

*A mes très chers parents,
frères, soeur,
tantes, oncles
et cousins*

*To my dear
Chaals*

*A mon cher professeur
Guy Pujolle*

Dédicaces

Je dédie humblement ce manuscrit à :

A celle qui m'a toujours ouvert ses bras et soutenue dans tout ce que j'ai entrepris ; celle qui a su être bonne, gentille et compréhensive avec moi ; celle dont je regrette l'absence à cette étape importante de ma vie ; celle qui me manque terriblement aujourd'hui **ma très chère et adorée grand mère.**

A celle qui s'est toujours dévouée et sacrifiée pour moi ; celle qui m'a aidée du mieux qu'elle pouvait pour réussir ; celle qui m'a accompagnée tout au long de ce parcours périlleux ; celle qui a toujours été là dans mes moments de détresse , **ma très chère mère.**

A celui qui m'a toujours encouragée et soutenue moralement, **mon très cher père.**

A celles qui m'ont toujours aidée, écoutée, soutenue et encouragée tout au long de mon parcours ; celles qui ont toujours été présentes pour moi, **mes très chère tantes Mimi et Zou.**

A **ma très chère sœur Farah** et à **mes très chers frères Idir et Nassim** qui m'ont énormément aidée et à qui je témoigne mon affection et ma profonde reconnaissance.

A **mes très chers oncles Kamel et Abderzak, ma cousine Nina** et **mes cousins Riad, Mehdi et Nanou.**

A celui qui a su m'aimer, me supporter (dans les deux sens du terme), **mon adoré et très cher Chaals**, en témoignage de sa gentillesse et de son affection.

May this manuscrit always remind me of his kindness, his generosity and his love for me.

A **mon très cher professeur Guy Pujolle** qui m'a toujours encouragée et soutenue depuis le début de ma thèse ; celui qui a toujours su trouver les mots pour me redonner la force de continuer et d'aller au bout de cette aventure qu'est la thèse !!

KARIMA

Remerciements

Ce travail de thèse a été le labeur de trois années et n'aurait probablement jamais été mené à terme sans le soutien d'un grand nombre de personnes que je tiens vivement et très sincèrement à remercier.

Je commencerai d'abord par remercier mon très cher professeur **Guy Pujolle** que j'ai eu la chance de rencontrer lors de mon passage au laboratoire Prism et à qui je dédie également cette thèse. Je le remercie pour son aide précieuse, pour la qualité de ses conseils, pour avoir toujours été là pour m'encourager, pour me soutenir, et pour m'avoir permis de reprendre confiance en moi lorsque j'ai eu une décision importante à prendre à un moment critique de ma thèse.

Je tiens à faire part de toute ma reconnaissance envers Melle **Zahia Guessoum** qui a accepté de m'encadrer durant les deux dernières années de ma thèse, au moment où je cherchais désespérément une aide dans le domaine des agents intelligents et des systèmes multi-agents. Je la remercie pour ses remarques et critiques pertinentes et fructueuses, pour les nombreuses discussions que nous avons eu, pour sa gentillesse, pour avoir su me faire confiance et trouver les mots qu'il fallait pour me redonner espoir à chaque fois que «la lumière du bout du tunnel s'affalibissait !».

Il y a également trois autres personnes à qui je suis profondément reconnaissante pour avoir su m'écouter et me redonner de l'énergie dans les moments les plus critiques :

- **Didier Loisel** pour qui ma gratitude est bien grande. Son aide m'a été très précieuse tout au long des trois années. Il a toujours été présent pour m'aider du mieux qu'il pouvait à avancer sur ma thèse. Je le remercie vivement pour l'intérêt qu'il n'a pas cessé de porter à mon travail, pour ses idées originales et pour avoir pris le temps de relire mes chapitres de thèse malgré la charge de travail énorme qu'il avait à assurer au sein du laboratoire de notre département Communication d'Entreprise.
- **Philippe Dubois** et **Karim Maouche** qui n'ont pas arrêté de me soutenir, qui ont su me pousser à en arriver au bout et qui ont supporté mes crises et ma mauvaise humeur chronique de thésard en phase terminale. Je remercie **Karim** pour avoir toujours été là pour m'inviter aux soirées du Pitou pour me faire sortir de mon environnement triste et ennuyeux de rédaction et m'offrir un environnement plus gaie.

Je remercie également mon directeur de thèse **Jacques Labetoulle** qui m'a poussée à sa manière à aller au bout de cette thèse, même si cela n'a pas été toujours facile et si les discussions étaient assez houleuses. Cela a été vraisemblablement fructueux et m'a permis de donner le meilleur de moi-même. Je le remercie également pour ses critiques parfois dures mais pertinentes lors de la relecture de mon manuscrit.

Je tiens à remercier **Giuliano Di-Vitantonio** et **Solange Ghernaouti-Hélie** pour avoir accepté de faire partie du jury et de juger ainsi mon travail. Je remercie également le président du jury **A. Shiper** et les autres membres du jury **R. Molva** et **S. Vaudenay**.

Je voudrai aussi remercier **David Tremouillhac** pour m'avoir aidé à faire la mise en page de ce manuscrit et sans qui la version finale de ce document aurait eu du mal à voir le jour.

