Le dernier chapitre présente le prototype d'un outil qui permet d'atteindre un tel objectif.

C. Aide à la détection d'intrusions

Une dégradation significative de la sécurité peut également témoigner d'une intrusion en cours de réalisation. Ceci montre que notre méthode peut également être utilisée en tant qu'aide à la détection d'intrusions. Cependant, la qualité demandée à un tel outil est d'agir en temps réel ce que ne peut faire notre système dont le graphe n'est reconstruit qu'à intervalle plus ou moins long. La détection d'intrusions avec le graphe arrive donc "trop tard". Toutefois nous pouvons envisager d'utiliser le graphe pour enrichir un véritable système de détection d'intrusions. En effet, si un système classique tel que *NIDES* [Anderson et al. 1993][5] détecte un comportement suspicieux, il est possible d'identifier sur le graphe l'état dans lequel l'intrus potentiel se trouve. Le niveau de sévérité de l'alarme peut être ajusté en fonction de la distance qui le sépare encore des objectifs de protection.

IV.5 Conclusions

Dans ce chapitre, nous avons précisé les variables utilisées pour mettre en œuvre l'évaluation quantitative de la sécurité. Nous avons présenté le formalisme de représentation du processus d'attaque qui nous permet de dériver automatiquement les différents scénarios offerts à l'assaillant. Nous avons montré le lien entre ces scénarios et le modèle mathématique d'évaluation choisi. Ses propriétés mathématiques ont été démontrées conformes à celles que nous avions espérées au chapitre III. Enfin, nous avons souligné les limites, mais également les avantages de notre méthode et indiqué des pistes pour son application pratique.

Cette réflexion est poursuivie dans le prochain et dernier chapitre qui présente la mise en œuvre pratique de la méthode d'évaluation définie dans ce mémoire. A l'aide d'un exemple simple, nous mettons en lumière les différentes étapes nécessaires à l'obtention de mesures de sécurité. Nous y abordons également le problème de leur interprétation et de leur utilisation.

CHAPITRE V *Mise en œuvre de la méthode*

Préambule

Dans les chapitres qui précèdent, nous avons montré que :

(1) L'état de l'art en matière d'évaluation de la sécurité de systèmes d'information laisse apparaître la nécessité de recourir à des méthodes de prévision de fautes.
(2) Les méthodes d'analyse des risques, souvent issues d'autres cadres d'applications, sont mal adaptées aux systèmes informatiques car elles les modélisent peu ou prou.
(3) Les modèles formels de sécurité adoptent, pour la plupart, une hypothèse de pire cas sur le comportement des attaquants ce qui n'offre pas une vision réaliste des systèmes étudiés.
(4) Un nouveau modèle, le graphe de privilèges, s'affranchit de cette hypothèse et une application concrète en a été donnée.
(5) Ce modèle peut servir de support à une méthode de prévision de fautes dont nous avons précisé les propriétés mathématiques.

Dans ce chapitre, nous donnons un exemple de mise en œuvre de la méthode d'évaluation de la sécurité décrite dans ce travail. Nous montrons comment utiliser le graphe des privilèges et les mesures qui en découlent pour étudier l'évolution de la sécurité d'un système UNIX dans son environnement opérationnel. En particulier, nous abordons sur un exemple simple le problème de l'interprétation et de l'utilisation des résultats.

Ce chapitre est composé de trois parties.

La première décrit la structure d'un prototype développé afin de valider expérimentalement les hypothèses et idées décrites dans ce mémoire. Trois étapes sont identifiées pour mener à bien la tâche d'évaluation quantitative : (i) modélisation de la sécurité et du système, (ii) calcul des mesures, (iii) interprétation des résultats.

La deuxième partie commente les résultats obtenus grâce au prototype et leur interprétation.

Enfin, la troisième et dernière partie offre une présentation critique des limites et avantages du prototype actuel.

V.1 Présentation du Prototype

V.1.1 Introduction

Le prototype d'évaluation que nous avons réalisé a pour but de confronter à la réalité d'un système concret les concepts développés jusqu'ici. Dans son état actuel, il se présente sous la forme d'une interface graphique qui intègre au sein d'un même environnement les différents outils nécessaires à l'évaluation d'un système.

Dans les pages qui suivent, nous avons privilégié une approche mettant en relief les différentes étapes de l'évaluation et la façon dont elles interagissent plutôt qu'une description détaillée du prototype. Nous engageons le lecteur intéressé par les aspects d'implémentation à consulter [Arnaud 1994][10].

La figure V.1 décrit les différentes étapes qui permettent de mener à bien l'évaluation quantitative.Les boîtes grisées représentent des outils tandis que les boîtes claires représentent les paramètres d'entrée et de sortie de chacun d'eux. Cette figure représente également l'interface graphique développée pour le prototype. A chaque boîte grisée sont associés des "boutons" qui permettent, en particulier, d'exécuter l'outil désigné ou de visualiser le détail des opérations qu'il implémente. Cette dernière fonctionnalité est illustrée au cours de ce chapitre.

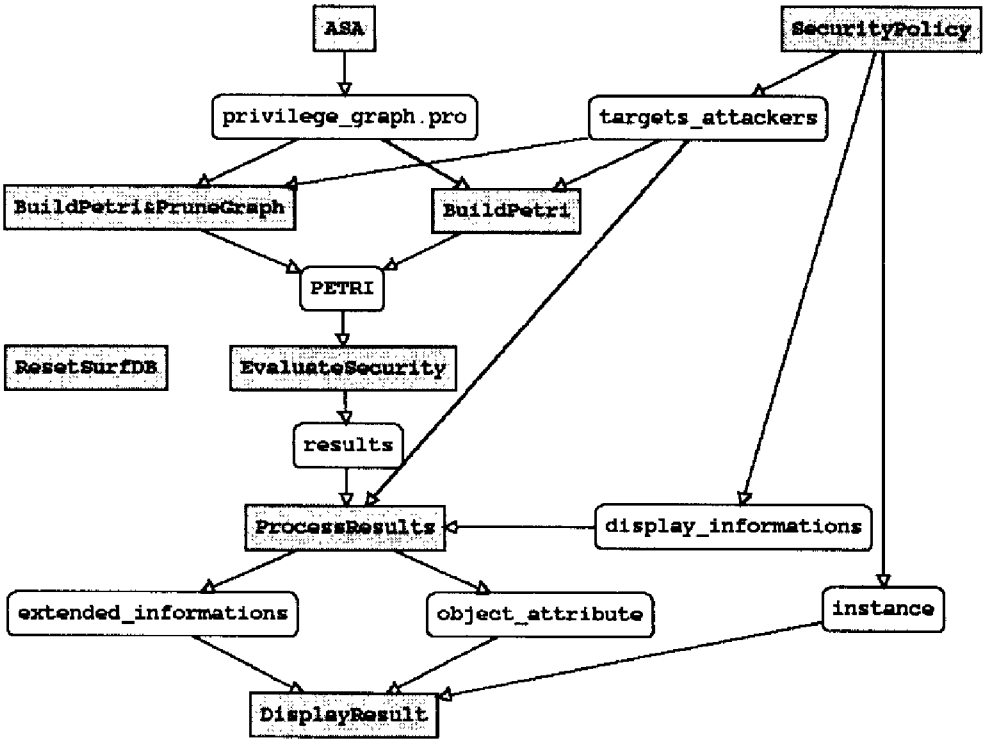


Figure V.1 Le prototype d'évaluation

Le processus complet d'évaluation se subdivise en quatre phases distinctes :

(1) **La construction du graphe des privilèges :** dans un premier temps, nous modélisons le système à étudier en construisant le graphe de privilèges avec l'outil *ASA*.

(2) **La spécification de la sécurité :** nous spécifions les objectifs de protection, c'est-à-dire la définition des cibles et des attaquants à l'aide de l'outil *SecurityPolicy*.

(3) **La phase de calcul :** conformément à ce qui a été expliqué au chapitre IV, (i) nous construisons le modèle qui va servir à l'évaluation (outil *BuildPetri* ou *BuildPetri&PruneGraph*[1]) et (ii) nous calculons un ensemble de mesures (outil *EvaluateSecurity*).

(4) **La gestion des résultats :** une fois les calculs effectués, il est nécessaire de les présenter sous une forme utile à l'utilisateur. Pour cela, nous traitons les résultats selon certains critères de sélection (outil *TreatResults*) puis nous les présentons à l'utilisateur (outil *DisplayResult*) en les confrontant aux objectifs de protection définis dans la phase de spécification de la sécurité.

Chacune de ces quatre phases fait l'objet d'une description plus détaillée dans les sections qui suivent.

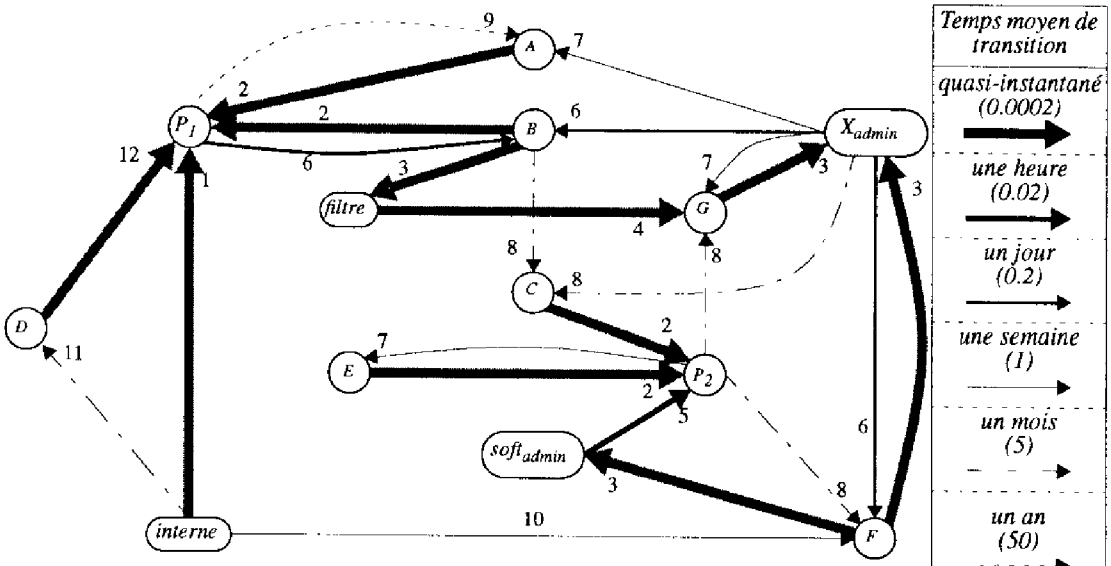
V.1.2 Construction du graphe des privilèges

Nous avons expliqué dans le chapitre III qu'il était possible de construire automatiquement un graphe de privilèges dans le cadre des systèmes UNIX. En nous basant sur le travail déjà réalisé en 1990 à l'université catholique de Louvain [Dacier et Rutsaert 1991a][36], nous avons défini et réalisé un nouveau logiciel qui dérive un graphe de privilèges par inspection d'un système de fichiers UNIX. Les variables "temps" et "effort" sont instanciées pour chaque arc au cours de cette même phase de construction. Nous utilisons pour cela les critères explicités dans le chapitre IV (section IV.1.4, page 77).

Nous avons reproduit dans la figure V.2 le graphe déjà décrit dans le chapitre III (figure III.5, page 67). Dans cette nouvelle représentation, l'épaisseur d'un arc est proportionnelle à la valeur de son taux de transition : plus un arc est épais, plus la transition associée est rapide.

Dans l'exemple choisi, un taux de transition unitaire correspond à une transition franchissable, en moyenne, une fois par semaine. Pour simplifier le discours nous disons qu'un taux valant 0.2 (respectivement, 0.02) correspond à une transition que l'on peut franchir, en moyenne, une fois par mois (respectivement, une fois par an). De même, la valeur 5 (respectivement, 50) correspond à une transition franchissable une fois par jour (respectivement, une fois par heure). Pour caractériser les attaques quasi instantanées, nous avons choisi un taux de transition très grand : 5000.

1. L'outil *BuildPetri&PruneGraph* retire toutes les impasses du graphe des privilèges. Son utilisation optimise le réseau de Petri, et par suite, le nombre d'états du graphe de Markov, mais ne permet pas d'identifier tous les scénarios possibles.



1) Y n'a pas de mot de passe ; 2) X est présent dans le .rhosts de Y ; 3) X est un sur-ensemble de Y ; 4) X attaque Y par le biais du courrier électronique ; 5) Y utilise, une fois par heure, un logiciel géré par X ; 6) Y utilise, une fois par jour, un logiciel géré par X ; 7) Y utilise, une fois par semaine, un logiciel géré par X ; 8) Y utilise, une fois par mois, un logiciel géré par X ; 9) Y utilise, une fois par an, un logiciel géré par X ; 10) mot de passe de Y devinable en une semaine ; 11) mot de passe devinable en un mois ; 12) X peut modifier l'exécution d'un programme dont le setuid bit est positionné appartenant à Y.

Figure V.2 Exemple de graphe de privilèges pondéré

V.1.3 Spécification de la sécurité

A. Définition des cibles

i) Miró : introduction

Une fois le graphe construit, nous offrons à l'utilisateur un moyen simple de spécifier les objectifs de protection. Nous utilisons pour cela un langage graphique : Miró [Heydon et al. 1990][70]. Miró permet de spécifier des contraintes sur l'état de protection d'un système et de vérifier si elles sont, ou non, respectées au sein d'un système de fichiers UNIX classique.

Miró utilise un environnement de programmation particulier, Garnet [Myers et al. 1990][106], qui fournit toutes les primitives de base nécessaires à la gestion des interfaces graphiques. Depuis sa première présentation en 1989 ([Heydon et al. 1989a][68],[Heydon et al. 1989b][69]), Miró a subi quelques modifications ([Maimone et al. 1990][99], [Heydon et al. 1990][70], [Heydon 1992][71] et [Heydon et Tygar 1994][72]). Certains concepts ont évolué au cours du temps et les articles les plus anciens ne sont plus une image fidèle de l'état actuel du langage. Dans la suite, nous utilisons [Heydon 1992][71] comme ouvrage de référence.

Miró est composé de *deux* langages. Le premier est appelé *langage d'instance* et sert à représenter les droits d'accès que possèdent les utilisateurs sur les objets d'un système de

fichiers. Le second, appelé *langage de contraintes*, sert à définir des contraintes sur les droits d'accès. Un outil, le *"verifier"*, permet de vérifier qu'une contrainte est, ou non, satisfaite dans une instance particulière. Nous détaillons ces notions ci-après.

ii) Le langage d'instance

Dans la terminologie de Miró, une *"instance"* désigne une représentation graphique des droits d'accès que possèdent les utilisateurs sur les objets du système. Un outil, le *"prober"*, construit automatiquement par inspection du système de fichiers, une telle instance pour un système UNIX classique.

Nous représentons dans la figure V.3 une instance dont l'interprétation est immédiate. Si toutes les instances étaient aussi simples, il serait possible, par simple inspection, de vérifier que les droits sont correctement attribués. Cependant, une telle représentation devient vite illisible lorsque le nombre d'utilisateurs et d'arcs augmente. Pour faire face à ce problème, Miró offre un autre langage qui permet d'exprimer de façon synthétique des contraintes sur les droits d'accès.

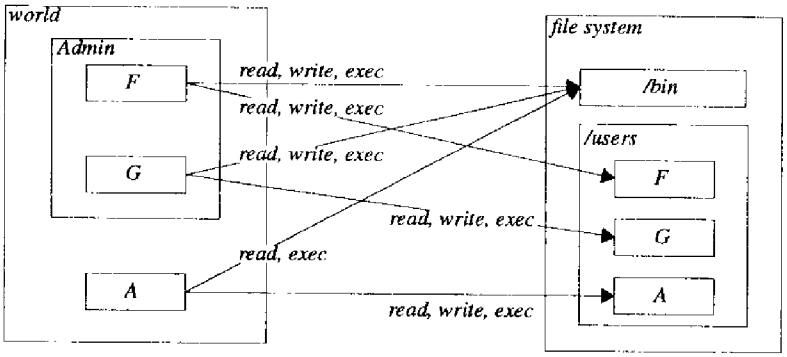


Figure V.3 Un exemple d'instance Miró

iii) Le langage de contraintes

Le langage de contraintes est plus riche que le langage d'instance dans la mesure où il manipule des prédicats et des variables. Une contrainte définit un ensemble (éventuellement infini) d'instances ; elle permet d'exprimer graphiquement des expressions du type *"si certaines boîtes et/ou flèches existent, alors certaines boîtes et/ou flèches doivent (ou ne doivent pas) exister"*. La partie conditionnelle de cette expression est représentée en traits épais tandis que la seconde partie apparaît en traits fins. Les prédicats et variables caractérisant les boîtes permettent de préciser les conditions sur les boîtes et arcs représentés. Toute chaîne de caractères commençant par le symbole $ est interprétée comme une variable dans le langage de contraintes.