Je ne peux oublier tous mes amis qui m'ont accompagnée tout au long de ce parcours périlleux et qui n'ont pas cessé de me soutenir moralement et de m'encourager :

- Mes très chers amis de *Paris* : **Mina, Khadija, Yasmina, Madjid, Neila, Véronique** et sa sœur **Valérie, Latifa** et **Richard, Anatole, Françoise** et **Michel, Serge, Josette** et **Miguel, Nanou** et **Kiké, Nazim** et son frère **Nacer**.
- Mes très chers amis d'*Alger* **Mohamed** et **Amar**.
- Ma très chère amie d'*Irlande* **Clare** qui a toujours été si gentille et si généreuse.
- Ma très chère amie de *Belgique* **Amal** que j'ai eu la chance de rencontrer lors de mon passage au LAAS à Toulouse avec qui nous avons partagé des nuits blanches au bureau à travailler et à se soutenir mutuellement puisque c'était la phase terminale de rédaction de son manuscrit de thèse.
- Mes très chers amis du *Brésil* que j'ai rencontré lors de mon passage au Prism et avec qui nous avons partagé des moments formidables : **Joao, Celestino, Kleber, Mauro** et sa femme **Anne-lise**.
- Mes très chers amis d'*Amérique du Nord* : **Salima, Yasmina, Safia** et son mari, **Jories** et en particulier **Marc** qui a été là depuis le début de cette aventure et qui a toujours su trouver les mots pour m'encourager.
- Mes très chers amis de *Guyane* **Taha** et sa femme **Mounsouk**
- Mes très chers amis de *Toulouse*, du laboratoire LAAS avec qui j'ai aussi passé des moments superbes et inoubliables et avec qui j'ai partagé les plaisirs de la bonne gastronomie du sud-ouest. Je n'oublierai jamais cette ambiance chaleureuse dans ce laboratoire Multimédia où nous travaillions dans la joie et la bonne humeur, ainsi que les fous rires mémoriaux que nous avons eu, je pense en particulier à **Laurent Robert, Philippe, Laurent Ostiz** et **Michel**. Je n'oublierai pas **Michel Diaz** qui m'a accueilli au sein de son groupe ainsi que tous ceux que j'ai rencontré là.
- Mes très chers amis de la *French Riviera* : **Hélène, Nadia, Guy** et **Odile, Emmanuelle** et sa famille.

Je voudrai également remercier **Christian Dupont** que j'ai eu la chance de rencontrer six mois avant la fin de ma thèse et qui a su sans aucune hésitation me donner les bons conseils pour aller au bout de ce chemin difficile.

Je ne peux pas non plus oublier ma très chère **famille du W3C**, qui a été très présente, en particulier durant la dernière ligne droite de ma thèse ; je pense à :

- **Josef** que j'ai rencontré lors de mes débuts à Eurécom,
- **Bert** qui a eu la patience de lire mes chapitres de thèse ;
- **Rigo** qui a durant les derniers mois toujours été là pour me sortir de ma bulle afin de respirer un bon bol d'air pour me ressourcer et voir du monde. Avec qui j'ai

partagé de nombreuses discussions qui me faisaient oublier pendant quelques heures mes chapitres de thèse.

- **Sijtsche, Max et Philipp** pour avoir été là, écouter mes plaintes et partager ma bonne et mauvaise humeur.

Une pensée sincère à **mon très cher et adoré Chaals** qui est arrivé dans ma vie au bon moment, pour moi (mais peut être pas pour lui !!!), pour me soutenir durant les trois derniers mois de ma thèse, les mois les plus dures. Il a su m'écouter, supporter ma mauvaise humeur, mes angoisses et mon caractère insupportable. Il a su me soutenir et s'occuper de moi. Il a été mon «beau soleil» et m'a apporté de la lumière dans ma vie pas très gaie à cette phase cauchemardesque qu'est la fin de thèse. Je lui dédie également cette thèse en témoignage de son affection, de son amour, de son soutien moral, de sa patience, de sa gentillesse, de sa bonté et de sa grande générosité.

Je voudrai également remercier **tonton Ahmed**, les familles **Gilles, Drappier, Vignand, Boussebissi, Zidelmel, Hanane** et mes cousines **Nedouche et Soraya**, qui n'ont jamais cessé de se soucier de mon sort, et qui me soutiennent depuis toute petite pour certains et depuis mon arrivée en France en 1992 pour d'autres.

Je remercie également toutes les personnes de l'Institut Eurécom qui m'ont accompagnée tout au long de ces trois années, je pense en particulier à **Agnes Rougier, Stephane Decrauzat, Irfan, David Marry, Karina, Raymond** et sa femme **Cathy**,...

Ne pouvant malheureusement pas citer toutes les personnes que j'ai rencontré durant mon parcours et qui ont contribué d'une façon ou d'une autre, de près ou de loin, à l'aboutissement de cette thèse, je leur dis à toutes merci d'avoir été là à cette instant précis où je les ai rencontrées et où ils m'ont apportée cette aide qui a sûrement contribué à aller au bout de cette aventure : **Ma Thèse !!!!!!!!!!!!!**

Résumé

Les réseaux et systèmes informatiques sont devenus des outils indispensables au fonctionnement des entreprises. Ils sont aujourd'hui déployés dans tous les secteurs professionnels : la banque, les assurances, la médecine ou encore le domaine militaire. Initialement isolés les uns des autres ces réseaux sont à présent interconnectés et le nombre de points d'accès ne cessent de croître. Ce développement phénoménal s'accompagne naturellement de l'accroissement du nombre d'utilisateurs. Ces utilisateurs, connus ou non, ne sont pas forcément pleins de bonnes intentions vis-à-vis de ces réseaux. Dès lors que ces réseaux sont apparus comme des cibles d'attaques potentielles, les sécuriser est devenu incontournable. La gestion de la sécurité de ces réseaux implique :

1. la mise en place des mécanismes de sécurité préventifs pour protéger les données et les ressources du système ou du réseau contre tout accès non autorisé ou abusif ;
2. le déploiement des outils de sécurité pour détecter les attaques qui réussiraient à porter atteinte à la sécurité du réseau ou du système malgré les mesures préventives ;
3. la mise en place des mécanismes de réponse aux attaques détectées.

Prévenir de toutes les violations de sécurité apparaît quelque peu irréel. En effet, il est pratiquement impossible d'avoir un réseau complètement sûr et de le protéger contre toutes les attaques possibles. Malgré la mise en place de politiques préventives de sécurité, les réseaux et les systèmes restent sujets à des attaques potentielles entreprises par des personnes qui réussissent à contourner ces mesures préventives par des comportements frauduleux. Par conséquent, dans ce travail, nous nous sommes focalisés sur le second aspect qui concerne les *mécanismes de détection d'intrusions*.

L'évolution des réseaux, en termes de nombre d'utilisateurs et de services, les rend toujours plus complexes et par conséquent vulnérables à de nouveaux types d'attaques. Les systèmes de détection d'intrusions existants ne sont pas facilement adaptables à cette complexité croissante des interactions et des comportements utilisateurs. Un autre facteur important est la mobilité des utilisateurs qui impose une gestion dynamique de la base de connaissances du comportement et des activités de ces utilisateurs. Par conséquent, les systèmes de détection d'intrusions doivent prendre en compte cette dynamique. Or, un système classique statique de détection d'intrusions ne peut pas traiter facilement ces nouveaux aspects de mobilité. Cette mobilité, entraîne naturellement l'apparition de problèmes de sécurité tels que de nouvelles vulnérabilités et de nouveaux types d'intrusions, qui doivent être inévitablement considérés.

La détection d'intrusions requière ainsi plus de sophistication pour s'adapter aux nouvelles contraintes des réseaux qui évoluent vers une nouvelle génération intégrant

plus d'"intelligence". En effet, les réseaux tendent vers plus de flexibilité, et se doivent d'intégrer de nouvelles fonctionnalités permettant de contrôler et de modifier dynamiquement leur comportement. C'est le cas des réseaux actifs qui promettent de gérer de manière plus efficace et plus flexible les éléments réseaux facilitant ainsi l'intégration de nouveaux services.

Les systèmes de détection d'intrusions existants ont été conçus pour des environnements connus et bien définis. Ils ne sont donc pas adaptés à des environnements dynamiques et c'est là leur principale faiblesse. Dans ce type d'environnement où les besoins en sécurité sont en perpétuelle augmentation, flexibilité et adaptativité deviennent des critères primordiaux.

L'objectif de ce travail est donc de rechercher une solution de détection d'intrusions souple et flexible pour s'adapter aux changements et à l'évolution complexe et non-prédictive des réseaux. Ces caractéristiques que nous recherchons ont également fait l'objet de nombreuses recherches notamment dans le cas de la gestion de réseaux et ce pour résoudre des problèmes tels que la gestion de fautes et l'analyse de trafic. Des solutions récentes ont montré que des approches à base d'agents étaient adaptées pour résoudre des problèmes complexes.

Dans la perspective d'offrir un système de détection d'intrusions pour les réseaux actuels mais aussi pour la nouvelle génération, nous proposons une nouvelle génération de systèmes de détection d'intrusions fondés sur les systèmes multi-agents pour modéliser et implémenter une détection d'intrusions *intelligente*.

Table des matières

CHAPITRE 1 INTRODUCTION GENERALE.....	1
1.1 INTRODUCTION	1
1.2 POSITION DU PROBLEME	1
1.3 OBJECTIF DE LA THESE.....	2
1.4 ORGANISATION DE LA THESE	4
CHAPITRE 2 LA GESTION DE LA SECURITE DES RESEAUX	5
2.1 INTRODUCTION	5
2.2 SERVICES DE SECURITE	6
2.3 ATTAQUES DE SECURITE	6
2.3.1 <i>Classification des attaques</i>	6
2.3.2 <i>Description d'attaques</i>	8
2.4 DETECTION D'INTRUSIONS.....	10
2.4.1 <i>Audit de sécurité</i>	10
2.4.2 <i>Caractéristiques des systèmes de détection d'intrusions</i>	14
2.4.3 <i>Intrusions</i>	17
2.4.4 <i>Méthodes de détection d'intrusions</i>	17
2.4.5 <i>Systèmes de détection d'intrusions existants</i>	25
2.4.6 <i>Caractéristiques d'un bon système de détection d'intrusions</i>	30
2.5 CONCLUSION	32
CHAPITRE 3 UNE APPROCHE MULTI-AGENTS POUR LA GESTION DE SECURITE.....	33
3.1 INTRODUCTION	33
3.2 LES SYSTEMES MULTI-AGENTS ET LA TECHNOLOGIE AGENT INTELLIGENT-.....	33
3.2.1 <i>Concepts de base</i>	34
3.2.2 <i>Les différents modèles d'agents</i>	40
3.2.3 <i>Méthodologies de conception d'un système multi-agents</i>	43
3.3 EVOLUTION DES BESOINS DE SECURITE.....	46
3.4 EFFICACITE DE LA DETECTION DES ATTAQUES DE SECURITE ET LES PROPRIETES AGENTS INTELLIGENTS.....	47
3.5 MODELE DU SYSTEME DE GESTION DE SECURITE	49
3.6 CONCLUSION	52
CHAPITRE 4 GESTION DES EVENEMENTS DE SECURITE.....	55
4.1 INTRODUCTION	55
4.2 MODELE CONCEPTUEL DE DONNEES	56
4.2.1 <i>Les événements</i>	57
4.2.2 <i>Les filtres d'événements</i>	64
4.2.3 <i>Les suites d'événements</i>	65
4.2.4 <i>Les schémas d'attaques</i>	68
4.2.5 <i>Phénomène de l'oubli</i>	71
4.3 MODELE CONCEPTUEL DE TRAITEMENT DES DONNEES	72
4.3.1 <i>Le langage descriptif</i>	73
4.3.2 <i>Le Langage symbolique</i>	77
4.3.3 <i>Traitement des événements pour la reconnaissance de schémas d'attaque</i>	80
4.4 CONCLUSION	85
CHAPITRE 5 GESTION DES POLITIQUES DE SECURITE.....	87
5.1 INTRODUCTION	87
5.2 DEFINITION D'UNE POLITIQUE DE GESTION DE SECURITE	89
5.2.1 <i>Notion de Domaine</i>	89
5.2.2 <i>Notion de Rôle</i>	89