A titre d'exemple, nous avons représenté dans la figure V.4 une contrainte qui illustre ce type de construction. Son interprétation est la suivante : *"S'il existe dans l'instance une boîte de type 'user' dont le nom est $X, alors elle doit posséder les droits 'read, write, exec' sur une autre boîte de type 'home' portant le même nom"*. En d'autres termes, tout utilisateur doit détenir les droits de lecture, écriture et exécution sur son répertoire personnel.

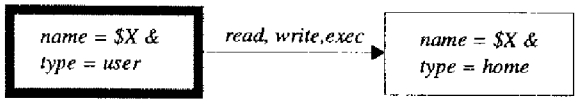


Figure V.4 Spécification des droits sur le répertoire personnel

Dans la figure V.5, nous spécifions que *A et E* doivent avoir accès en lecture au fichier ayant pour nom "*confidentiel*". Si nous voulons indiquer qu'aucun autre utilisateur, mis à part *A* et *E*, ne peut avoir accès en lecture à cet objet, nous devons ajouter une troisième boîte en traits épais, non reliée à l'objet "*confidentiel*", et dont le prédicat serait "name=$X & type = user". La contrainte exprime alors qu'aucun utilisateur autre que *A* et *E* ne peut détenir le droit de lecture sur l'objet "*confidentiel*". La condition exprimée est double. D'une part, elle spécifie les droits que doivent avoir certains utilisateurs (*A* et *E* en l'occurrence) ; nous parlons ici de la **partie positive** de la contrainte. D'autre part, elle permet de spécifier également des droits qui ne peuvent pas être présents dans l'instance, nous parlons alors de **partie négative** de la contrainte.

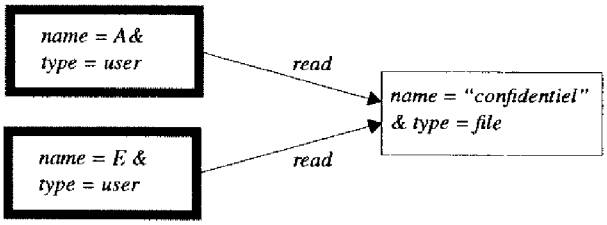


Figure V.5 Spécification des droits sur le fichier "*confidentiel*"

Nous terminons cette brève présentation du langage de contraintes par un dernier exemple extrait de [Heydon 1992][71], représenté dans la figure V.6. La contrainte qui est spécifiée peut se paraphraser de la façon suivante : "*Pour toute paire de boîtes W et R telles que W est de type world et R de type root, les trois conditions suivantes doivent être vérifiées dans l'instance étudiée : i) un arc étiqueté 'read' doit connecter W à R ; ii) chaque boîte directement contenue dans W doit être de type 'group' (à cause de la flèche verticale en traits gras) ; iii) il doit exister une boîte directement contenue dans R de type 'dir' et dont le nom doit être 'usr' (à cause de la flèche verticale en traits fins)*".

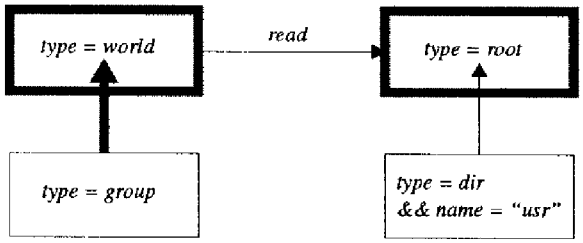


Figure V.6 Un exemple de contrainte Miró

Ce dernier exemple laisse entrevoir la richesse d'expression du langage de contraintes. A titre d'exemple, Heydon et Tygar montrent dans [Heydon et Tygar 1994][72] comment transcrire, en termes de contraintes Miró, les conseils donnés dans [Grampp et Morris 1984][58] sur la "bonne" définition des droits d'accès sur les objets d'un système UNIX.

Remarquons que la vérification des contraintes porte sur les droits d'accès présents dans le système. Or, comme nous l'avons expliqué, la seule inspection des droits n'est pas suffisante pour vérifier les objectifs de protection, en raison du problème du transfert possible des privilèges. Supposons, par exemple, que nous voulons étudier un système dont le graphe des privilèges associé est celui donné dans la figure V.2 (page 100). Nous pouvons écrire une contrainte qui stipule que seuls *A* et *E* ont le droit de lire le fichier "*confidentiel*". Supposons que sur l'instance du système, cette contrainte soit vérifiée. Il n'en reste pas moins vrai que, par transitivité des transferts de privilèges, tous les autres utilisateurs peuvent, *de facto*, lire le fichier "*confidentiel*"!

Pour pallier ce problème, nous enrichissons l'analyse des contraintes de Miró par l'utilisation conjointe du graphe des privilèges. Dans ce but, nous avons modifié le code associé au "*verifier*" de Miró. Dans un premier temps, ce programme vérifie, pour chaque contrainte, qu'elle est respectée dans l'instance. Le cas échéant, nous récupérons les noms des utilisateurs identifiés dans chaque contrainte comme pouvant, légitimement, avoir certains privilèges sur les objets à protéger. Ces utilisateurs, du point de vue d'un attaquant, représentent des cibles privilégiées, au sens premier du terme. Dès lors, à l'aide du graphe des privilèges, nous pouvons calculer les temps et effort nécessaires pour s'accaparer leurs privilèges. Il est ainsi possible d'utiliser la richesse du langage de contraintes pour définir nos cibles.

B. Définition des attaquants

Nous avons réalisé un éditeur graphique qui permet de regrouper les utilisateurs selon les rôles qu'ils occupent dans le système. Chaque rôle définit un ensemble d'attaquants potentiels. Nous avons donc autant d'ensembles d'attaquants à considérer que de rôles définis. A titre d'exemple, nous donnons dans la figure V.7 la représentation des rôles des différents utilisateurs du système fictif décrit dans le chapitre III. Ce système comprend 10 utilisateurs (*A, B, C, D, E, F, G,* P_1, P_2, *interne*) que nous regroupons en neuf rôles (*PROJ*1, *PROJ*2, *PROF*1, *PROF*2, *PROFESSOR, PhD, STUDENT, SYSTEM, insider*). L'appartenance à un rôle est symbolisée

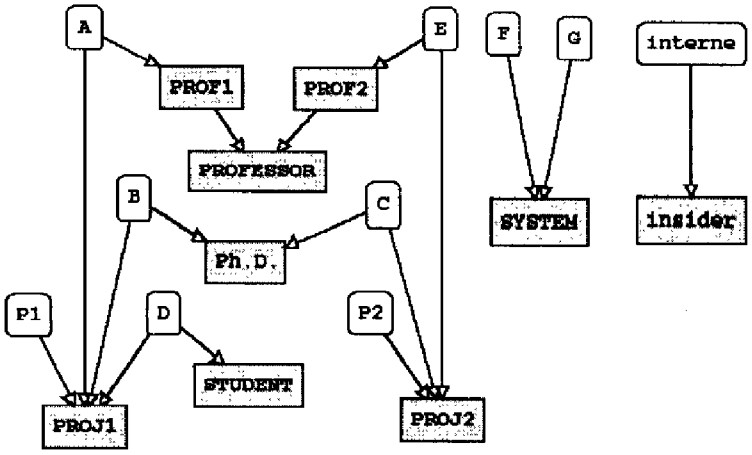


Figure V.7 Définition des rôles.

par une flèche pointant vers ce rôle.Par exemple, nous voyons que *B* appartient au rôle *PhD* (c'est un doctorant) et au rôle *PROJ1* (il fait partie du projet *PROJ1*).

La figure V.8 montre comment ces deux phases de définition des attaquants et des cibles s'intègrent dans notre démonstrateur. Elle représente les différentes composantes de l'outil *SecurityPolicy*. Sans rentrer dans le détail [Arnaud 1994][10], mentionnons que l'outil Miró génère l'instance et les contraintes tandis que l'éditeur de rôles fournit les attaquants. Un traitement est ensuite effectué qui génère dans le fichier *"targets_attackers"* toutes les combinaisons de groupes de cibles et de groupes d'attaquants. En effet, nous désirons calculer, pour chaque ensemble de cibles, le temps et l'effort nécessaires à chaque ensemble d'attaquants pour l'atteindre. Chaque ligne de ce fichier contient donc un couple "ensemble de cibles - ensemble d'attaquants". Le fichier *"display_informations"*, quant à lui, est utilisé dans la phase d'affichage des résultats.

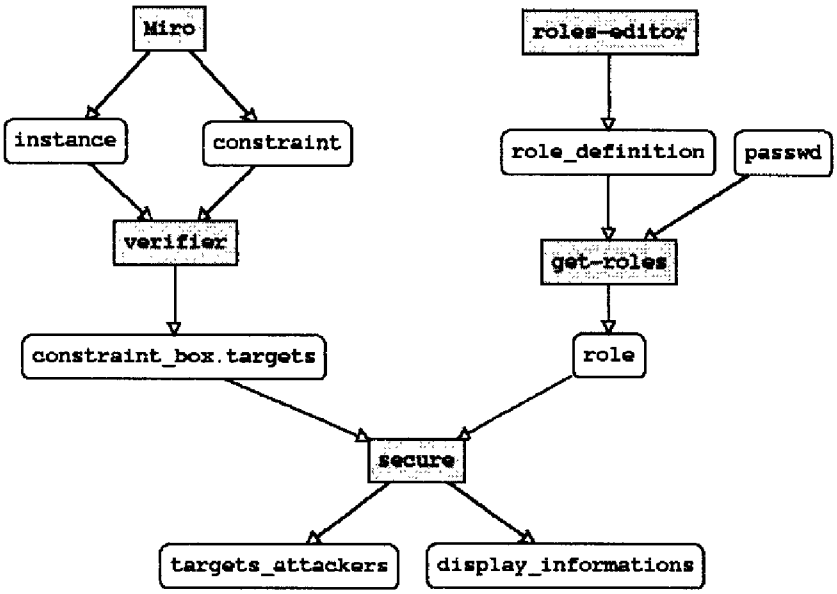


Figure V.8 Détails de l'outil *SecurityPolicy*

V.1.4 Phase de calcul

A. Obtention du modèle d'évaluation

Nous venons d'indiquer comment (i) construire le graphe des privilèges, (ii) définir les cibles et (iii) définir les ensembles d'attaquants. Il nous faut à présent intégrer ces données dans le modèle d'évaluation, à savoir le réseau de Petri stochastique décrit dans le chapitre IV.

Nous avons expliqué dans le chapitre IV les trois étapes nécessaires pour mener à bien la tâche d'évaluation : (i) transformer le graphe de privilèges en un réseau de Petri stochastique, (ii) définir le marquage initial de ce réseau en fonction de l'ensemble d'attaquants considéré, (iii) définir les conditions d'arrêt du processus d'attaque en fonction des cibles. Ces trois opérations sont réalisées à partir du fichier *"privilege_graph.pro"* et du fichier

"*targets_attackers*" par l'outil "*BuildPetri*" ou par "*BuildPetri&PruneGraph*" ; ceci est représenté sur la figure V.1, page 98.

B. Calcul des mesures

A l'aide du logiciel Surf-2 [Béounes et al. 1993][17], nous calculons ensuite l'ensemble des mesures désirées. Les résultats de l'évaluation sont ensuite extraits de la base de données de Surf-2 afin d'être présentés à l'utilisateur.

Surf-2 est un outil logiciel développé au *LAAS-CNRS* pour l'évaluation de la sûreté de fonctionnement des systèmes. Il permet de modéliser leur comportement à l'aide de graphes de Markov ou de réseaux de Petri stochastiques généralisés. Trois classes de mesures sont considérées dans cet outil :

- les probabilités transitoires et asymptotiques de présence dans un sous-ensemble d'états du modèle,
- le temps moyen de séjour dans un sous-ensemble d'états du modèle,
- les récompenses transitoire et asymptotique cumulées lors des séjours dans les états et lors des transitions entre les états du modèle.

Nous n'utilisons qu'une faible partie des possibilités de cet outil puisque nous ne nous intéressons qu'au calcul du *MTTF*. L'utilisation des autres mesures nécessiterait, au préalable, une étude approfondie de leurs propriétés afin de permettre leur interprétation dans notre cadre d'application particulier. Un tel travail pourrait utilement faire l'objet de travaux ultérieurs.

V.1.5 Gestion des résultats.

A. Traitement des résultats

Le nombre de résultats est potentiellement grand et il est donc nécessaire de synthétiser cette information. Dans ce but nous établissons un code de couleurs afin d'identifier aisément les objets les plus menacés dans le système. Chaque objet sur lequel portent des contraintes de protection apparaît à l'utilisateur dans une teinte qui est fonction des résultats de l'évaluation. La définition de la couleur d'un objet se fait en plusieurs étapes :

(1) Chaque objet est potentiellement menacé par plusieurs ensembles d'attaquants. Nous avons donc autant de *MTTF* associés à l'objet que d'ensembles d'attaquants. Nous analysons chaque valeur indépendamment des autres afin de la "qualifier". Dans le prototype actuel, le qualificatif représente simplement l'ordre de grandeur du *MTTF* (quasi instantané, heure, jour, semaine, mois, an). Cependant, cette étape pourrait être utilisée pour qualifier la "gravité" d'un *MTTF* vis-à-vis d'un rôle particulier. Par exemple, l'administrateur de sécurité peut considérer comme "normale" une valeur X pour le *MTTF* associé au rôle *SYSTEM* alors que la même valeur associée au rôle *STUDENT* sera qualifiée de "grave".

(2) La deuxième étape utilise ces qualificatifs pour en attribuer un à l'objet lui-même. Ici aussi, plusieurs choix sont possibles. Nous nous sommes limité, dans la version actuelle, à attribuer à l'objet le plus mauvais des qualificatifs qui lui sont associés. Cependant, nous pourrions, par exemple, lui attribuer un qualificatif qui rendrait compte du nombre de rôles différents qui le menacent "grandement".

B. Affichage des résultats

En fonction de l'attribut de l'objet, nous choisissons la couleur qui doit lui être associée. Dans le prototype actuel, chaque attribut correspond à une couleur précise. Cependant, nous pourrions faire en sorte que la couleur rende compte, par exemple, d'une modification de l'attribut, par rapport à sa valeur précédente ou par rapport à une valeur de référence.

V.2 Discussion des résultats

V.2.1 Introduction

Afin d'illustrer notre propos, nous présentons quelques résultats issus de ces différentes phases. Ils sont basés sur le graphe des privilèges présenté dans la figure V.2 (page 100) et les contraintes des figures V.4[1] et V.5 (page 102). Nous avons représenté dans la figure V.9 les résultats[2] tels qu'ils sont fournis par le prototype actuel, au code de couleurs près qui ne peut être rendu que de façon approximative par les différents grisés apparaissant dans ce document. Cette figure représente les objets sur lesquels portent des contraintes de protection (*A*, *E*, *F*, *confidentiel*). A chacun de ces objets est associée une boîte où apparaît, pour chaque ensemble d'attaquants, le temps moyen nécessaire pour acquérir les privilèges des cibles associées à cet objectif[3]. Ces valeurs sont triées en ordre croissant.

V.2.2 Etude de la situation initiale

L'analyse des résultats poursuit un double objectif : (i) identifier les groupes qui menacent plus particulièrement certains objets et (ii) appréhender les parties critiques du graphe afin de trouver un ensemble de modifications susceptibles d'améliorer la sécurité globale sans que la facilité d'utilisation en pâtisse.

(1) Les groupes les plus menaçants pour les différentes contraintes sont *PROJ1* (*A, B, D,* P_1), *PhD* (*C* et *B*) et *SYSTEM* (*F* et *G*). L'étude du graphe mise en corrélation avec cette remarque nous indique que c'est la possibilité pour *B* (appartenant au rôle *PhD*)

1. Dans cette contrainte, nous avons limité le domaine de valeurs de $X à l'ensemble {*A,E,F*}.
2. Par souci de concision nous ne parlons que de la variable "temps" dans ce qui suit. Le modèle mathématique sous-jacent étant le même pour la variable "effort", les problèmes liés à l'interprétation des résultats sont similaires.
3. Rappelons que, comme cela a été expliqué à la page 101, les contraintes portent sur des objets. Les détenteurs de privilèges sur ces objets constituent les cibles utilisées durant la phase d'évaluation.

d'acquérir les privilèges de *G* (appartenant au rôle *SYSTEM*) qui permet aux autres membres du rôle *PROJ*1 d'avoir une telle influence sur les mesures de sécurité.

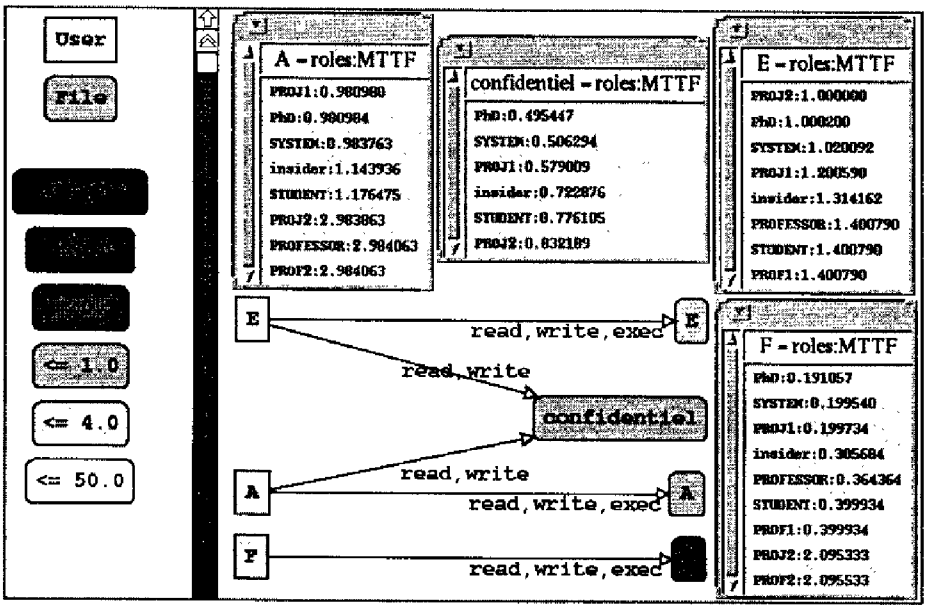


Figure V.9 Affichage des résultats pour notre exemple de travail

(2) Le temps moyen mis par le rôle *PROJ*1 pour atteindre l'objet *A* est de 0.98, soit un peu moins d'une semaine. Cette valeur montre l'influence prépondérante du chemin *B-filtre-G-*X_{admin}*-A* dont le *MTTF*, s'il avait été calculé, vaudrait un peu plus de 1.

(3) Pour ce même objet *A*, comparons les valeurs associées aux rôles *STUDENT* (*D*) et *insider* (*interne*) pour atteindre *A*. Tout d'abord, au vu de la remarque précédente, il est évident que, dans le graphe des privilèges, le chemin de *D* à *A* "passe" par le nœud *B*. Le temps que met *D* pour atteindre *A* est donc égal au temps mis par *B* (0.98) plus le délai nécessaire pour arriver en *B* (0.0002+0.2). La valeur obtenue (1.176) est très proche de cette somme. En revanche, la valeur associée à *insider* est plus faible (1.143) ce qui témoigne de l'influence d'un deuxième chemin, d'une longueur supérieure, qui permet d'atteindre *A*.

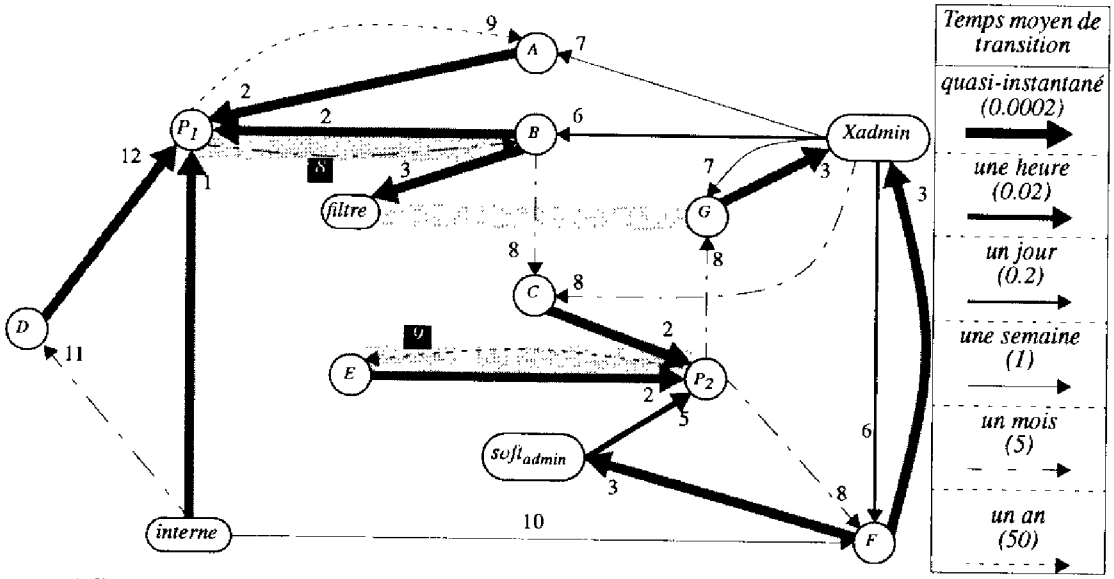
(4) La valeur très faible du temps que met le rôle *insider* pour menacer *F* (0.305) nous indique de façon claire que le danger ne réside pas tant dans le chemin constitué du seul arc *interne-F*, mais plutôt dans l'existence du chemin *interne-*P_1*-B-filtre-G-*X_{admin}*-F*. Cependant, la valeur calculée est plus petite que la valeur de ce chemin pris seul et nous pouvons donc en déduire l'existence d'autres chemins suffisamment courts pour influencer la valeur finale.

(5) Les valeurs associées à l'objet *confidentiel* confirment que la menace est directement proportionnelle au nombre de cibles. En effet, les droits sur cet objet sont détenus par *A* et par *E*. Il suffit donc d'obtenir les privilèges de l'un d'eux pour violer l'objectif de sécurité. L'existence de plusieurs chemins différents vers chacune des cibles augmente la chance de réussite de l'assaillant. Par exemple, le groupe *PhD* (*C* et *B*) peut acquérir les privilèges de *A* en à peu près une semaine (0.98) et ceux de *E* dans le même laps de

temps (1.0002). En revanche, moins d'une demi-semaine (0.495) lui est nécessaire pour acquérir les droits de l'un ou de l'autre et détenir ainsi les droits de lecture et écriture sur l'objet *confidentiel*.

(6) De façon similaire, l'existence de plusieurs chemins différents entre un attaquant et une cible influence fortement la sécurité. Par exemple, le rôle *PROF*2 qui ne contient que l'utilisateur *E*, parvient à acquérir les privilèges de *F* en un peu plus de deux semaines (2.095) alors qu'aucun chemin entre *E* et *F* n'a une longueur inférieure à 4 semaines (puisqu'ils doivent passer soit par l'arc P_2-*F*, soit par l'arc P_2-*G*).

Pour poursuivre ces réflexions, nous allons modifier le graphe de privilèges et regarder si les *MTTF* qui en découlent répondent à nos attentes. En particulier, nous allons diminuer l'importance du point de passage privilégié qu'est l'utilisateur *B* en supprimant l'arc *filtre-G* et en diminuant la valeur du taux associé à l'arc P_1-*B*.



1) Y n'a pas de mot de passe ; 2) X est présent dans le .rhosts de Y ; 3) X est un sur-ensemble de Y ; 4) X attaque Y par le biais du courrier électronique ; 5) Y utilise, une fois par heure, un logiciel géré par X ; 6) Y utilise, une fois par jour, un logiciel géré par X ; 7) Y utilise, une fois par semaine, un logiciel géré par X ; 8) Y utilise, une fois par mois, un logiciel géré par X ; 9) Y utilise, une fois par an, un logiciel géré par X ; 10) mot de passe de Y devinable en une semaine ; 11) mot de passe devinable en un mois ; 12) X peut modifier l'exécution d'un programme dont le setuid bit est positionné appartenant à Y.

Figure V.10 Exemple : première modification

V.2.3 Première modification

A. Graphe des privilèges

De façon concrète, modifier le taux de transition associé à l'arc P_1-*B* revient à demander à *B* de changer ses habitudes de travail. Plutôt que d'exécuter, sous sa propre identité, les logiciels développés par le projet *PROJ*1, l'administrateur peut lui demander de les exécuter sous l'identité de P_1, grâce au fichier *.rhosts*. Le graphe permet de simuler le comportement de *B* en

modifiant la valeur de l'arc et offre ainsi des arguments objectifs à l'administrateur pour formuler sa requête.

Supprimer l'arc *filtre-G* se ramène à interdire à *G* d'utiliser le filtre de courrier électronique que *B* a installé. Enfin, nous avons également modifié le taux de transition associé à l'arc P_2-*E*. Toutes ces modifications sont reprises sur le graphe de la figure V.10 où nous avons indiqué par un rectangle grisé les endroits où des arcs avaient été modifiés ou supprimés.

B. Résultats

Après avoir reporté ces modifications sur le graphe de Markov, nous pouvons recalculer les *MTTF* désirés. Les nouveaux résultats sont repris à la figure V.11 ; ils nous permettent de mettre quelques résultats intéressants en avant :

(1) De façon non surprenante, aucun rôle ne peut acquérir les privilèges de *E* en moins d'un an. Ceci est évident puisque tous les chemins passent par l'arc P_2-*E* qui a un taux de transition très faible.

(2) Pour ce même objet, certaines valeurs sont égales à la longueur du plus court chemin entre les attaquants et *E*. Ceci nous convainc de ce qu'il n'existe qu'un chemin entre cette cible et ses attaquants (*PROFESSOR* (*A* et *E*), par exemple, ne peut atteindre *E* que par le chemin *A*-P_1-*B*-*C*-P_2-*E* : 58.0004=0.0002+4+4+0.0002+50).

(3) Le répertoire personnel de *A* est plus menacé par le rôle *PROJ2* (*C*, *E*, P_2) que par le rôle *PROJ1* (*A*, *B*, *D*, P_1). Ce résultat est très gênant dans la mesure où, *a priori*, *A* a plus de raisons de faire confiance à des gens du projet *PROJ1* (auquel il participe) qu'à des personnes qu'il ne connaît normalement pas. La raison de cette menace réside dans la facilité relative qu'ont les gens du groupe *PROJ2* d'acquérir les privilèges des membres du rôle *SYSTEM* (*F* et *G*) qui, eux, menacent fortement *A*.

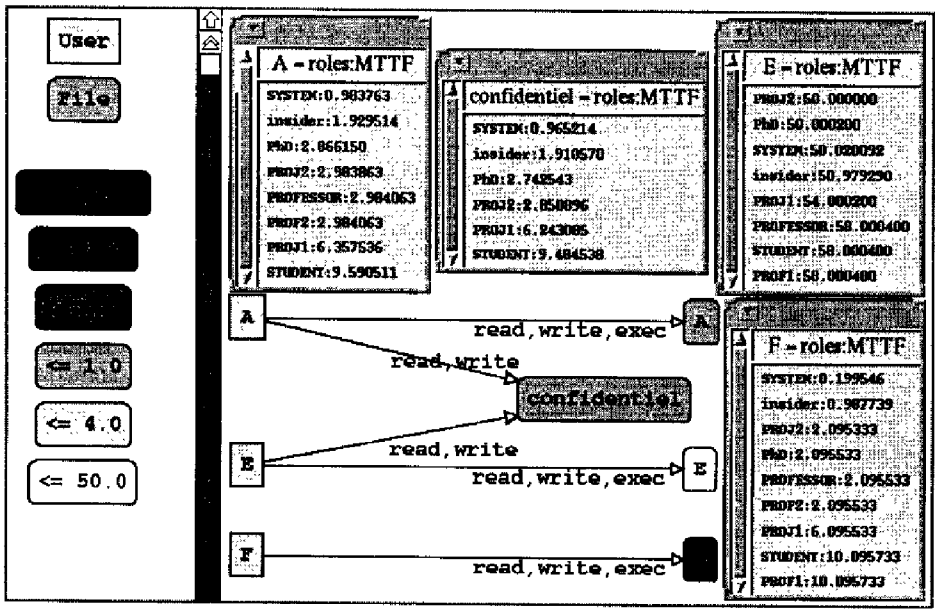


Figure V.11 Affichage des résultats suite à la première modification

(4) Le rôle *insider* menace également fortement *A* et la comparaison avec le résultat précédent est très intéressante. Dans la situation initiale, le plus court chemin entre *interne* et *A* passait par *B*. Une fois ce chemin coupé, l'évaluation met à présent en exergue l'existence d'un autre chemin, passant par *F* qui est pratiquement aussi dangereux. Ceci corrobore tout à fait les conclusions auxquelles nous étions arrivé précédemment sur l'existence de deux chemins partant du nœud *interne* vers le nœud *F* (point 3, page 107).

(5) De même, la valeur associée au rôle *insider* pour atteindre l'utilisateur *F* montre que, maintenant, le chemin le plus court passe par l'arc *interne-F* et en est du même ordre de grandeur.

Tirant parti de ces remarques, nous allons tenter de trouver une configuration qui protège encore mieux les privilèges de l'équipe *SYSTEM* (*F* et *G*), puisqu'ils servent, apparemment, à de nombreuses attaques. De plus, nous allons mieux isoler l'utilisateur *interne* des autres cibles.

V.2.4 Deuxième modification

A. Graphe de privilèges

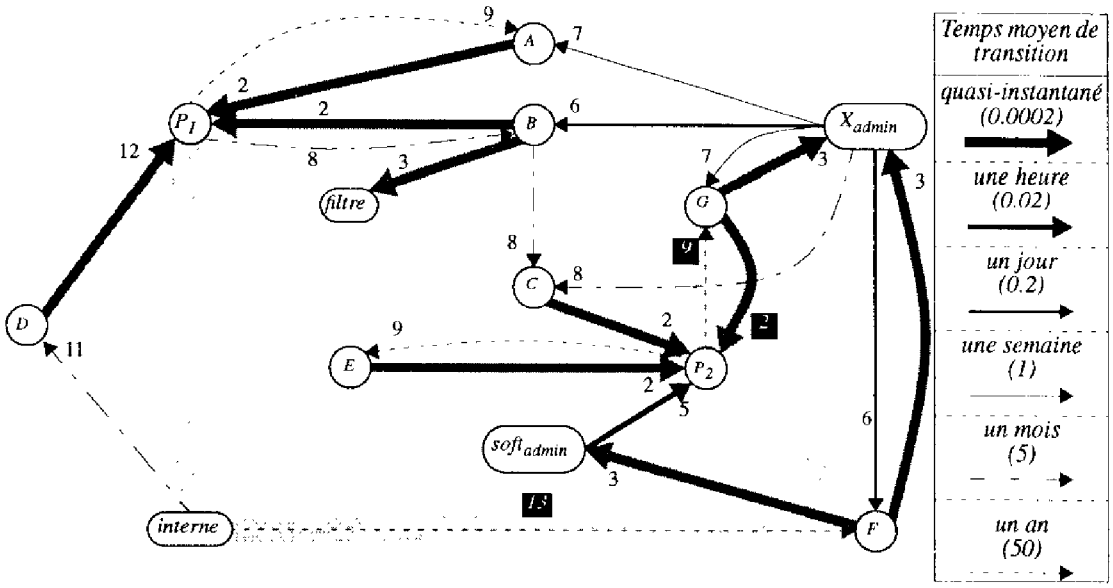
F et *G* utilisent régulièrement un logiciel géré par P_2. Cet usage se révèle être préjudiciable à la sécurité générale. Deux solutions sont possibles : i) supprimer les arcs revient à interdire l'utilisation du logiciel géré par P_2 ; (ii) réduire l'importance de l'arc impose à *F* et *G* de se servir moins souvent de ce logiciel.

Ces deux solutions sont contraignantes puisqu'elles restreignent la liberté de *F* et *G*. Nous pouvons aussi imaginer de mettre *G* dans le fichier *.rhosts* de P_2, ce qui permet à *G* de se servir, sans danger, du logiciel de P_2. Cette nouvelle configuration est représentée dans la figure V.12 où nous avons également supprimé l'arc *interne-*P_1(en supprimant la possibilité pour P_1 de se connecter directement en donnant son mot de passe) et diminué le taux de transition *interne-F* (en demandant à *F* de choisir un meilleur mot de passe).

Une telle solution n'est envisageable que si nous pouvons, d'autre part, être sûr qu'elle n'a pas une influence négative sur d'autres contraintes de protection. Cette évaluation est possible grâce au prototype. Nous en donnons les résultats dans la figure V.13.

B. Résultats

L'inspection des résultats présentés dans la figure V.13 nous montre que la sécurité du système a été très fortement améliorée par rapport au système de départ, sans que nous ayons apporté de très grandes modifications au graphe. Seul le groupe *SYSTEM* (*F* et *G*) peut encore violer les objectifs de protection relativement facilement. Ceci matérialise la notion de confiance que l'on place toujours dans les administrateurs du système. Grâce à l'évaluation quantitative,



1) Y n'a pas de mot de passe ; 2) X est présent dans le .rhosts de Y ; 3) X est un sur-ensemble de Y ; 4) X attaque Y par le biais du courrier électronique ; 5) Y utilise, une fois par heure, un logiciel géré par X ; 6) Y utilise, une fois par jour, un logiciel géré par X ; 7) Y utilise, une fois par semaine, un logiciel géré par X ; 8) Y utilise, une fois par mois, un logiciel géré par X ; 9) Y utilise, une fois par an, un logiciel géré par X ; 10) mot de passe de Y devinable en une semaine ; 11) mot de passe devinable en un mois ; 12) X peut modifier l'exécution d'un programme dont le setuid bit est positionné appartenant à Y ; 13) mot de passe devinable en un an ;

Figure V.12 Exemple : deuxième modification

nous sommes en mesure de faire apparaître cette notion de façon explicite. Quelques remarques sont importantes :

(1) L'importance des chemins multiples est plus nette. Nous le voyons par exemple pour les rôles *insider*, *PhD* (*B* et *C*), *PROJ1* (*A*, *B*, *D*, P_1) et *STUDENT* (*D*) qui menacent *A*. Tous les chemins qu'ils peuvent suivre ont une "longueur" supérieure à 50 (ils doivent passer soit par P_1-*A*, soit par P_2-*G*, soit par *interne*-*F*). Pourtant, le temps moyen global est inférieur à 30. Ceci est dû à l'existence de plusieurs chemins dans le graphe. Ceci nous amène à penser qu'un système dont tous les taux de transition associés aux arcs sont faibles n'en est pas sûr pour autant.