5.2.3	<i>Types de politiques de sécurité</i>	89
5.3	HIERARCHIE ET RAFFINEMENT D'UNE POLITIQUE DE SECURITE.....	90
5.3.1	<i>Le modèle de Wies</i>	90
5.3.2	<i>Le modèle de Heilbronner</i>	91
5.3.3	<i>Autres modèles</i>	92
5.3.4	<i>Notre modèle</i>	92
5.4	SPECIFICATION D'UNE POLITIQUE DE SECURITE.....	96
5.4.1	<i>Formalisme de Sloman</i>	96
5.4.2	<i>Formalisme de Heilbronner</i>	97
5.4.3	<i>Formalisme proposé</i>	97
5.5	CONCLUSION.....	102
CHAPITRE 6 LE SYSTEME MULTI-AGENTS.....		105
6.1	INTRODUCTION.....	105
6.2	MODELE ORGANISATIONNEL DU SYSTEME MULTI-AGENTS.....	106
6.2.1	<i>Identification des rôles</i>	106
6.2.2	<i>Interactions entre les rôles</i>	109
6.3	STRUCTURE ORGANISATIONNELLE DU SYSTEME MULTI-AGENTS.....	110
6.4	MODELE FONCTIONNEL D'UN AGENT DE SECURITE.....	113
6.4.1	<i>Fonction d'interaction</i>	114
6.4.2	<i>Fonction de délibération</i>	116
6.5	LANGAGE DE COMMUNICATION.....	128
6.6	CONCLUSION.....	130
CHAPITRE 7 DISTRIBUTION DE LA DETECTION DES SCHEMAS D'ATTAQUE.....		131
7.1	INTRODUCTION.....	131
7.2	DISTRIBUTION DE LA DETECTION DES ATTAQUES.....	131
7.2.1	<i>L'opérateur d'itération</i>	134
7.2.2	<i>L'opérateur de séquence</i>	140
7.2.3	<i>les opérateurs non séquentiels</i>	141
7.2.4	<i>Combinaison d'opérateurs</i>	142
7.2.5	<i>Phénomène de l'oubli</i>	144
7.3	DETECTION CENTRALISEE VS DISTRIBUEE.....	144
7.4	DELEGATION DES BUTS DE DETECTION.....	151
7.5	CONCLUSION.....	152
CHAPITRE 8 IMPLEMENTATION.....		153
8.1	INTRODUCTION.....	153
8.2	LE SIMULATEUR DE MESSAGES.....	154
8.3	LE SYSTEME MULTI-AGENTS.....	155
8.3.1	<i>L'environnement de développement</i>	155
8.3.2	<i>Le système multi- agents</i>	156
8.3.3	<i>Discussion</i>	162
8.3.4	<i>Exemple de scénario</i>	168
8.4	CONCLUSION.....	169
CHAPITRE 9 CONCLUSION GENERALE.....		171
9.1	INTRODUCTION.....	171
9.2	CONTRIBUTIONS.....	172
9.3	PERSPECTIVES.....	173
PUBLICATIONS.....		175
REFERENCES BIBLIOGRAPHIQUES.....		177
ANNEXE A.....		189

Table des figures

FIGURE 1 : LES DIFFERENTS POINTS DE VUE DE GESTION DE SECURITE	51
FIGURE 2 : LA REPRESENTATION EN COUCHES DU MODELE D'INFORMATION POUR LA GESTION DE LA SECURITE RESEAU	52
FIGURE 3 : TRAITEMENT DES EVENEMENTS DE SECURITE	56
FIGURE 4 : SELECTION DES EVENEMENTS ET SUITES D'EVENEMENTS.....	57
FIGURE 5 : LES DIFFERENTS FLUX D'ACTIVITE DANS UN RESEAU D'ENTREPRISE.....	60
FIGURE 6 : LA CLASSE PRINCIPALE D'EVENEMENT DE SECURITE.....	62
FIGURE 7 : LES CLASSES D'EVENEMENTS DE SECURITE.....	63
FIGURE 8 : LA CLASSE FILTRE D'EVENEMENT	64
FIGURE 9 : SELECTION DES EVENEMENTS	65
FIGURE 10 : LES CLASSES DE SUITES D'EVENEMENTS	66
FIGURE 11 : SELECTION DES SUITES D'EVENEMENTS	69
FIGURE 12 : LA CLASSE SCHEMA D'ATTAQUE	71
FIGURE 13 : FENETRE TEMPORELLE GLISSANTE	72
FIGURE 14 : SIMULATION DE LA FENETRE TEMPORELLE.....	72
FIGURE 15 : PROCESSUS DE TRAITEMENT DES EVENEMENTS.....	81
FIGURE 16 : PREMIERE OPTION DE FILTRAGE - UN SEUL FILTRE	82
FIGURE 17 : DEUXIEME OPTION DE FILTRAGE – DES FILTRES SEPARES.....	82
FIGURE 18 : FILTRAGE DES SUITES D'EVENEMENTS PAR RAPPORT A L'ATTRIBUT <i>SOURCE</i>	84
FIGURE 19 : ROLE D'UNE POLITIQUE DE SECURITE.....	88
FIGURE 20 : MODELE DE WIES.....	91
FIGURE 21 : LE MODELE DE HEILBRONNER.....	92
FIGURE 22 : RAFFINEMENT ET HIERARCHIE D'UNE POLITIQUE	94
FIGURE 23 : CLASSES DES POLITIQUES DE SECURITE.....	99
FIGURE 24 : INSTANCIATION D'UN SCHEMA D'ATTAQUE.....	101
FIGURE 25 : INTERACTIONS ENTRE LES ROLES	110
FIGURE 26: ARCHITECTURE FONCTIONNELLE DU SYSTEME MULTI-AGENTS.....	112
FIGURE 27: INTERACTIONS ENTRE LES DIFFERENTES FONCTIONS DE L'AGENT.....	114
FIGURE 28 : LES CLASSES DE CROYANCES	124
FIGURE 29 : LA CLASSE DE SUSPICION	124
FIGURE 30 : LES CLASSES DE BUTS	125
FIGURE 31 : INTERACTIONS ENTRE LES DIFFERENTS ELEMENTS D'INFORMATION	127
FIGURE 32: VUE GENERALE DE L'IMPLEMENTATION.....	153
FIGURE 33: LA METHODE DE CREATION DE MESSAGES.....	154
FIGURE 34 : SIMULATEUR DE MESSAGES.....	155
FIGURE 35 : L'ARCHITECTURE D'UN AGENT HYBRIDE DANS DIMA.....	156
FIGURE 36: MODULES D'UN AGENT SURVEILLANT LOCAL	157
FIGURE 37: LA METHODE DE SELECTION DE MESSAGES A FILTRER.....	159
FIGURE 38 : FILTRAGE DES MESSAGES RESEAUX.....	159
FIGURE 39: LA METHODE DE FILTRAGE D'EVENEMENTS	159
FIGURE 40: LE MOTEUR DE DELIBERATION	160
FIGURE 41: LES MODULES D'UN AGENT GESTIONNAIRE EXTRANET/INTRANET	161
FIGURE 42 : LES MODULES D'UN AGENT GESTIONNAIRE DES POLITIQUES DE SECURITE	162
FIGURE 43 : EXEMPLE DE SCENARIO	169