(2) En revanche, lorsque tous les chemins doivent emprunter un même arc, son poids n'est pas divisé par les chemins qui y mènent ou en sortent. Ceci est clairement montré par les valeurs associées aux rôles qui menacent *F*. Tous les rôles, *insider* excepté, doivent emprunter l'arc P_2-*G* et son importance est prépondérante dans le calcul final du *MTTF*. Modifier le taux de transition associé à ce type d'arc est donc une façon très efficace d'améliorer la sécurité du système global.

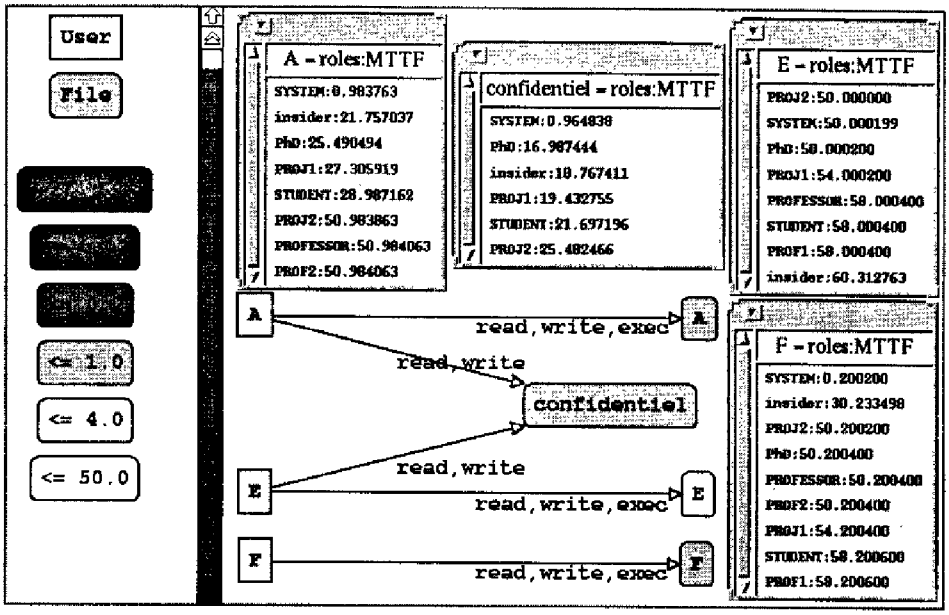


Figure V.13 Affichage des résultats suite à la deuxième modification

V.3 Avantages et limites du prototype actuel

V.3.1 Avantages

A. Etude de la sécurité opérationnelle

Une fois les objectifs de protection définis, les différentes phases de l'évaluation ne nécessitent pas d'intervention de la part de l'opérateur. Elles peuvent se dérouler de façon totalement automatique. Il est donc possible d'utiliser notre prototype pour évaluer chaque jour un ensemble de mesures et réaliser de la sorte un suivi de la sécurité opérationnelle du système. Une dégradation des résultats observés mettra en exergue l'introduction de nouvelles fautes au sein du système. Une étude approfondie des résultats permettra alors de savoir s'il s'agit de fautes intentionnelles malveillantes (une intrusion en cours), non malveillantes (un changement d'habitude de certains utilisateurs) ou de fautes accidentelles (par exemple, une modification intempestive de la configuration du système).

B. Modularité

Comme nous l'avons expliqué, le traitement des résultats est réalisé à l'aide d'un ensemble de filtres et débouche sur une présentation graphique, synthétique, de la sécurité du système. Ces filtres peuvent être définis en fonction des besoins de l'utilisateur. En particulier, si l'outil doit servir à étudier la sécurité opérationnelle, ils devront comparer les résultats journaliers aux résultats antérieurs afin de mieux faire ressortir les variations constatées. L'outil a été conçu pour permettre d'implémenter très facilement ce genre de traitements.

C. Outil de déduction et de simulation

L'outil peut servir à étudier les solutions proposées par l'administrateur de sécurité. En effet, il est possible de modifier le graphe des privilèges (normalement construit automatiquement à partir du système de fichiers) afin d'évaluer l'efficacité d'une solution envisagée. C'est le type de raisonnement que nous avons mené dans la section V.2. De plus, une définition judicieuse des différents rôles doit permettre l'identification des parties critiques du graphe de privilèges. En effet, dans certains cas de figure comme ceux discutés dans la section V.2, il est possible, par une analyse minutieuse des résultats, de déduire les modifications qui réalisent le meilleur compromis entre les contraintes imposées aux utilisateurs et la sécurité globale du système. C'est ce que nous avons tenté d'illustrer dans les pages qui précèdent. Il faut reconnaître, toutefois, qu'une certaine expertise reste nécessaire et que l'outil gagnerait à offrir une aide de ce point de vue.

V.3.2 Limites

A. Temps de calcul

Le prototype actuel est utilisable et a été expérimenté. Nous l'avons voulu simple d'usage et facilement modifiable. Nous l'avons réalisé à l'aide d'outils existants que nous avons intégré dans un environnement logiciel cohérent. Il répond à nos besoins, mais pourrait être amélioré du point de vue de son efficacité algorithmique. En particulier, nous avons voulu utiliser toute la richesse de l'outil Surf-2 pour réaliser les calculs. Ce faisant, nous ne profitons pas de la structure très particulière des graphes de Markov que nous manipulons. Il serait possible, par exemple, de définir un algorithme de calcul du *MTTF* beaucoup plus efficace, mais moins général en tenant compte de la structure d'arbre qu'ont les graphes de Markov générés à partir du graphe des privilèges.

B. Spécification limitée des objectifs de protection

Le langage de contraintes de Miró est très riche comme nous l'avons mentionné. Pour des raisons touchant à des problèmes d'implémentation, nous n'en avons utilisé qu'une faible partie en ne considérant que des contraintes suffisamment simples pour permettre l'identification rapide des cibles qui nous intéressaient. L'utilisation de toute la puissance du langage demande une réflexion plus poussée sur les cibles à considérer dans tous les cas de figure ainsi que sur la façon la plus pertinente de présenter les résultats de l'évaluation. *A contrario*, si nous nous contentons de ce seul type de contraintes, il est probable que l'algorithme général de vérification des contraintes pourrait être grandement optimisé.

C. Intégration de NIDES

En l'état actuel, le prototype utilise une version remaniée de l'outil *ASA* [Dacier et Rutsaert 1991a][36] pour construire le graphe des privilèges. *ASA* n'est pas encore interfacé avec *NIDES* [Anderson et al. 1993][5] et, de ce fait, utilise des approximations basées sur

les classes d'attaques pour attribuer les valeurs des taux de transition. Pour pouvoir obtenir de manière automatique des résultats aussi précis que ceux discutés dans la section V.2, il faut développer un nouveau prototype qui utilise les profils statistiques produits par *NIDES* pour affiner la définition des taux. Il serait également intéressant d'inclure alors dans le graphe des privilèges la notion de réseaux de machines interconnectées qui lui fait encore défaut.

V.4 Conclusions

Dans ce chapitre, nous avons illustré les différentes phases du processus d'évaluation quantitative de la sécurité. Nous avons montré comment ces étapes avaient été implémentées au sein d'un prototype expérimental et nous avons expliqué son fonctionnement. Nous avons choisi un exemple simple afin de montrer l'enchaînement des différents outils utilisés, depuis la construction du graphe de privilèges et la spécification de la sécurité, jusqu'à l'affichage des résultats. Nous avons ainsi pu mettre en lumière comment le modèle permet la déduction de solutions et la simulation de leurs conséquences. Les limites du prototype actuel ont également été mentionnées.

Conclusion Générale

1. Démarche

En guise de conclusion, nous nous proposons de rappeler tout d'abord en quelques phrases la démarche adoptée pour mener à bien le travail exposé. Ensuite, nous dégageons les principaux résultats obtenus et terminons en proposant plusieurs pistes de recherche intéressantes pour dépasser leurs limites actuelles.

Au cours de ce mémoire, nous avons voulu adopter une démarche de pensée qui soit à la fois *globale*, *rigoureuse* et *pragmatique*. Ce triple objectif a influencé le document final et il nous semble intéressant, à ce stade, de le reconsidérer selon ces trois points de vue.

Notre volonté de *globalisation* nous a amené à envisager les différentes écoles de pensée qui s'intéressent au problème de l'évaluation de la sécurité. C'est pourquoi nous avons présenté les différents critères d'évaluation internationaux, les méthodes d'analyse des risques et les modèles formels de sécurité. Cependant, nous avons voulu dépasser la simple revue de l'état de l'art pour offrir une lecture nouvelle de ces travaux. Dans ce sens, nous avons montré que la nécessité de recourir à des méthodes de prévision des fautes transparaissait dans les critères les plus récents d'évaluation. Une telle façon de les présenter n'est, certainement pas la plus classique. De même, nous avons voulu éviter l'écueil de l'exposé fastidieux des différentes méthodes d'analyse des risques et nous avons préféré nous pencher sur la raison profonde, à savoir le manque de modélisation des systèmes étudiés, qui empêche un essor plus grand de ces méthodes en dehors de cette seule communauté. Enfin, nous avons présenté différents modèles formels, en gardant à l'esprit les résultats issus des autres approches. Nous avons ainsi mis en avant l'hypothèse de pire cas sous-jacente à ces modèles sur le comportement des utilisateurs du système étudié.

Ces analyses nous ont permis de discerner les points de conflit entre les différentes écoles de pensée mais également les contributions que chacune d'entre elles pourrait apporter à une méthode d'évaluation globale :

- Des critères d'évaluation, nous retenons qu'ils ont évolué au fil des ans et reconnaissent aujourd'hui l'utilité et la nécessité de recourir à des méthodes de prévision des fautes pour évaluer la sécurité des systèmes informatiques.
- La revue des méthodes d'analyse des risques nous permet d'isoler les caractéristiques essentielles d'une méthode idéale de prévision des fautes.
- Enfin, l'étude des modèles formels nous livre des pistes de réflexion intéressantes pour leur intégration dans ce type d'approche.

Fort de cette réflexion globale, nous avons ensuite défini un nouveau modèle formel, le graphe des privilèges, et montré comment l'intégrer au sein d'une méthode d'évaluation quantitative de la sécurité. Afin de bâtir notre solution avec autant de *rigueur* que possible, nous avons d'abord défini formellement le graphe des privilèges puis expliqué comment le transformer en un réseau de Petri afin de modéliser le processus d'intrusion dans un système. Enfin, nous avons prouvé les propriétés mathématiques des résultats que nous étions à même d'obtenir à l'aide de ce cadre mathématique.

Au-delà de ce souci de rigueur, nous avons tenu à choisir une *approche pragmatique*. C'est dans cet esprit que nous avons montré, dans le cadre du modèle *take-grant*, comment relâcher l'hypothèse de pire cas adoptée par ce modèle sur le comportement des utilisateurs. De même, nous avons été amené à définir le graphe des privilèges afin de répondre à un problème d'efficacité au regard de la complexité algorithmique des autres solutions proposées. Enfin, c'est pour pouvoir valider les hypothèses théoriques en les confrontant à la réalité que nous avons développé un prototype d'expérimentation de notre méthode.

Notre travail laisse de nombreuses questions sans réponses et, paradoxalement, c'est peut-être là un de ses principaux apports. En effet, toute réponse n'est que la source de nouvelles questions et si, aujourd'hui, certaines perspectives de recherche se révèlent à nous, c'est aussi, sans doute, en raison des réponses qu'a fournies notre travail. Il n'est donc pas inutile de revenir brièvement sur les résultats obtenus.

2. *Résultats*

Dans ce mémoire, nous avons montré la *nécessité* et la *possibilité* de ne plus considérer la sécurité des systèmes d'information comme une simple propriété binaire, de type tout ou rien.

La présentation originale des critères d'évaluation et des méthodes d'analyse des risques a indiqué pourquoi une telle approche était nécessaire. Elle nous a amené également à la conclusion selon laquelle la définition d'un modèle formel de représentation des systèmes et de leur sécurité était un prérequis à toute définition d'un modèle d'évaluation.

Nous avons rendu possible la définition d'un tel modèle en apportant deux contributions personnelles au domaine des modèles formels de sécurité :

(1) Le premier se situe dans la définition d'un nouveau modèle, basé sur un réseau de Petri, qui permet de relâcher l'*hypothèse de pire cas* sur le comportement des attaquants adopté par le modèle *take-grant*.

(2) Le second apport consiste en la définition d'un nouveau modèle, le *graphe des privilèges*, plus efficace que le modèle *TAM*, dans le cas de certains schémas d'autorisation bien définis.

De plus, nous avons appliqué ce modèle dans le cadre des systèmes Unix et montré comment le même modèle pouvait ensuite être utilisé en tant que *modèle d'évaluation quantitative de la sécurité opérationnelle*. Nous avons également expliqué les principes d'un prototype que nous avons réalisé et qui permet la mise en œuvre de la méthode.

La méthode d'évaluation que nous avons définie est une pierre capitale dans l'édifice de notre argumentation en faveur d'une approche nouvelle de la sécurité. Cependant, nous estimons qu'elle vaut plus par les questions qu'elle ne peut manquer de susciter que par son existence elle-même. Plus qu'une fin en soi, notre méthode est l'illustration des idées défendues dans ce document. Nous espérons qu'elle pourra servir de base de réflexion pour des travaux ultérieurs et, dans ce but, nous concluons en esquissant quelques directions de recherche qui nous semblent intéressantes.

3. *Perspectives*

Tout d'abord, nous pensons que le prototype en lui-même peut utilement s'enrichir de travaux dans quatre directions différentes mais complémentaires :

- Une réelle expérimentation sur un site de test permettrait d'évaluer ses qualités en tant qu'outil d'aide à la détection d'intrusions. Un tel usage nécessite, au préalable, d'intégrer l'outil actuel dans un système de détection d'intrusions et de lui fournir une liste plus riche de méthodes de cession de privilèges. Une étude de sensibilité des variables "temps" et "effort" sera alors possible.
- D'autres mesures que le *MTTF* pourraient également être étudiées. En particulier, il pourrait être intéressant d'étudier non plus le temps moyen pour qu'un attaquant parvienne à une cible, mais la distribution de probabilité qu'il y parvienne en fonction du temps ou de l'effort dépensé. L'interprétation de telles courbes n'est pas simple, mais devrait offrir une vision plus fine de l'état de la sécurité du système.
- Il peut être intéressant d'envisager d'autres types de distribution de probabilité afin de dépasser le cadre markovien. Toutefois, une telle modification du modèle nécessite, au préalable, de vérifier que les nouveaux résultats satisfont toujours les propriétés intuitives décrites dans le corps du texte.
- Enfin, puisque nous modélisons l'incertitude liée au processus d'attaque d'un intrus, il pourrait être intéressant d'examiner l'apport possible d'autres formalismes utilisés pour représenter l'incertain (théorie des possibilités [Zadeh 1978][148], théorie des fonctions de croyance [Shafer 1976][125]).

Ensuite, il est utile de conserver à l'esprit que nous n'avons abordé ici que le seul problème de l'évaluation de la sécurité. La problématique plus générale de la gestion de la sécurité se heurte encore à plusieurs problèmes non résolus :

(1) Comment *spécifier* les objectifs de sécurité ? Nous avons brièvement évoqué ce problème lors de l'explication du langage Miró. Dans notre cadre d'application, nous

pouvions nous contenter de ce formalisme. Cependant, nous avons indiqué ses limitations ainsi que l'intérêt d'étudier d'autres types de représentation.

(2) Comment *gérer* l'ensemble des résultats obtenus ? En effet, l'inspection systématique, journalière des valeurs obtenues peut s'avérer être une tâche insurmontable dans le cas où le nombre d'objectifs est important. D'autres solutions doivent alors être proposées pour gérer cette somme d'informations mise à la disposition de l'administrateur du système.