Chapitre 1 Introduction Générale

1.1 Introduction

Les réseaux et systèmes informatiques sont devenus des outils indispensables au fonctionnement des entreprises. Ils sont aujourd'hui déployés dans tous les secteurs professionnels : la banque, les assurances, la médecine ou encore le domaine militaire. Initialement isolés les uns des autres ces réseaux sont à présent interconnectés et le nombre de points d'accès ne cessent de croître. Ce développement phénoménal s'accompagne naturellement de l'accroissement du nombre d'utilisateurs. Ces utilisateurs, connus ou non, ne sont pas forcément pleins de bonnes intentions vis-à-vis de ces réseaux. Ils peuvent exploiter les vulnérabilités des réseaux et systèmes pour essayer d'accéder à des informations sensibles dans le but de les lire, les modifier ou les détruire, pour porter atteinte au bon fonctionnement du système ou encore tout simplement par jeu.

Dès lors que ces réseaux sont apparus comme des cibles d'attaques potentielles, les sécuriser est devenu incontournable. De nombreuses solutions ont été proposées pour prévenir des types d'attaques spécifiques; cependant de nombreuses méthodes subsistent pour obtenir des droits accès non autorisés [Ko et al. 1993]. Ainsi, avoir un réseau complètement sécurisé et le protéger contre toutes les attaques possibles est, en pratique, irréalisable. De plus, dû aux nombreux efforts réalisés dans l'architecture des réseaux de communications ouverts, il n'est pas possible de déployer de nouveaux réseaux à la fois sécurisés et fermés [Ko et al.1993]. Par conséquent, il est nécessaire de pouvoir détecter les violations de sécurité lorsqu'elles se produisent. Cela est rendu possible grâce au mécanisme de détection d'intrusions.

1.2 Position du problème

La nouvelle génération de réseaux tend vers plus de flexibilité, et se doit d'intégrer de nouvelles fonctionnalités permettant de contrôler et modifier dynamiquement leur comportement. C'est le cas des réseaux actifs qui ont été conçus pour permettre à des utilisateurs d'injecter des programmes personnalisés dans les nœuds réseaux pour modifier leur comportement. Ils autorisent ainsi le chargement de nouveaux services à la demande dans l'infrastructure [Tennenhouse et al. 1997] [Smith et al. 1997]. Ces réseaux actifs promettent de gérer de manière plus efficace et plus flexible les éléments réseaux facilitant ainsi l'intégration de nouveaux services. [Kaplan et al. 200].

L'évolution des réseaux, en termes de nombre d'utilisateurs et de services, les rend toujours plus complexes et par conséquent vulnérables à de nouveaux types d'attaques. Les systèmes de détection d'intrusions existants ne sont pas facilement adaptables à cette complexité croissante, des interactions et des comportements utilisateurs. Un autre facteur aggravant est la mobilité des utilisateurs. Les caractéristiques nomades des utilisateurs et les besoins correspondants en termes de ressources réseaux sont imprévisibles [Alagha and Labiod 1999]. Pour un utilisateur fixe, la base de connaissances du comportement et des activités est statique. A contrario, le comportement et les activités d'un nomade varient en fonction de la mobilité. Cette mobilité impose donc une gestion dynamique de la base de connaissances. Par conséquent, le système de détection d'intrusions doit prendre en compte ces nouvelles connaissances. Or, un système classique statique de détection d'intrusions ne peut pas traiter facilement ces nouveaux aspects de mobilité. Cette mobilité, entraîne naturellement l'apparition de problèmes de sécurité tels que de nouvelles vulnérabilités et de nouveaux types d'intrusions, qui doivent être inévitablement considérés.

Jusqu'alors, les éléments des réseaux associés à la détection d'intrusions avaient des comportements assez simples, utilisant une entité centralisée (gestionnaire, directeur...) et des entités distribuées sur les entités à surveiller. C'est le cas dans certains systèmes de détection d'intrusions. Dans ces systèmes, la tâche globale du gestionnaire centralisé consiste à communiquer avec les entités de surveillance afin de collecter et analyser leurs rapports d'information pertinents. En se basant sur cette information, le gestionnaire détermine si des activités intrusives ont eu lieu. Dans le cas de la détection d'anomalies, il envoie des alarmes et/ou des rapports de sécurité à l'administrateur. Les entités de surveillance supervisent localement leurs hosts. Elles peuvent communiquer entre elles afin de s'échanger des informations sur des utilisateurs lorsqu'ils se déplacent dans le réseau. Toutes ces entités n'ont pas la connaissance sur les hosts et systèmes voisins. Elles délèguent la tâche de gestion globale à un niveau supérieur. Une conséquence évidente est l'augmentation du temps de réponse du système, dûe aux échanges excessifs entre l'entité centrale et les entités de surveillance, notamment dans le cas de réseaux longue distance (WAN).

L'objectif de ce travail sera de rechercher une solution de gestion de sécurité, en particulier de détection d'intrusions souple et flexible pour s'adapter aux changements et à l'évolution complexe et non-prédictive des réseaux. Ces caractéristiques que nous recherchons ont également fait l'objet de nombreuses recherches notamment dans le cas de la gestion de réseaux et ce pour résoudre des problèmes tels que la gestion de fautes et l'analyse de trafic.

Des solutions récentes [Rao and Georgeff 1995] [Oliveira 1998] [Alagha and Labiod 1999] ont montré que des approches à base d'agents étaient adaptées pour résoudre des problèmes complexes.

1.3 Objectif de la thèse

La détection d'intrusions requière toujours plus de sophistication pour s'adapter aux nouvelles contraintes des réseaux qui évoluent vers une nouvelle génération intégrant plus d'"intelligence".

L'objectif de cette thèse sera donc de concevoir un système capable de gérer la sécurité, en particulier la détection d'intrusions dans les réseaux actuels mais aussi dans la nouvelle génération. Dans cette perspective, nous proposons d'utiliser un

système multi-agents pour modéliser et implémenter une détection d'intrusions *intelligente*.

L'approche que nous proposons a pour objectif de remplir les fonctions nécessaires à la détection des attaques de sécurité. Pour réaliser cela, il nous faut proposer :

- un modèle pour spécifier les politiques de sécurité, qui permettront d'identifier les schémas d'attaques à détecter ;
- un modèle pour distribuer la détection de ces schémas d'attaques aux différentes entités du système multi-agents ;
- un modèle pour détecter ces attaques qui se caractérisent par des suites d'événements spécifiques.

Pour remplir ces fonctions, nous concevons notre approche autour de trois modèles : un modèle de politiques de sécurité, un modèle événementiel et un modèle agent.

Le premier modèle permet de spécifier les politiques de sécurité. Dans ce modèle nous utilisons le langage de spécification des politiques de gestion proposé par [Mariott and Sloman 1996] [Lupu and Sloman. 1997d] [Lupu et al. 1997]. Nous enrichissons ce langage afin de permettre la définition des schémas d'attaque.

Le deuxième modèle représente le modèle opérationnel de notre système. Il permet le traitement des événements en vue d'identifier les attaques qui se produisent dans le réseau. Dans ce modèle, nous proposons un langage pour non seulement la représentation des événements et des schémas d'attaque mais aussi pour le filtrage des événements et la reconnaissance des schémas d'attaque.

Le troisième modèle représente le modèle abstrait de notre système, construit sur le modèle BDI [Rao and Georgeff 1991a] [Rao and Georgeff 1991b] [Rao and Georgeff 1992], qui a été conçu par la communauté agents pour représenter l'état mental d'un agent. Dans ce modèle, nous représentons les attitudes mentales des agents qui guident leurs comportements pour la détection des attaques.

Les trois modèles s'articulent de la manière suivante. Le modèle de politiques guide le modèle BDI, alors que le modèle événementiel est piloté à un niveau plus abstrait par le modèle BDI.

Pour concevoir notre système multi-agents, nous identifions :

- les rôles nécessaires à la reconnaissance des schémas d'attaque,
- l'organisation des agents dans notre système,
- l'architecture interne des agents.

En ce qui concerne le dernier point, nous optons pour une architecture d'agent hybride qui allie des capacités délibératives à des capacités réactives afin de détecter les attaques complexes et de réagir rapidement aux attaques simples.

1.4 Organisation de la thèse

L'organisation de ce document reflète la démarche que nous avons adoptée lors de la réalisation de notre travail. Le document est composé de neuf chapitres incluant ce chapitre d'introduction générale.

Le chapitre 2 propose un état de l'art sur la gestion de la sécurité, plus précisément sur les attaques de sécurité et les mécanismes de détection d'intrusions. Une synthèse de l'existant nous permet d'identifier les caractéristiques d'un bon système de détection d'intrusions.

Le chapitre 3 décrit brièvement les systèmes multi-agents, notamment leurs propriétés, les différentes architectures d'agents ainsi que les méthodologies de conception de systèmes multi-agents. Une discussion est proposée sur l'apport des propriétés des systèmes multi-agents pour une détection d'intrusions efficace et flexible. Finalement, une vue globale du modèle que nous proposons est présentée.

Le chapitre 4 propose un modèle de gestion des événements de sécurité. Il définit un langage pour représenter et traiter les différents éléments du modèle.

Le chapitre 5 décrit le modèle de gestion des politiques de sécurité, qui permettra d'instancier des schémas d'attaque et de définir la portée géographique de leur détection.

Le chapitre 6 présente le système multi-agents proposé pour identifier les schémas d'attaques instanciés. Dans ce chapitre, nous identifions les différents rôles d'agents, la structure organisationnelle du système multi-agents et l'architecture interne des agents.

Le chapitre 7 décrit la distribution de la détection des schémas d'attaque dans notre système multi-agents.

Le chapitre 8 présente brièvement une implémentation du système multi-agents conçu pour la détection de schémas d'attaque connus.

Nous finirons ce document par le chapitre présentant nos conclusions ainsi que les perspectives de notre travail.

Chapitre 2 La gestion de la sécurité des réseaux

2.1 Introduction

La gestion de la sécurité d'un réseau ou d'un système implique :

1. la mise en place des mécanismes de sécurité préventifs pour protéger les données et ressources du système ou réseau contre tout accès non autorisé ou abusif ;
2. le déploiement des outils de sécurité pour détecter les attaques qui réussiraient à porter atteinte à la sécurité du réseau ou système malgré les mesures préventives ;
3. la mise en place des mécanismes de réponse aux attaques détectées.

Prévenir de toutes les violations de sécurité apparaît quelque peu irréal. En effet, il est pratiquement impossible d'avoir un réseau complètement sûr et de le protéger contre toutes les attaques possibles. Malgré la mise en place de politiques préventives de sécurité, les réseaux et systèmes restent sujets à des attaques potentielles entreprises par des personnes qui réussissent à contourner ces mesures préventives par des comportements frauduleux.

Pour évaluer et assurer les besoins de sécurité d'une entreprise, il faut considérer trois aspects :

- les services de sécurité,
- les attaques de sécurité,
- les mécanismes de sécurité pour la prévention, la détection des attaques et la réponse à ces attaques.