(3) Comment *mettre en œuvre* la sécurité ? En termes de graphes de privilèges, ceci revient à trouver les arcs à modifier pour assurer la satisfaction des objectifs de sécurité. Nous avons expliqué qu'une étude attentive des résultats de l'évaluation pouvait aider à l'identification des modifications les plus efficaces. Cependant, nous avons également admis le besoin de définir des méthodes d'analyse du graphe pour automatiser cette tâche. Il devrait être possible de tirer parti du fait que nos graphes sont des arbres afin de dériver l'expression analytique des mesures intéressantes et raisonner sur ces équations afin de déterminer l'influence des différentes variables. De plus, il est clair qu'il faut d'abord retirer du graphe les arcs qui représentent des fautes intentionnelles malveillantes ou accidentelles (puisqu'ils n'apportent aucune fonctionnalité utile aux utilisateurs bien intentionnés). Enfin, l'approche analytique devra être menée de telle sorte que soient prises en compte les contraintes imposées aux utilisateurs par les modifications proposées : les mesures les moins contraignantes devront être préférentiellement choisies. Ce type d'approche sera d'autant plus utile que les graphes étudiés seront grands et fortement connexes.

Il n'est pas anodin que nous terminions cette thèse par autant de questions sans réponses. Nous avons ainsi voulu montré le vaste champ d'exploration qui s'offre à nous dès lors que la sécurité ne se résume plus à une simple propriété de type tout ou rien et que l'on prend également en considération l'importance de la vie opérationnelle. Une telle approche de la sécurité est nécessaire et possible. Nous espérons l'avoir démontré.

Annexe A

Retranscription du modèle TG en un réseau de Petri

Dans cette annexe, nous détaillons le processus de transformation d'un modèle *Take-Grant* en un réseau de Petri [Dacier 1993][39]. Ce texte vient en complément du document principal ; les définitions et notions présentées dans le chapitre II sont supposées connues.

A.1. Définition du problème

Le problème que nous cherchons à résoudre à l'aide de cette transformation s'exprime de la façon suivante : *Dans le modèle take-grant, soient un graphe de protection* Γ_0, *un ensemble* C *de sujets (les conspirateurs), un attaquant* P ($P \in C$) *et deux nœuds* R *et* Q *tels qu'il existe un arc étiqueté* α *de* R *vers* Q *dans* Γ_0 *(noté par la suite* $R - \alpha \rightarrow Q$ *dans* Γ_0*), montrez s'il existe ou non une séquence de graphes* $\Gamma_0, \Gamma_1, \ldots \Gamma_n$ *telle que (i)* $P - \alpha \rightarrow Q$ *dans* Γ_n *et (ii)* $\forall X$: X *applique une règle* r_i *de réécriture du graphe* $\Rightarrow X \in C$, ($i \in [1,\ldots,n]$)

Si une telle séquence existe, nous disons que "*P peut atteindre Q*". Notre solution à ce problème passe par trois phases distinctes:

(1) Transformation du graphe de protection Γ_0 en un réseau de Petri équivalent, noté Π.
(2) Identification de l'ensemble C des conspirateurs par le biais du marquage de places idoines de Π.
(3) Exécution d'une séquence maximale de tirs.

Au bout de ces trois étapes, la présence d'un jeton dans au moins une place R de Π, telle que $R - \alpha \rightarrow Q$ dans Γ_0, induit que "*P peut atteindre Q*". Formellement: *P peut atteindre Q si et seulement si* $\exists R \quad R - \alpha \rightarrow Q$ *dans* Γ_0 *et*[1] $m_f(R) > 0$. Avant d'entrer dans les détails des règles de la transformation, nous expliquons ces étapes dans les deux prochaines sections.

A.2. Le processus global de résolution

A.2.1 Construction du réseau de Petri

Chaque nœud N de Γ est représenté par deux places dans Π. Elles sont notées N et N_c. Chaque droit apparaissant dans l'étiquette d'un arc de Γ donne lieu à la construction d'une structure particulière dans Π.

1. $m_f(X)$ dénote le marquage final de X, i.e. le nombre de jetons dans la place X, après le tir d'une séquence maximale.

A.2.2 Identification de l'ensemble C des collaborateurs

Le marquage initial des places N est égal à zéro pour tous les nœuds, exception faite du nœud correspondant à P (l'attaquant) pour lequel il vaut un. Ce jeton joue un rôle particulier par la suite et nous y faisons référence sous le nom de **jeton-A** (A pour Attaquant). Le marquage initial des places N_c vaut un pour tous les nœuds N appartenant à C, l'ensemble des collaborateurs potentiels. Ces jetons sont appelés par la suite les **jetons-C** (C pour collaborateur). Le marquage des places N_c vaut zéro pour tous les nœuds qui n'appartiennent pas à C.

Ainsi, grâce au marquage des places N_c, nous intégrons dans le réseau de Petri la connaissance que l'on a sur l'identité des collaborateurs potentiels. Formellement, nous avons : $\forall N, N \neq P \Rightarrow m_0(N) = 0$; $N = P \Rightarrow m_0(N) = 1$ et $\forall N, N \in C \Rightarrow m_0(N_c) = 1$; $\forall N, N \notin C \Rightarrow m_0(N_c) = 0$.

A.2.3 Tir d'une séquence maximale

Le réseau de Petri est construit de telle sorte que toute séquence de tir maximale soit de longueur finie et mène à la même solution. Trouver la valeur de vérité du prédicat "*peut atteindre*" revient à propager le jeton-A jusqu'à une place spécifique (R). Alors que le graphe de protection Γ_0 évoluait par l'ajout de nouveaux arcs entre les nœuds, le réseau de Petri évolue en modifiant son marquage. Avant d'expliquer les règles de construction en détail, nous expliquons la différence entre le jeton-A et le jeton-C.

A.3. Différence entre les jetons-A et les jetons-C

Pour formaliser notre propos, nous utilisons quelques conventions d'écriture décrites ci-après. Soit un *alphabet* composé des quatre symboles suivants: $\{\vec{t}, \vec{g}, \overleftarrow{t}, \overleftarrow{g}\}$. Un *mot* est une suite quelconque de symboles de l'alphabet. Un symbole de l'alphabet affecté d'une *étoile (*)* représente une séquence de *n* symboles, *n* valant entre *zéro* et l'infini. Par exemple $\vec{g}\ \vec{t}^{\,*}$ correspond aussi bien au mot $\vec{g}\ \vec{t}$, qu'à $\vec{g}\ \vec{t}\ \vec{t}\ \vec{t}$ ou à $\vec{g}$. Un symbole de l'alphabet affecté d'un *plus (+)* représente une séquence de n symboles, n valant entre *un* et l'infini. Par exemple $\vec{g}\ \vec{t}^{\,+}$ correspond aussi bien au mot $\vec{g}\ \vec{t}$, qu'à $\vec{g}\ \vec{t}\ \vec{t}\ \vec{t}$ mais pas à $\vec{g}$. Une façon aisée de définir des familles de mots est d'utiliser le symbole de disjonction "|". Ainsi, le mot $\vec{g}\ [\vec{t}\,|\overleftarrow{t}]\ \overleftarrow{g}$ désigne les mots $\vec{g}\ \vec{t}\ \overleftarrow{g}$ et $\vec{g}\ \overleftarrow{t}\ \overleftarrow{g}$.

Dans Γ, nous pouvons associer des mots de l'alphabet $\{\vec{t}, \vec{g}, \overleftarrow{t}, \overleftarrow{g}\}$ à chaque chemin entre deux nœuds P et R. A titre d'exemple, la figure A.1 représente un graphe de protection comportant une infinité de chemins entre P et R à laquelle on peut associer le mot $\overleftarrow{g}\ ((\vec{g}\ \overleftarrow{g})^*(\vec{t}\ \overleftarrow{t})^*)^*\vec{g}$

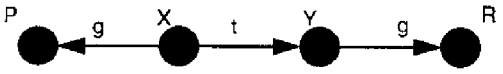


Figure A.1 Exemples de chemin entre P et R

Si P ne dispose d'aucune aide, c'est-à-dire si aucun autre nœud ne peut appliquer de règle dans le modèle TG, le prédicat "*P peut atteindre Q*" ne sera satisfait que si on peut associer le mot $\vec{t}^{+}$ à au moins un chemin entre P et R. Ceci est dû au fait que seule la règle "*take*", initiée par P, est à même d'augmenter les privilèges qu'il possède au départ et donc de l'aider à atteindre Q. Dès le moment où P "rencontre" autre chose qu'un arc $\vec{t}$ sur son chemin, il ne peut avancer seul.

De même, nous construisons le réseau de Petri de telle sorte que le jeton-A puisse se propager jusqu'aux places R telles qu'il existe au moins un chemin entre P et R auquel on puisse associer le mot $\vec{t}^{+}$. Si nous transformons chaque arc "*take*" par la construction représentée à la figure A.2 nous sommes certains de respecter cette propriété.

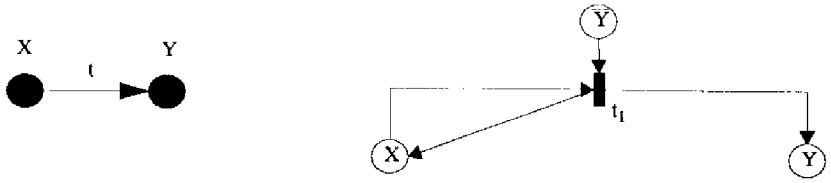


Figure A.2 Réécriture partielle d'un arc *take* pour les jetons-A

Dans cette figure, la place $\bar{Y}$ est une **place de garde.** Son marquage initial vaut un ce qui nous assure que la transition t_1 ne peut être tirée qu'une seule fois. D'autre part, nous voyons que la place X sert à la fois de place d'entrée et de sortie. La raison en est qu'il pourrait y avoir plusieurs chemins partant de X que le jeton-A pourrait emprunter. Dès lors, après avoir tiré la transition, nous remettons un jeton en X afin qu'il puisse servir pour d'autres transitions. Nous reviendrons plus en détail sur ces deux points lorsque nous envisagerons le problème de l'évaluation de la complexité.

Cette construction ne tient pas compte des places N_c. Cependant, il est raisonnable de penser que les collaborateurs, de leur côté, vont également essayer d'accroître leurs droits afin d'aider P. C'est pourquoi il est nécessaire de représenter la propagation des jetons-C le long des chemins $\vec{t}^{*}$. C'est ce qui est fait dans la figure A.3.

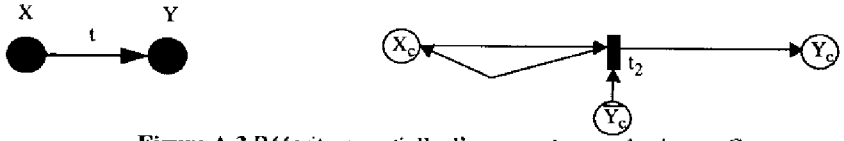


Figure A.3 Réécriture partielle d'un arc *take* pour les jetons-C

Si nous ne créons pas ces places N_c et choisissons de mettre un jeton dans la place N pour tout collaborateur, alors s'il existe un chemin $\vec{t}^{*}$ entre un collaborateur X et la cible Q, l'exécution d'une séquence maximale de tir aura pour effet de mettre un jeton dans Q et ce même s'il n'existe aucun chemin entre l'attaquant et la cible. C'est pourquoi nous utilisons deux types de jetons différents : les jetons-A et les jetons-C[1]. Nous insistons sur le fait que chaque arc "*take*" d'un modèle TG va être remplacé dans le réseau de Petri à la fois par la construction

1. Nous aurions pu utiliser un réseau de Petri coloré qui offre la possibilité de typer les jetons, mais nous avons préféré rendre la construction plus explicite en définissant des places qui, chacune, ont un rôle précis.

représentée dans la figure A.2 et par celle de la figure A.3. Ces constructions interagissent dans le réseau complet comme nous le montrons par la suite.

Maintenant que nous avons expliqué comment les Jetons-A et les jetons-C peuvent se propager le long de chemins $\vec{t}^{\,*}$, nous allons nous intéresser au cas plus général où un attaquant (un jeton-A) a besoin d'un collaborateur (un jeton-C) pour pouvoir franchir une transition. Pour cela, nous devons caractériser dans quelles circonstances un jeton-C peut aider à la propagation d'un jeton-A. En d'autres termes, nous pouvons voir le processus de conspiration comme la progression de deux jetons qui viennent à la rencontre l'un de l'autre. D'un côté, le jeton-A progresse jusqu'à être bloqué et le jeton-C vient vers lui afin de l'aider à franchir la transition bloquante. D'un point de vue abstrait, nous avons donc un chemin $\vec{t}^{\,*}?\overleftarrow{t}^{\,*}$. Le point d'interrogation symbolise la transition bloquante, le jeton-A y arrive par le chemin $\vec{t}^{\,*}$ tandis que le jeton-C vient à sa rencontre par le chemin $\overleftarrow{t}^{\,*}$. Il nous faut définir ce qu'est le point d'interrogation et ce qui peut s'y passer. Nous montrerons que si la collaboration réussit, c'est-à-dire si le jeton-C aide le jeton-A à passer au-delà du point de jonction, alors le jeton-A peut se propager sans nouvelle aide le long du chemin déjà parcouru par le jeton-C. S'il est à nouveau bloqué avant d'atteindre la cible, une nouvelle aide est requise selon le même scénario. Nous pouvons donc associer à tout chemin entre l'attaquant et la cible le mot $(\vec{t}^{\,*}?\overleftarrow{t}^{\,*})^*$ où chaque point d'interrogation représente une collaboration. Nous allons à présent nous attacher à caractériser ces collaborations et étudier ce qui s'y passe.

A.4. Le processus de conspiration

A.4.1 Les 3 jonctions possibles

Si un chemin existe entre P et R, le mot qui lui est associé est nécessairement de la forme: $((\vec{t})^*(\vec{g})^*(\overleftarrow{t})^*(\overleftarrow{g})^*)^*$ puisque cette expression générale reprend tous les mots possibles. Cette expression peut se réécrire sous la forme $((\vec{t})^*[\vec{t}\,|\,\vec{g}\,|\,\overleftarrow{g}\,|\,\overleftarrow{t}\,](\overleftarrow{t})^*)^*$ ce qui nous permet d'identifier le point d'interrogation que nous avions dans la section précédente avec la disjonction $[\vec{t}\,|\,\vec{g}\,|\,\overleftarrow{g}\,|\,\overleftarrow{t}\,]$. Ainsi, nous pouvons isoler trois types de collaboration possibles, c'est-à-dire de jonction entre un jeton-A et un jeton-C: i) $\vec{t}^{\,*}\overleftarrow{t}^{\,+}$, ii) $\vec{t}^{\,*}\vec{g}\,\overleftarrow{t}^{\,*}$ et iii) $\vec{t}^{\,*}\overleftarrow{g}\,\overleftarrow{t}^{\,*}$. Nous allons considérer ces trois possibilités successivement.

A.4.2 La jonction $\vec{t}^{\,*}\overleftarrow{t}^{\,+}$

La figure A.4 donne une exemple d'une jonction $\vec{t}^{\,*}\overleftarrow{t}^{\,+}$. Dans ce cas, il est clair que P ne peut atteindre Q sans la collaboration à la fois de S et de R.

En utilisant les constructions décrites aux figures A.2 et A.3 nous constatons que (i) si R est un collaborateur, un jeton-C peut atteindre S et (ii) si P est l'attaquant, un jeton-A peut arriver en S. Cependant, la conspiration ne peut réussir que si S, également, applique une règle ("*create*" en l'occurrence). Pour distinguer le fait que S est effectivement un collaborateur de ce qu'il a pu

1) S crée un arc ayant les droits tg vers un nouveau sujet N
2) P prend le droit (t sur N) à S
3) R prend le droit (g sur N) à S
4) R accorde le droit (α sur Q) à N
5) P prend le droit (α sur Q) à N

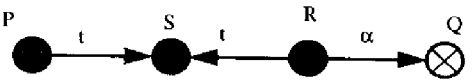


Figure A.4 Exemple de jonction $\vec{t}^{*}\overleftarrow{t}^{+}$

être atteint par un conspirateur (auquel cas il y a un jeton dans s_c même si s n'est pas un collaborateur) nous avons besoin de définir une nouvelle place. Nous l'appellerons s_i où le "i" rappelle que cette place marque l'exécution volontaire de règles : s est l'instigateur de la règle si la place s_i contient un jeton. Muni de cette nouvelle place, nous proposons, dans la figure A.5, une règle de réécriture de la jonction $\vec{t}^{*}\overleftarrow{t}^{+}$. Au vu de cette figure, il apparaît clairement que le jeton-A ne peut passer de Y vers X que si:

(1) il y a, au moins, un jeton-A dans la place Y ($Y \geq 1$)[1]
(2) Y peut initier une règle ($Y_i = 1$)
(3) X est, ou peut être atteint par, un conspirateur ($X_c = 1$)
(4) Aucune conspiration n'a déjà eu lieu pour atteindre X ($\overline{X}_s = 1$)

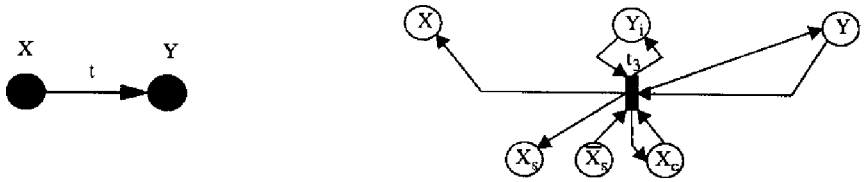


Figure A.5 Motif du Réseau de Petri pour l'arc *take* dans la jonction $\vec{t}^{*}\overleftarrow{t}^{+}$

Si ces quatre conditions sont vérifiées, alors (i) le jeton-A est propagé de Y vers X (X=1) et (ii) la collaboration réussit (X_s=1; "s" pour succès)

A.4.3 La jonction $\vec{t}^{*}\vec{g}\overleftarrow{t}^{*}$

La figure A.6. donne une exemple d'une jonction $\vec{t}^{*}\vec{g}\overleftarrow{t}^{*}$. Dans ce cas, il est clair que P peut atteindre Q avec la seule collaboration de R.

1) P crée un arc avec les droits tg vers un nouveau sujet N
2) P prend le droit (g sur R) à S
3) P accorde le droit (g sur N) à R
4) R accorde le droit (α sur Q) à N
5) P prend le droit (α sur Q) à N

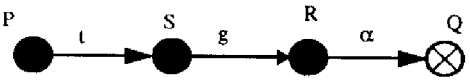


Figure A.6 Exemple de jonction $\vec{t}^{*}\vec{g}\overleftarrow{t}^{*}$

Au vu de la figure A.7, il apparaît clairement que le jeton-A ne peut passer de X vers Y que si:

(1) il y a, au moins, un jeton-A dans la place X ($X \geq 1$)
(2) Y est, ou peut être atteint par, un conspirateur ($Y_c = 1$)
(3) Aucune conspiration n'a déjà eu lieu pour atteindre Y ($\overline{Y}_s = 1$)

1. Par souci de concision, nous écrirons "Z=1" en lieu et place de m(Z)=1 pour représenter la valeur du marquage de Z durant une séquence de tir.

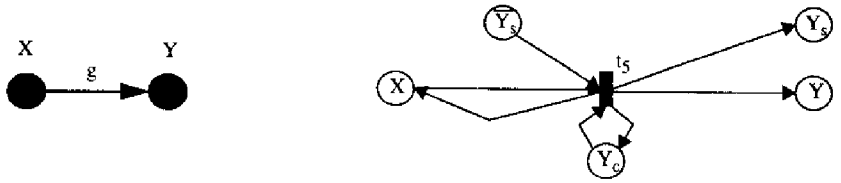


Figure A.7 Motif du réseau de Petri pour l'arc *grant* dans la jonction $\vec{t}^{*}\,\vec{g}\,\overleftarrow{t}^{*}$

Si ces trois conditions sont vérifiées, alors (i) le jeton-A est propagé à Y (Y=1) et (ii) la collaboration réussit (Y_s=1)

A.4.4 La jonction $\vec{t}^{*}\,\overleftarrow{g}\,\overleftarrow{t}^{*}$

La figure A.8 donne un exemple d'une jonction $\vec{t}^{*}\,\overleftarrow{g}\,\overleftarrow{t}^{*}$. Dans ce cas, il est clair que P peut atteindre Q avec la seule collaboration de R.

1) R accorde le droit (α sur Q) à S
5) P prend le droit (α sur Q) à S

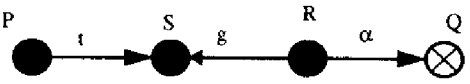


Figure A.8 Exemple de jonction $\vec{t}^{*}\,\overleftarrow{g}\,\overleftarrow{t}^{*}$

Au vu de la figure A.9, il apparaît clairement que le jeton-A ne peut passer de Y vers X que si:

(1) il y a, au moins, un jeton-A dans la place Y ($Y \geq 1$)
(2) X est, ou peut être atteint par, un conspirateur ($X_c = 1$)
(3) Aucune conspiration n'a déjà eu lieu pour atteindre X ($\overline{X}_s = 1$)

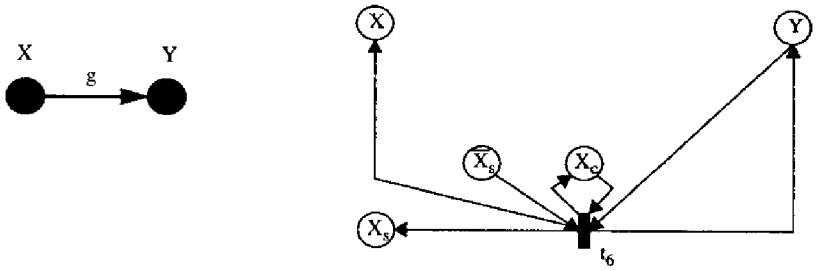


Figure A.9 Motif du réseau de Petri pour l'arc *take* de la jonction $\vec{t}^{*}\,\overleftarrow{g}\,\overleftarrow{t}^{*}$

Si ces trois conditions sont vérifiées, alors (i) le jeton-A est propagé à X (X=1) et (ii) la collaboration réussit (X_s=1)

A.4.5 Propagation "arrière" du jeton-A

Dans les sections précédentes, nous avons étudié les conditions sous lesquelles la coopération pouvait réussir, mais nous n'avons pas étudié la façon dont le jeton-A pouvait ensuite se propager le long d'un chemin préalablement parcouru par le jeton-C venu l'aider à franchir la transition bloquante. Nous illustrons ceci dans la figure A.10 et y indiquons également comment P peut acquérir le droit α sur Q avec la seule aide de R.

1) R prend le droit (g sur T) à S
2) R accorde le droit (α sur Q) à T
3) P prend le droit (α sur Q) à T

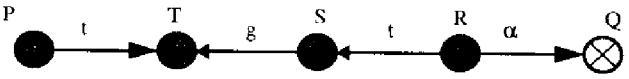


Figure A.10 Propagation "arrière" du jeton-A

Dans cet exemple, la collaboration réussit en s (s_s=1). Ensuite, le jeton-A dans s doit être propagé jusque R car le chemin a été "ouvert" par le jeton-C venu de R. Pour modéliser cela nous complétons la règle de réécriture de l'arc "take" par une nouvelle construction représentée dans la figure A.11.

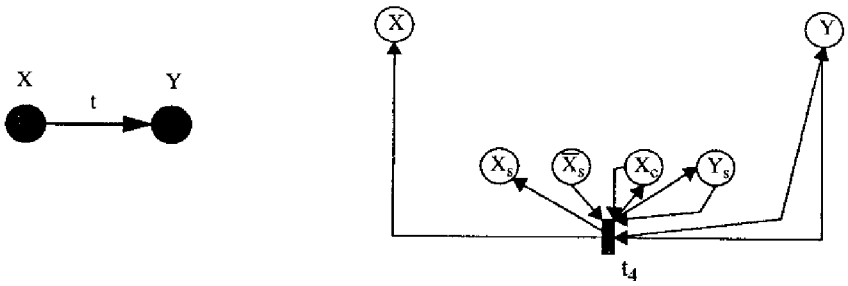


Figure A.11 Propagation "arrière" du jeton-A

A.4.6 Synthèse des différentes constructions de réécriture

Maintenant que nous avons défini des structures pour chaque cas de coopération ainsi que pour la propagation arrière du jeton-A, nous en avons fini avec les différents cas à considérer. Nous rassemblons les différentes motifs expliqués ci-dessus et obtenons deux règles de réécriture: la première, représentée dans la figure A.12, est à appliquer pour tous les arcs "*take*"; la seconde,

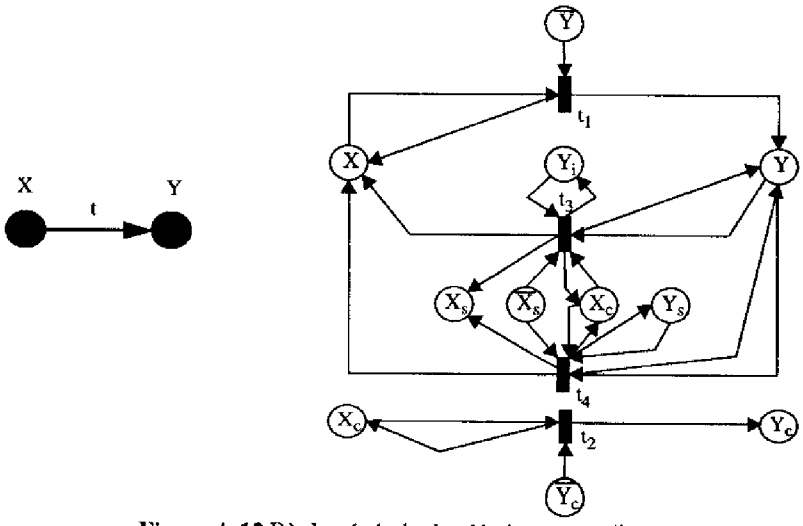


Figure A.12 Règle générale de réécriture pour l'arc *take*

représentée dans la figure A.13, est à appliquer pour tous les arcs "*grant*".

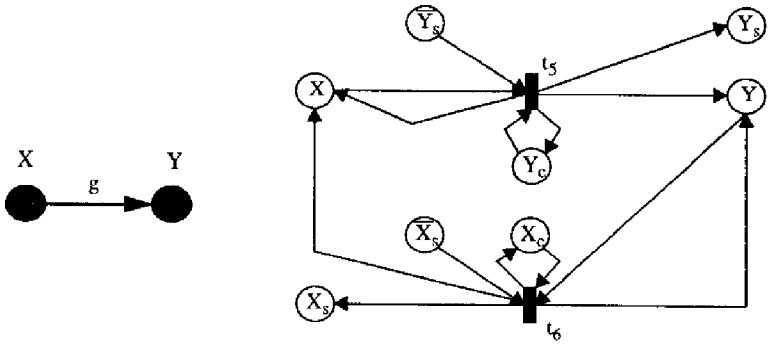


Figure A.13 Règle générale de réécriture pour l'arc *grant*

A.5. Evaluation de la complexité algorithmique

Par construction, nous savons que chaque transition de Π est tirée au plus une fois en raison du marquage initial des places de garde. Ceci nous assure l'existence d'une séquence de tir maximale de longueur finie pour Π.

Par construction, le nombre total de jetons dans les différentes places du réseau de Petri ne peut que croître, exception faite des places de garde. Dès lors, si une transition est tirable à un instant t, elle l'est toujours à l'instant t+1 sauf si (i) elle a été tirée à l'instant t ou si (ii) une autre transition, ayant les mêmes places de sortie[1], a été tirée à l'instant t. Ceci est vrai par construction[2]. Cela nous assure que toutes les séquences de tir donnent le même marquage des places de sortie. Donc, toute séquence de tir maximale produit la même valeur de vérité pour le prédicat "*peut atteindre*".

Par construction, chaque nœud peut être atteint au plus deux fois par un jeton-A (via la transition t_1 et via l'une des transitions t_3, t_4, t_5 ou t_6) et une seule fois par un jeton-C. Donc, une séquence maximale comporte au plus 3N tirs de transitions, N valant le nombre de nœuds dans le modèle TG. De plus, la transformation du modèle en un réseau de Petri nécessite O(L) applications d'une règle de réécriture, L valant le nombre d'étiquettes "*take*" et "*grant*" dans le graphe de protection initial.

La complexité de notre algorithme de résolution est donc linéaire, comme annoncé. Un autre avantage de cette méthode réside dans sa grande souplesse qui permet une définition aisée d'ensembles quelconques d'attaquants et de collaborateurs potentiels, par le biais du marquage initial du réseau de Petri construit.

1. Nous ne parlons ici que des "vraies" places de sortie. C'est-à-dire que nous ne considérons pas les places qui servent à la fois de place d'entrée et de sortie à la même transition et dont le marquage ne change pas du fait du tir de la transition.
2. $\forall t \quad m_{t+1}(\overline{Z_s}) = m_t(\overline{Z_s}) - 1 \Leftrightarrow (m_{t+1}(Z) = m_t(Z) + 1) \wedge (m_{t+1}(Z_s) = m_t(Z_s) + 1)$

   $\forall t \quad m_{t+1}(\overline{Z}) = m_t(\overline{Z}) - 1 \Leftrightarrow (m_{t+1}(Z) = m_t(Z) + 1) \wedge (m_{t+1}(Z_s) = m_t(Z_s))$

   $\forall t \quad m_{t+1}(\overline{Z_c}) = m_t(\overline{Z_c}) - 1 \Leftrightarrow (m_{t+1}(Z_c) = m_t(Z_c) + 1)$

Références bibliographiques

1. [Abrams et Toth 1994] M. D. Abrams et P. R. Toth, (Ed.), *Proc. of an Invitational Workshop on Information Technology Assurance and Trustworthiness*, NIST, Williamsburg (Virginie, USA), 75 pages, 21-23 mars 1994.
2. [Ammann et Sandhu 1992] P. E. Ammann et R. S. Sandhu, "Implementing Transaction Control Expressions by Checking for Absence of Access Rights", *Proc. of the Eigth Annual Computer Security Applications Conference*, San Antonio (Texas, USA), pp. 131-140, 30 novembre - 4 décembre 1992.
3. [Anastovski 1993] Z. Anastovski, *Risk Analysis in Computer Networks - A Review*, Rapport technique, Centre For Computer Security Research, Wollongong (NSW, Australia), 40 pages, 1993.
4. [Anderson 1991] A. M. Anderson, "Comparing Risk Analysis Methodologies", *Information Security*, D. T. Lindsay et W. L. Price (Eds.), Elsevier Science Publishers B.V. (North-Holland), pp. 301-311, 1991.
5. [Anderson et al. 1993] D. Anderson, T. F. Lunt, H. Javitz, A. Tamaru et A. Valdes, *Safeguard Final Report: Detecting Unusual Program Behaviour Using the Nides Statistical Component*, Final Report, SRI International, Project 2596, 2 décembre 1993.
6. [ARAGO-15 1994] *Informatique Tolérante aux Fautes*, Observatoire français des techniques avancées, Editions Masson, Collection ARAGO, vol. 15, 1994.
7. [Archimbaud 1992] J.-L. Archimbaud, *Conseils de sécurité sur l'administration de machines Unix sur un réseau TCP/IP*, Rapport Technique (version 4), 8 pages, disponible par ftp sur la machine ftp.urec.fr sous le nom /pub/securite/conseils.admin.4.ps, mars 1992.
8. [Archimbaud 1993] J.-L. Archimbaud, "Documents sur la sécurité des systèmes et des réseaux disponibles en ligne", *Le Micro-Bulletin du CNRS*, n° 49, pp. 90-92, CNRS/UREC, avril/mai 1993.
9. [Arlat et al. 1989] J. Arlat, S. Bachmann, C. Béounes, J.-E. Doucet, K. Kanoun, J.-C. Laprie, J. M. d. Souza, D. Powell et P. Spiesser, *SURF-2 : Spécification de Conception*, Rapport technique, LAAS 89-098, LAAS-CNRS, Toulouse (France), mars 1989.
10. [Arnaud 1994] C. Arnaud, *Démonstrateur pour l'évaluation quantitative de la sécurité d'un système UNIX*, Rapport LAAS 94322, Rapport de stage préparé au LAAS-CNRS en vue de l'obtention du diplôme de DESS de l'université de Bordeaux I, 90 pages, septembre 1994.
11. [Baird et al. 1987] B. J. Baird, L. L. Baird et R. P. Ranauro, "The Moral Cracker ?", *Computers & Security*, vol. 6, pp. 471-478, 1987.
12. [Baldwin 1987] R. W. Baldwin, "Rule Based Analysis of Computer Security", *Proc. of IEEE Compcon*, pp. 227-233, 1987.
13. [Baskerville 1991] R. Baskerville, "Risk Analysis as a Source of Professional Knowledge", *Computers & Security*, vol. 10, n° 8, pp.749-764, 1991.
14. [Baskerville 1993] R. Baskerville, "Information System Security Design Methods: Implications for Information Systems Development", *ACM Computing Surveys*, vol. 25, n° 4, pp. 375-414, 1993.
15. [Bell et La Padula 1976] D. E. Bell et L. J. LaPadula. *Secure Computer Systems: Unified Exposition and Multics Interpretation*, Rapport Technique, MTR 2997 Rev. 1, MITRE Corp., Bedford (Massachussetts, USA), 1976.
16. [Bellovin 1992] S. M. Bellovin, "There Be Dragons", *Proc. of the Third Usenix Security Symposium*, Baltimore, septembre 1992.
17. [Béounes et al. 1993] C. Béounes, M. Aguéra, J. Arlat, S. Bachmann, C. Bourdeau, J.-E. Doucet, K. Kanoun, J.-C. Laprie, S. Metge, J. Moreira de Souza, D. Powell et P. Spiesser. "SURF-2: A Program for Dependability Evaluation of Complex Hardware and Software

Systems", *Proc. of the Twenty-Third Annual International Symposium on Fault-Tolerant Computing (FTCS-23)*, pp. 668-673, Toulouse (France), IEEE Computer Society Press, 1993.