Dans le cadre de cette thèse, nous nous intéressons au troisième aspect : les mécanismes de détection d'intrusions.

Nous allons donc dans ce chapitre, commencer par introduire les services de sécurité, puis nous allons aborder les attaques de sécurité, et enfin nous étudierons les mécanismes de détection d'intrusions.

2.2 Services de sécurité

Un *service de sécurité* est un service qui accroît la sécurité des systèmes de traitement des données et des transferts d'informations d'une organisation [Stallings 1995]. Il existe six services de sécurité génériques [Ford 1994] [Stallings 95] :

- l'*authentification* : fournit une assurance de l'identité d'une entité (une personne ou un système).
- le *contrôle d'accès* : protège contre les utilisations ou manipulations non autorisées des ressources.
- la *confidentialité* : protège contre la révélation d'informations à des entités non autorisées.
- l'*intégrité des données* : protège contre la modification, la destruction et la substitution de données sans autorisation.
- la *non répudiation* : au cours d'une communication, il peut arriver que l'un des deux interlocuteurs nie avoir participé à l'échange d'information. Ce service permet de protéger l'autre interlocuteur.
- la *disponibilité* : protège les utilisateurs légitimes contre une indisponibilité des ressources, services et informations.

Dans le cadre de ce travail, nous nous limiterons à la définition de ces six services de sécurité.

2.3 Attaques de sécurité

Une attaque peut être définie comme toute action ou ensemble d'actions qui peut porter atteinte à la sécurité des informations d'un système ou réseau informatique. Etant donné le nombre important d'attaques possibles, nous allons d'abord commencer par les classer puis nous en présenterons quelques-unes.

2.3.1 Classification des attaques

Malgré la diversité des attaques de sécurité, nous retiendrons quatre classifications possibles.

Première classification

Un système informatique peut être attaqué :

- soit par des utilisateurs internes, dans le but d'abuser de leurs droits et privilèges, on parlera alors d'*attaques internes* ;
- soit par des utilisateurs externes qui essaient d'accéder à des informations ou ressources de manière illégitime et non autorisée et dans ce cas on parlera d'*attaques externes*.

Deuxième classification

C'est la classification la plus classique, elle regroupe quatre types d'attaques. Ainsi une attaque peut porter atteinte à :

- la **confidentialité** des informations en brisant des règles privées,
- l'**intégrité**, en altérant les données,
- l'**authenticité** des données,
- la **disponibilité**, en rendant un système ou réseau informatique indisponible. On parle alors d'attaque de **déni de service**.

Troisième classification

Elle correspond à la classification de Stallings [Stallings 1995], qui identifie deux types d'attaques :

- les **attaques passives**,
- les **attaques actives**.

Les **attaques passives** regroupent les attaques portant atteinte à la confidentialité. Il en existe deux types :

- la **lecture de contenus de messages** confidentiels : courrier électronique, fichier transféré ;
- l'**analyse de trafic** pour déterminer la nature d'une communication : identité des "hosts" communiquants, fréquence et longueur des messages échangés.

Les attaques passives ne sont pas facilement détectables car elles n'impliquent aucune altération des informations.

Les **attaques actives** concernent celles qui entraînent une modification des données ou création de données incorrectes. Autrement dit, celles qui portent atteinte à l'intégrité, l'authenticité et la disponibilité. On retrouve alors quatre types d'attaques actives :

- l'**usurpation** : c'est lorsqu'une entité se fait passer pour une autre,
- le **rejeu** : retransmission de messages capturés lors d'une communications, et cela à des fins illégitimes,
- la **modification de messages**,
- le **déni de service**,

Quatrième classification

Les attaques de sécurité peuvent également être classées en termes :

- d'**attaques réseaux** : leur but principal est d'empêcher les utilisateurs d'utiliser une connexion réseau, de rendre indisponible une machine ou un service et de surveiller le trafic réseau dans le but de l'analyser et d'en récupérer des informations pertinentes.
- d'**attaques systèmes** : ce sont des attaques qui portent atteinte au système, comme par exemple effacer des fichiers critiques (tel que le fichier "password") ou modifier la page web d'un site dans le but de le discréditer ou tout simplement le ridiculiser.

2.3.2 Description d'attaques

Dans la littérature, nous retrouvons un grand nombre d'attaques mais, dans le cadre de cette thèse nous nous limiterons à une brève description de quelques-unes d'entre elles.

2.3.2.1 Attaques réseaux

- **IP spoofing** : l'*IP spoofing* [Cert 1995] [Heberlein 1996] est l'action d'envoyer des paquets avec une autre adresse source que celle de l'expéditeur réel afin de laisser le serveur croire que les paquets proviennent d'une autre machine, de préférence une machine à qui il est permis d'établir une connexion. En d'autres termes, l'agresseur change l'adresse de sa machine pour faire croire qu'il est un client certifié par le serveur. Le but de cette attaque est de se faire passer pour un client certifié afin d'obtenir des droits d'accès et surtout un accès root.
- **ICMP flooding** : connue aussi sous le nom de "**Smurf attack**" [Fenzi 1998] [Cert 1998], est une attaque récente et très dangereuse. Son principe est d'envoyer un grand nombre de paquets ICMP "echo request" (qui sont essentiellement des requêtes "ping") dont l'adresse IP de destination est une adresse "broadcast" du réseau cible. Son but est de noyer ainsi le réseau et de le rendre indisponible. En effet, lorsque le routeur reçoit les paquets ICMP, il les diffuse sur toutes les machines du réseau en congestionnant ainsi le réseau, vu l'importance du trafic. De plus, l'adresse source est souvent modifiée et remplacée par l'adresse d'une autre machine, qui sera à son tour victime car elle recevra toutes les réponses "ICMP reply" engendrées par les paquets ICMP "echo request".
Il existe un cousin de cette attaque, appelé "**fraggle attack**", qui utilise le même principe que l'**ICMP flooding**, sauf qu'au lieu que cela soit des paquets "ICMP echo", se sont des paquets "UDP echo".
- **TCP SYN flooding** : elle est également connue sous le nom de "SYN attack" [Shuba et al. 1997] [Cert 1996]. Son objectif est de rendre indisponible un service TCP offert sur une machine. Le principe de cette attaque est de créer des connexions TCP semi-ouvertes sur la machine cible afin de remplir la file d'attente où sont stockées les demandes d'ouverture de connexions. L'attaquant envoie un grand nombre de requêtes SYN à la machine cible et remplace son adresse source avec l'adresse d'une machine indisponible ou inexistante afin que les réponses SYN/ACK ne soient jamais reçues et que donc les messages ACK ne soient jamais générés, ce qui signifie que la file d'attente restera pleine. Les

conséquences de cette attaque est que toutes les requêtes arrivant sur le port TCP cible seront ignorées et de ce fait le service fourni sur ce port sera indisponible. Dans certains cas, la machine peut aussi devenir indisponible.