18. [Biba 1977] K. J. Biba, *Integrity Considerations for Secure Computer Systems*, Technical Report, ESD-TR-76-372 ; MTR-3153, the MITRE Corporation, 1977.
19. [Bieber et Cuppens 1992] P. Bieber et F. Cuppens, "A Logical View of Secure Dependencies", *Journal of Computer Security*, vol. 1, n° 1, IOS Press, pp. 99-129, 1992.
20. [Bishop et Snyder 1979] M. Bishop et L. Snyder "The Transfer of Information and Authority in a Protection System", *Proc. of the Seventh ACM Symposium on Operating Systems Principles*, ACM SIGOPS, Pacific Grove (Californie, USA), pp. 45-54, 10- 12 décembre 1979.
21. [Bishop 1981] M. Bishop, "Hierarchical Take-Grant Protection Systems", *Proc. of the Eighth ACM Symposium on Operating Systems Principles*, Pacific Grove (Californie, USA), 14-16 décembre 1981, in *ACM SIGOPS*, vol. 15, n° 5, pp.109-122, 1981.
22. [Bishop 1982] M. Bishop, *Security Problems with the* UNIX *Operating System*, Rapport Technique, Computer Science Dept., Purdue University, West Lafayette (Indiana, USA), 31 mars 1982.
23. [Biskup 1984] J. Biskup, "Some Variants of the Take-Grant Protection Model", *Information Processing Letters*, vol. 19, pp.151-156, octobre 1984.
24. [Bodeau et al. 1990] D. J. Bodeau, F. N. Chase et S. G. Kass, "ANSSR: A Tool for Risk Analysis of Networked Systems", *Proc. of the 13th NIST/NCSC National Computer Security Conference*, Washington (District Colombia, USA), NIST, pp. 687-696, 1-4 octobre 1990.
25. [Bradenhorst et Eloff 1990] K. P. Bradenhorst et J. H. P. Eloff, "Computer Security Methodology: Risk Analysis and Project Definition", *Computers & Security*, vol. 9, n° 4, pp. 339-346, juin 1990.
26. [Brand 1990] R. Brand, *Coping with the Threat of Computer Security Incidents: a Primer from Prevention through Recovery*, Rapport CERT, version 0.8, 44 pages, 8 juin 1990 (disponible par ftp sur cert.sei.cmu.edu ; fichier /pub/info/primer).
27. [Brewer et Nash 1989] F. C. Brewer et D. M. J. Nash, "The Chinese Wall Security Policy", *Proc. of the 1989 IEEE Symposium on Security and Privacy*, Oakland (Californie, USA), pp. 206-214, 1-3 mai 1989.
28. [Brewer et al. 1991] D. Brewer, M. Nash, B. Chorley, R. Lampard et F. Williams, "Security Criteria Harmonization: The Information Technology Security Evaluation Criteria", *Preprints of the Proc. of IFIP/SEC'91*, Brighton (UK), pp. 18-29, 15-17 mai 1991.
29. [Brunnstein 1991] K. Brunnstein. "Risk Analysis of Trusted Computer Systems", *Computer Security and Information Integrity*, K. Dittrich, S. Rautakivi et J. Saari (Eds.), Elsevier Science Publishers B.V. (North-Holland), pp. 71-88, 1991.
30. [Caelli et al. 1991] W. Caelli, D. Longley et M. Shain, *Information Security Handbook*, MacMillan Publishers, 1991.
31. [CERT 1992] *CERT/CC Generic Security Information*, Rapport Technique, Computer Emergency Response Team / Coordination Center, Pittsburgh (Pensylvanie, USA), 2 juillet 1992.
32. [Cheswick 1991] B. Cheswick, *An Evening with Berferd in Which a Cracker is Lured, Endured, an Studied*, Rapport Technique, AT&T Bell Laboratories, 11 pages, 1991.
33. [Clark et Wilson 1987] D. D. Clark et D. R. Wilson, "A Comparison of Commercial and Military Computer Security Policies", *Proc. of the 1987 IEEE Symposium on Security and Privacy*, Oakland (Californie, USA), pp. 184-194, mai 1987.
34. [CTCPEC 1993] *The Canadian Trusted Computer Product Evaluation Criteria*, Version 3.0e, Canadian System Security Centre, Communications Security Establishment Government of Canada, janvier 1993.

35. [Curry 1990] D. A. Curry, *Improving the Security of Your Unix System,* Rapport Technique, ITSTD-721-FR-90-21, SRI International, avril 1990.
36. [Dacier et Rutsaert 1991a] M. Dacier et M. Rutsaert, "Comment gérer la transitivité en sécurité.", *Proc. de la Convention Unix 91*, AFUU, mars 1991.
37. [Dacier et Rutsaert 1991b] M. Dacier et M. Rutsaert, "Gérer la transitivité en sécurité", *Bancatique - Dossier Sécurité*, pp 575-579, novembre 1991.
38. [Dacier 1992] M. Dacier, "C.A.S. : Conseiller Automatique en Sécurité, Prototype d'évaluation de la sécurité sous Unix", *Tribunix - Dossier Sécurité*, vol. 8, n° 42, pp.20-23, mars/avril 1992.
39. [Dacier 1993] M. Dacier, "A Petri Net Representation of the Take-Grant Model", *Proc. of the Computer Security Foundations Workshop VI,* Franconia (New Hampshire, USA), IEEE Computer Society Press, pp. 99-108, 15-17 juin 1993.
40. [Dacier et al. 1993] M. Dacier, M. Kaâniche et Y. Deswarte, "A Framework For Security Assessment of Insecure Systems", *Predictably Dependable Computing Systems; First Year Report,* pp. 561-578, Toulouse (France), ESPRIT BRA Project 6362 - PDCS 2, septembre 1993.
41. [Dacier 1994] M. Dacier, "A Fault Forecasting Approach for Operational Security Monitoring", *Preprints of the Fourth Int. Working Conference on Dependable Computing for Critical Applications (DCCA-4),* IFIP WG 10.4, Position Paper for the Panel "Qualitative vs. Quantitative Assessment of Security", San Diego (Californie, USA), pp. 142-143, 4-5 janvier 1994.
42. [Dacier et Deswarte 1994] M. Dacier et Y. Deswarte, "Privilege Graph: an Extension to the Typed Access Matrix Model", *Lecture Notes in Computer Science,* Springer Verlag, vol. 875, pp. 319-334, novembre 1994 (Proc. of Esorics'94, novembre 1994, Brighton, UK).
43. [Denning 1971] P. J. Denning, "Third Generation Computer Systems", *Computing Surveys,* vol. 3/4, pp.171-216, décembre, 1971.
44. [Denning 1990] D. E. Denning, "Concerning Hackers Who Break into Computer Systems", *Proc. of the 13th NIST/NCSC National Computer Security Conference,* Washington (District Colombia., USA), NIST, vol. 2, pp. 653-664, 1-4 octobre 1990.
45. [Deswarte 1994] Y. Deswarte, "Improving Security by Fault-Tolerance", *Preprints of the Fourth Int. Working Conference on Dependable Computing for Critical Applications (DCCA-4),* IFIP WG 10.4, Position Paper for the Panel "Common Techniques in Fault-Tolerance and Security", San Diego (Californie, USA), pp. 254-255, 4-5 janvier 1994.
46. [Duff 1989] T. Duff, "Experience with Viruses on Unix Systems", *Computing Systems,* vol. 2, n° 2, pp. 155-171, Printemps 1989.
47. [Farmer et Spafford 1990] D. Farmer et E. H. Spafford, "The Cops Security Checker System", *Proc. of the Summer Usenix Conference,* Anaheim (Californie, USA), 1990.
48. [Farmer et Venema 1993] D. Farmer et W. Venema, *Improving the Security of your Site by Breaking Into it,* forum comp.security.unix, message 1388, 2 décembre 1993.
49. [Farquhar 1991] B. Farquhar, "One Approach to Risk assessment", *Computers & Security,* vol. 10, pp. 21-23, 1991.
50. [Federal Criteria 1992a] *Federal Criteria for Information Technology Security,* Draft, Volume I, National Institute of Standards and Technology (NIST) and National Security Agency (NSA), 1992.
51. [Federal Criteria 1992b] *Federal Criteria for Information Technology Security,* Draft, Volume II, National Institute of Standards and Technology (NIST) and National Security Agency (NSA), 1992.
52. [Fitch III et Hoffman 1993] J. A. Fitch III et L. J. Hoffman, "A Shortest Path Network Security Model", *Computer & Security,* vol. 12, pp. 169-189, 1993.
53. [Frenkel 1987] K. A. Frenkel, "Brian K. Reid: A graphics Tale of a Hacker Tracker", *Communications of the ACM,* vol. 30, n° 10, pp. 820-823, octobre 1987.

54. [Ganta et Sandhu 1993] S. Ganta et R. S. Sandhu, "On Testing for Absence of Rights in Access Control Models", *Proc. of the Computer Security Foundations Workshop VI*, Franconia (New Hampshire, USA), IEEE Computer Society, pp. 109-118, 15-17 juin 1993.
55. [Gardner 1989] P. E. Gardner, "Evaluation of Five Risk Assessment Programs", *Computers and Security* , vol. 8, n° 6, pp. 479-485, 1989.
56. [GBT 1993] *Les méthodes d'analyse de risques*, GBT Conseils, 46 pages, 9 juin 1993
57. [Graham et Denning 1972] G. S. Graham et P. J. Denning, "Protection - Principles and Practices", *1972 AFIPS Spring Joint Computer Conf.*, Arlington (Virginie, USA), AFIPS Press, pp. 417-429, 1972.
58. [Grampp et Morris 1984] F. T. Grampp et R. H. Morris, "UNIX Operating System Security", *AT&T Bell Laboratories Technical Journal*, vol. 63, n° 8, pp. 1649-1672, octobre 1984.
59. [Green Book 1993] *Green Book on the Security of Information Systems*, Draft 4.0, Commission des Communautés Européennes, DG XIII, Directorat B, 18 octobre 1993.
60. [Grissonanche 1987] A. Grissonanche, "La sécurité des systèmes d'information", *Micro-Systemes Entreprises*, pp. 39-43, mars 1987.
61. [Guarro 1987] S. B. Guarro, "Principles and Procedures of the LRAM Approach to Information Systems Risk Analysis and Management", *Computers & Security*, vol. 6, n° 6 , pp. 493-504, 1987.
62. [Gupta et Gligor 1992] S. Gupta et V. D. Gligor, "Experience with a Penetration Analysis Method and Tool", *Proc. of the 15th NIST/NCSC National Computer Security Conference*, Baltimore (Maryland, USA), NIST, pp. 165-183, 1992.
63. [Haga et Zviran 1991] W. J. Haga et M. Zviran, "The Practice of Passwords Usage: Some Empirical Evidence", *Preprints of the Proc. of IFIP/SEC'91*, Brighton (UK), pp. 164-172, 15-17 mai 1991.
64. [Harrison et Ruzzo 1975] M. A. Harrison et W. L. Ruzzo, "On Protection in Operating Systems", *Proc. of the fifth ACM Symposium on Operating Systems Principles*, 19-20 novembre 1975, Austin (Texas, USA), in *ACM SIGOPS*, vol. 9, n° 5, pp.14-24, 1975.
65. [Harrison et Ruzzo 1976] M. A. Harrison et W. L. Ruzzo, "Protection in Operating Systems", *Communications of the ACM*, vol. 19, n° 8, pp. 461-470, 1976.
66. [Hellewell et al. 1992] J. Hellewell, M. Anderson et B. Billard, *Technical Rationale for Australian Computer Security Risk Analysis Guidelines*, Rapport Technique, ERL-0621-RR, DSTO Electronic Research Laboratory, Salisbury (South Australia), 1992.
67. [Herschberg 1988] I. S. Herschberg, "Make the Tigers Hunt for You", *Computers and Security*, vol. 7, n° 2, pp.197-203, 1988.
68. [Heydon et al. 1989a] A. Heydon, M. Maimone, J. D. Tygar, J. Wing et A. M. Zaremski, "Constraining Pictures with Pictures", *Information Processing 89*, IFIP, pp. 157-162, 1989.
69. [Heydon et al. 1989b] A. Heydon, M. W. Maimone, J. D. Tygar, J. M. Wing et A. M. Zaremski, "Miró Tools", *Proc. of the 1989 IEEE Workshop on Visual Languages*, pp. 86-91, octobre 1989.
70. [Heydon et al. 1990] A. Heydon, M. W. Maimone, J. D. Tygar, J. M. Wing et A. M. Zaremski, "Miró : Visual Specification of Security", *IEEE Transactions on Software Engineering*, vol. 16, n° 10, pp. 1185-1197, octobre 1990.
71. [Heydon 1992] C. A. Heydon, *Processing Visual Specifications of File System Security* Ph.D. Thesis, CMU-CS-91-201, School of Computer Science, Pittsburgh, PA, 215 pages, 31 janvier 1992.
72. [Heydon et Tygar 1994] A. Heydon and J. D. Tygar, "Specifying and Checking Unix Security Constraints", *Computing Systems*, vol. 7, n° 1, pp.91-112, 1994.

73. [Hoffman 1986] L. J. Hoffman, *Risk Analysis and Computer Security : Problems and Issues*, Rapport Technique, GWU-IIST-86-06, Dpt. of Electrical Engineering and Computer Science, mars 1986.

74. [Hoffman 1989] L. J. Hoffman, "Smoking Out the Bad Actors : Risk Analysis in the Age of The Microcomputer", *Computers & Security*, vol. 8, n° 4, pp. 299-302, 1989.

75. [HP 1988] *HP-UX System Security, HP-9000 Series 300/800 Computers*, Hewlett-Packard Company, Cupertino (Californie, USA), 1988.

76. [Irvine et al. 1989] C. E. Irvine, T. Levin et S. J. Padilla, "A Formal Model for Unix Setuid", *Proc. of the IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, Oakland (Californie, USA), pp. 73-83, 1-3 mai 1989.

77. [ITSEC 1991] *Critères d'évaluation de la sécurité des systèmes informatiques (ITSEC)*, Version 1.2, Commission des Communautés Européennes, DG XIII, juin 1991.

78. [ITSEM 1993] *Information Technology Security Evaluation Manual (ITSEM)*, Version 1.0, Commission des Communautés Européennes, DG XIII, septembre 1993.

79. [Javitz et Valdez 1991] H. S. Javitz et A. Valdes, "The SRI-IDES Statistical Anomaly Detector", *Proc. of the IEEE Symposium on Security and Privacy*, pp. 316-326, Oakland (Californie, USA), mai, 1991.

80. [Javitz et Valdez 1994] H. S. Javitz et A. Valdez, *The NIDES Statistical Component: Descriptions and Justification*, Annual Report, A010, SRI International, 7 mars 1994.

81. [JCSEC 1992] *The Japanese Computer Security Evaluation Criteria - Functionality Requirements*, Draft V1.0, Ministry of International Trade and Industry, août 1992.

82. [Jones et al. 1976]A. K. Jones, R. J. Lipton et L. Snyder, "A Linear Time Algorithm for deciding Security", *Proc. of the 17th Annual Symposium on Fundations of Computer Science*, Houston (Texas, USA), pp. 33-41, 1976.

83. [Kemeny et Snell 1959] J. G. Kemeny et J. L. Snell, *Finite Markov Chains*, Princeton, van Nostrand, 1959.

84. [Klein et al. 1992] Y. Klein, E. Roche, F. Taal, M. Van Dulm, U. Van Essen, P. Wolf et J. Yates, "The IT Security Evaluation Manual (ITSEM)", *Proc. of the 15th NIST/NCSC National Computer Security Conference*, Baltimore (Maryland, USA), NIST, pp 10, 1992.

85. [Kristiansen 1991] K. E. Kristiansen, "Information Systems Auditing - A Support to Management", *Computer Security and Information Integrity*, K. Dittrich, S. Rautakivi et J. Saari (Eds.), Elsevier Science Publishers (North-Holland), pp. 245-254, 1991.

86. [Lamère 1986] J.-M. Lamère, "Method of Analysis and Reduction of Risks Carried by the Network (Marion-RSX)", *Preprints of the Proc. of IFIP/SEC'86*, , Monte-Carlo (France), pp 100-104, 2-4 décembre 1986.

87. [Lampson 1971] B. W. Lampson, "Protection", *ACM SIGOPS*, vol. 8, n° 1, pp. 18-24, janvier 1974 (publié initialement dans *Proc. 5th Princeton Symp. Information Sciences and Systems*, mars 1971).

88. [Lampson 1973] B. W. Lampson, "A Note on the Confinment Problem", *Communications of the ACM*, vol. 16, n° 10, pp. 613-615, octobre 1973.

89. [Landwehr 1981] C. E. Landwehr, "Formal Models for Computer Security", *ACM Computing Surveys*, vol. 13, n° 3, pp. 247-278, septembre, 1981.

90. [Landwehr 1983] C. E. Landwehr, "The Best Available Technologies for Computer Security", *IEEE Computer*, vol. 16, n° 7, pp. 86-100, juillet 1983.

91. [Landwehr et al. 1993] C. E. Landwehr, A. R. Bull, J. P. McDermott et W. S. Choi. *A Taxonomy of Computer Program Security Flaws, with Examples*, Rapport Technique, NRL/FR/5542-93-9591, Naval Research Laboratory, 1993.

92. [Laprie 1992] J. C. Laprie (Ed.), *Dependability: Basic Concepts and Terminology in English, French, German, Italian and Japanese*, Dependable Computing and Fault-Tolerant Systems, vol. 5, Vienna, Austria, Springer-Verlag, 1992.

93. [Laprie et Deswarte 1993] J. C. Laprie et Y. Deswarte, "Fautes accidentelles et fautes intentionnelles : une perspective", *Proc. du Premier Congrès Biennal de l'AFCET*, tome 6 (Sécurité et Sûreté Informatiques), pp. 1-10, 8-10 juin 1993.

94. [Le Roux 1991] Y. Le Roux, "Technical Criteria for Security Evaluation of Information Technology Products", *Computer Security and Information Integrity*, K. Dittrich, S. Rautakivi et J. Saari (Eds.), Elsevier Science Publishers (North-Holland), pp. 265-273, 1991.

95. [Lipton et Snyder 1977] R. J. Lipton et L. Snyder, "A Linear Time Algorithm for Deciding Subject Security", *Journal of the ACM*, vol. 24, n° 3, pp. 455-464, juillet 1977.

96. [Littlewood et al. 1991] B. Littlewood, S. Brocklehurst, N. E. Fenton, P. Mellor, S. Page, D. Wright, J. E. Dobson, J. A. McDermid et D. Gollmann, "Towards Operational Measures for Computer Security", *PDCS Project Second Year Report (Esprit BRA Project 3092)*, vol. 3, chap. 2, 1991.

97. [Littlewood et al. 1993] B. Littlewood, S. Brocklehurst, N. E. Fenton, P. Mellor, S. Page, D. Wright, J. E. Dobson, J. A. McDermid et D. Gollmann, "Towards Operational Measures for Computer Security", *Journal of Computer Security*, vol. 2, n° 2/3, pp. 211-229, 1993.

98. [Lunt et al. 1990] T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. G. Neumann et C. Jalali, "IDES: A Progress Report", *Proc. Of the Sixth Annual Comp. Security Applications*, Tucson (Arizona, USA), 13 pages, décembre 1990.

99. [Maimone et al. 1990] M. W. Maimone, J. D. Tygar et J. M. Wing, "Miró Semantics for Security", *Proc. of the 1988 IEEE Workshop on Visual Languages*, pp. 45-51, 10-12 octobre 1988.

100. [McDermott et Jajodia 1993] J. McDermott, S. Jajodia, "Orange Locking: Channel-Free Database Concurrency Control via Locking", *Database Security, VI: Status and Prospects*, B. M. Thuraisingham et C. E. Landwehr (Eds.), Elsevier Science Publishers B.V. (North-Holland), pp. 267-284, 1993.

101. [McIlroy 1989] M. D. McIlroy, "Virology 101", *Computing Systems*, vol. 2, n° 2, pp.173-181, printemps 1989.

102. [Metge 1994] S. Metge, "Modeling and Evaluation of Hardware and Software Systems Dependability Using Surf-2", *Proc. of the Conference on Modelling Techniques and Tools for Computer Performance Evaluation (MTT'94)*, Vienne (Autriche), Universität Wien, pp. 19-22, 1994.

103. [Millet et Vu 1992] J.-R. Millet et F. Vu, *Vous avez dit: Sécurité sous Unix ?*, Rapport Technique, Institut National Polytechnique de Grenoble, 29 juin 1992.

104. [Molloy 1982] M. Molloy, "Performance analysis using Stochastic Petri nets", *IEEE Trans. on Computers*, vol. 39, n° 9, pp. 913-917, 1982.

105. [Morris 1985] R. T. Morris, *A Weakness in the 4.2 BSD Unix TCP/IP Software*, Rapport Technique, AT&T Bell Laboratories, 25 février 1985.

106. [Myers et al. 1990] B. A. Myers, D. A. Giuse, R. B. Dannenberg, B. V. Zanden, D. S. Kosbie, E. Pervin, A. Mickish et P. Marchal, "Garnet : Comprehensive Support for Graphical, Highly Interactive User Interfaces", *IEEE Computer*, vol. 23, n° 11, pp.71-85, novembre 1990.

107. [Natkin 1980] S. Natkin, *Les réseaux de Petri stochastiques*, Thèse de Docteur Ingénieur, CNAM, Paris, 1980.

108. [Neugent 1987] B. Neugent, "Password-Based Authentication (Versus User Volatile Memory)", *ACM SIGSAC*, vol. 5, n° 4, pp. 10-13, automne 1987.

109. [Neumann 1994] P. G. Neumann, "Reliability and Security", *Preprints of the Fourth Int. Working Conference on Dependable Computing for Critical Applications (DCCA-4)*, IFIP WG 10.4, Position Paper for the Panel "Common Techniques in Fault-Tolerance and Security", San Diego (Californie, USA), p. 256, 4-5 janvier 1994.

110. [Olovsson et al. 1993] T. Olovsson, E. Jonsson, S. Brocklehurst et B. Littlewood, "Data Collection for Security Fault Forecasting: Pilot Experiment", *Predictably Dependable Computing Systems; First Year Report*, pp. 515-560, Toulouse (France), ESPRIT BRA Project 6362 - PDCS 2, septembre 1993.

111. [Parker 1986] D. B. Parker, "Consequential Loss from Computer Crime", *Preprints of the Proc. of IFIP/SEC'86*, , Monte-Carlo (France), pp. 425-429, 2-4 décembre 1986.

112. [Peterson 1981] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, 1981.

113. [Randell 1994] B. Randell, "Fault-Tolerance and Security", *Preprints of the Fourth Int. Working Conference on Dependable Computing for Critical Applications (DCCA-4)*, IFIP WG 10.4, Position Paper for the Panel "Common Techniques in Fault-Tolerance and Security", San Diego (Californie, USA), p. 257, 4-5 janvier 1994.

114. [Reid 1986] B. Reid, "Lessons from the UNIX Breakins at Stanford", *ACM SIGSOFT Software Engineering Notes*, vol. 11, n° 5, pp. 29-35, octobre 1986.

115. [Riddle et al. 1989] B. L. Riddle, M. S. Miron et J. A. Semo, "Passwords in Use in a University Timesharing Environment", *Computers & Security*, vol. 8, pp. 569-579, 1989.

116. [Ritchie 1986] D. M. Ritchie, "On the Security of Unix", Rapport Technique, mai 1975. Repris dans *Unix System Manager's Manual*, 4.3 Berkeley Software Distribution, University of California, Berkeley, avril 1986.

117. [Rochlis et Eichin 1989] J. A. Rochlis et M. W. Eichin, "With Microscope and Tweezers: The Worm from MIT's Perspective", *Communications of the ACM*, vol. 32, n° 6, pp. 689-698, juin 1989.

118. [Saari 1991] J. Saari, "Top Management Challenge - From Quantitative Guesses to Prudent Baseline of Security", *Information Security*, D. T. Lindsay et W. L. Price (Eds.), Elsevier Science Publishers B.V. (North-Holland), pp. 287-292, 1991.

119. [Sandhu 1988] R. S. Sandhu, "The Schematic Protection Model: Its Definition and Analysis of Acyclic Attenuation Schemes", *Journal of the ACM*, n° 2, pp. 404-432, 1988.

120. [Sandhu 1992a] R. S. Sandhu, "Expressive Power of the Schematic Protection Model", *Journal of Computer Security*, vol. 1, n° 1, pp. 59-98, 1992.

121. [Sandhu 1992b] R. S. Sandhu, "The Typed Access Matrix Model", *Proc. of the 1992 IEEE Symposium on Research in Security and Privacy*, Oakland (Californie, USA), pp. 122-136, 4-6 mai 1992.

122. [Sandhu et al. 1992] R. S. Sandhu, R. Thomas et S. Jajodia, "Supporting timing-channel free computations in multilevel secure object-oriented databases", *Database security, V : Status and Prospects*, C.E. Landwehr et S. Jajodia (Eds.), Elsevier Science Publishers B.V. (North-Holland), pp. 297-314, 1992.

123. [Sandhu 1993] R. S. Sandhu, "Lattice-based Access Control Models", *Computer*, pp. 9-19, novembre 1993.

124. [Seeley 1989] D. Seeley, "A Tour of the Worm", *Proc. of the Winter 1989 USENIX Conf.*, San Diego (Californie, USA), pp. 287-304, février 1989.

125. [Shafer 1976] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, 1976.

126. [Snyder 1977] L. Snyder, "On the Synthesis and Analysis of Protection Systems.", *Proc. of the Sixth ACM Symposium on Operating Systems Principles*, 16-18 novembre 1977, Purdue University (USA), in *ACM SIGOPS*, vol. 11, n° 5, pp.141-150, 1977.

127. [Snyder 1981a] L. Snyder, "Formal Models of Capability-Based Protection Systems", *IEEE Transactions on Computers*, vol. C-30, n° 3, pp. 172-181, mars 1981.

128. [Snyder 1981b] L. Snyder, "Theft and Conspiracy in the Take-Grant Model", *Journal of Computer and System Sciences*, vol. 23, pp. 333-347, décembre 1981.

129. [Spafford 1986] E. H. Spafford, "The Internet Worm: Crisis and Aftermath", *Communications of the ACM*, vol. 32, n° 6, pp. 678-687, juin 1986.

130. [Spafford 1988] E. H. Spafford, *The Internet Worm Program : An Analysis*, Rapport Technique, CSD-TR-823, Purdue University, 29 novembre 1988 (revu en décembre 1988).

131. [Spafford 1989] E. H. Spafford, "Some Musings on Ethics and Computer Break-Ins", *Proc. of the Winter 1989 USENIX Conf.*, San Diego (Californie, USA), pp. 305-311, février 1989.

132. [Spafford et Garfinkel 1991] E. Spafford et S. Garfinkel, *Practical Unix Security*, O'Reilly & Associates (Inc.), 483 pages, septembre 1991.

133. [Spafford 1992] E. H. Spafford, "OPUS: Preventing Weak Password Choices", *Computers & Security*, vol. 11, n° 3, pp. 273-278, 1992.

134. [Stanley 1986] P. M. Stanley, "Computer Crime Investigation - The Lessons Learned from Experience", *Preprints of the Proc. of IFIP/SEC'86*, , Monte-Carlo (France), pp. 319-326, 2-4 décembre 1986.

135. [Stanley 1991] P. M. Stanley, "Computer Crime Investigation Training - Concept, Content and Cases", *Preprints of the Proc. of IFIP/SEC'91*, Brighton (UK), pp. 420-429, 15-17 mai 1991.

136. [Stoll 1988] C. Stoll, "How Secure are Computers in the USA", *Computers & Security*, vol. 7, pp. 543-547, 1988.

137. [Strack 1990] H. Strack, "Extended Access Control in UNIX System V - ACLs and Context", *Proc. of the UNIX Security Workshop*, USENIX, mai 1990.

138. [Tanenbaum 1987] A. S. Tanenbaum, *Operating Systems Design and Implementation*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

139. [TCSEC 1983] *Department of Defense Trusted Computer System Evaluation Criteria*, Department of Defense, Rapport Technique, CSC-STD-001-83, 1983.

140. [TCSEC 1985] *Department of Defense Trusted Computer System Evaluation Criteria*, DoD Standard, Department of Defense, Rapport Technique, DoD 5200.28-STD, 1985.

141. [Thompson 1984] K. Thompson, "Reflections on Trusting Trust", *Communications of the ACM*, 27 (8), pp. 761-763, 1984.

142. [von Solms et de Villiers 1988] S. H. von Solms et D. P. de Villiers, "Protection Graph Rewriting Grammars and the Take/Grant Security Model", *Quæstiones Informaticæ*, vol. 6, n° 1, pp. 15-18, mai 1988.

143. [von Solms et Edwards 1989] S. H. von Solms et N. G. Edwards, "Designing and Implementation of a New Security Model", *International Journal of Computer Mathematics*, vol. 29, pp. 139-149, 1989.

144. [Ware 1970] W. H. Ware (Ed.), *Security Controls for Computer Systems: Report of Defense Science Board Task Force on Computer Security*, Rapport Technique, AD-A076617/0, Santa Monica (Californie, USA), Rand Corporation, 1970 (republié octobre 1979).

145. [Warren 1983] A. D. Warren, "Evaluating Risks of Computer Fraud and Error", *Computers & Security*, vol. 2, n° 2, pp. 132-143, juin 1983.

146. [Wood et Kochan 1987] P. H. Wood et S. G. Kochan, *Unix System Security*, Collection Hayden Unix System Library, Hayden Book Company, 1987.

147. [Wu 1981] M. S. Wu, *Hierarchical Protection Systems*, Oakland (Californie, USA), Department of Management Sciences, University of Iowa, pp. 113-123, 1981.

148. [Zadeh 1978] L. A. Zadeh, "Fuzzy Sets as a Basis for a Theory of Possibility", *Fuzzy sets and Systems*, Vol.1, n° 1, pp. 3-28, 1978.

149. [Zviran et al. 1990] M. Zviran, J. C. Hoge et V. A. Micucci, "SPAN - a DSS for Security Plan Analysis", *Computers & Security*, vol. 9, n° 2, pp. 153-160, avril 1990.

Index des Références Bibliographiques

Liste des figures

Liste des tables

Table des matières

Thèse de doctorat de Monsieur Marc Dacier

"Vers une évaluation quantitative de la sécurité informatique"

RÉSUMÉ

Les systèmes d'information actuels doivent, à la fois, protéger les informations qui leur sont confiées et se plier à des environnements opérationnels variables. Ces deux objectifs, sécurité et flexibilité, peuvent être antinomiques. Ce conflit conduit généralement à l'utilisation de systèmes offrant un niveau de sécurité acceptable, mais non maximal. Définir un tel niveau présuppose l'existence de méthodes d'évaluation de la sécurité. Cette problématique fait l'objet de cette thèse. L'auteur y passe en revue les différents critères d'évaluation existant ainsi que les méthodes dites d'analyse des risques. Ceci introduit la nécessité de définir un cadre formel capable de modéliser tout système et d'évaluer dans quelle mesure il satisfait à des objectifs de protection précis.

Les modèles formels, développés pour l'étude de la sécurité informatique, n'offrent pas le cadre mathématique désiré. L'auteur montre qu'ils adoptent une hypothèse de pire cas sur le comportement des utilisateurs, incompatible avec une modélisation réaliste. Après avoir montré, sur la base du modèle *take-grant*, comment s'affranchir de cette hypothèse, l'auteur définit un nouveau modèle, le graphe des privilèges, plus efficace pour gérer certains problèmes de protection. Il illustre son utilisation dans le cadre des systèmes Unix.

Enfin, l'auteur propose d'évaluer la sécurité en calculant le temps et l'effort nécessaires à un intrus pour violer les objectifs de protection. Il montre comment définir un cadre mathématique apte à représenter le système pour obtenir de telles mesures. Pour cela, le graphe des privilèges est transformé en un réseau de Petri stochastique et son graphe des marquages est dérivé. Les mesures sont calculées sur cette dernière structure et leurs propriétés mathématiques sont démontrées. L'auteur illustre l'utilité du modèle par quelques résultats issus d'un prototype développé afin d'étudier la sécurité opérationnelle d'un système Unix.

Mots-clés : sécurité informatique, analyse de risques, critères d'évaluation, modèles formels, graphes de Markov, réseaux de Petri.

"Towards a quantitative evaluation of computer security"

ABSTRACT

Current computing systems have to protect the data they hold and to fit easily into versatile working environments. These objectives, security and flexibility, can be antinomic. The conflict usually leads to the use of systems that offer an acceptable security level, yet not maximal. Defining such security level requires evaluation methods. This is the topic of the thesis.

The author presents the existing evaluation criterias and the so-called risk analysis methods. This introduces the need for a formal framework that can modelize any system and evaluate its effectiveness to implement well-defined protection goals.

Computer security formal models do not fit that purpose. The author shows that they use a worst case hypothesis about the behaviour of the users, incompatible with a realistic modeling.

The author offers a solution to relax the hypothesis for the "take-grant" model. Then, he defines a new model, called the privilege graph, more efficient to deal with some protection schemes. He illustrates its use in the context of Unix systems.

Finally, the author proposes a computer security evaluation method by computing the time and effort required to an intruder for breaking into the system. He defines a mathematical framework to represent the system and to obtain such measures. To do so, the privilege graph is transformed into a stochastic Petri net and its marking graph is derived. Based on this structure, the measures are evaluated and their mathematical properties are proofed. The author gives an illustration of the usefulness of the model by analyzing some results obtained with a prototype developed to study the operational security of a Unix system.

Keywords: computer security, risk analysis, evaluation criteria, formal models, Markov graph, Petri nets.