- **Doorknob Rattling** : elle se traduit par des essais répétés de "login" (login/password) sur plusieurs machines différentes dans le but d'obtenir un accès à un compte.
- **Ping of Death** : cette attaque se traduit par l'envoi de paquets IP dont la taille excède la longueur maximale (65507 octets) autorisée par le protocole IP. Ce qui "rebootera" ou rendra indisponible la machine cible.
- **Sniffing** : elle se traduit par l'observation et l'analyse du trafic réseau. Son but est d'obtenir des informations pertinentes afin de préparer d'autres attaques. Ces informations peuvent être :
 - ◆ les mots de passes et les noms de "login",
 - ◆ les adresses IP des machines,
 - ◆ les fonctionnalités des machines, comme par exemple reconnaître qu'une machine est un serveur de fichier,
 - ◆ les noms de communautés SNMP, etc.

2.3.2.2 Attaques systèmes

- **Virus et bombe logique** : un *virus* est un programme écrit dans le but de détruire un système (exemple effacer des disques, des fichiers pertinents, etc.); alors qu'une *bombe logique* est un programme conçu uniquement dans le but de détruire des activités lorsqu'elles sont lancées.
- **Vers** : c'est un agent autonome capable de se propager sans une action extérieure (programme ou personne) mais uniquement en utilisant les ressources d'une machine pour attaquer d'autres machines. L'exemple le plus célèbre est celui qui s'est produite en novembre 1988 [Spafford 1988], lorsqu'un étudiant lança un programme sur Internet qui était capable de se développer par lui-même à travers le réseau de serveur. Huit heures après, 2000 à 3000 machines avaient été infestées et commençaient à tomber en panne.
- **Cheval de Troie** : c'est un programme qui se cache lui-même dans un autre programme apparemment au-dessus de tout soupçon. Quand la victime lance ce programme, elle lance par la même occasion le cheval de Troie caché. Un cheval de Troie peut par exemple, lorsqu'il est exécuté, ouvrir l'accès au système à des personnes particulières ou même à tout le monde.
- **Trappe** : c'est un point d'entrée dans un système informatique qui passe au-dessus des mesures de sécurité normales. La plupart du temps, c'est un programme caché qui est souvent activé par un événement ou une action normale et dont le but est de fragiliser un système de protection ou même le rendre inefficace.
- **"Craquage" de mots de passe** : le moyen le plus classique, de "craquer" des mots de passe est d'utiliser un programme appelé "cracker". Ce programme utilise un

dictionnaire de mots et de noms propres et opère sur les mots de passe encryptés, qui se trouvent dans un fichier (par exemple le fichier UNIX /etc/passwd).

2.4 *Détection d'intrusions*

La détection d'intrusions est un terme général qui désigne des méthodes automatiques qui, basées sur l'analyse de séquences d'événements temps réel et/ou enregistrés, peuvent alerter l'administrateur de sécurité de possibles violations de sécurité [Ford 1994].

La détection d'intrusions fait référence à la capacité d'un système informatique de déterminer automatiquement, à partir d'événements relevant de la sécurité, qu'une violation de sécurité se produit ou s'est produite dans le passé [Meyer et al. 1995].

Pour se faire, la détection d'intrusions nécessite qu'un grand nombre d'événements de sécurité soient collectés et enregistrés afin d'être analysés. On parle alors d'*audit de sécurité*.

Dans cette partie, nous proposons un état de l'art en matière de détection d'intrusions. Pour cela, nous allons commencer par présenter les différentes activités liées à l'audit de sécurité. Puis nous allons décrire les caractéristiques des systèmes de détection d'intrusions, et définir les types d'intrusions. Nous décrivons, ensuite, les différentes méthodes de détection d'intrusions. Enfin, après avoir présenté brièvement quelques systèmes de détection d'intrusions, nous identifions les critères auxquels doit répondre un bon système de détection d'intrusions.

2.4.1 *Audit de sécurité*

Lorsqu'un utilisateur (administrateur ou opérateur), un processus ou une application effectue une opération sur un système informatique, cela se traduit par une suite d'*actions* effectuées par le système. Ces actions sont appelées *activités système* [Mé et Alanou 1996]. Comme ces activités systèmes se produisent à un moment donné, elles sont appelées *événements*.

Dans ce cas, on parle souvent d'*événements de sécurité*. Un *événement de sécurité* est tout événement qui essaye :

- soit de modifier l'état de sécurité du système (exemple : changer le niveau de sécurité d'un objet, ou un "password" utilisateur, etc.);
- ou de violer la politique de sécurité du système (exemple : essais répétés de login, etc.).

Ces événements de sécurité, lorsqu'ils sont analysés, peuvent fournir des conclusions significatives car une séquence de ces événements traduit, très souvent, une attaque. Ainsi pour une détection efficace, la détection d'intrusions nécessite qu'un grand nombre d'événements de sécurité potentiels soient enregistrés.

Le mécanisme permettant la collecte de ces événements, afin d'en faire une analyse a posteriori pour détecter d'éventuelles attaques, est appelé *audit de sécurité*.

D'après [Gallagher 1987], le mécanisme d'audit d'un système informatique a cinq buts de sécurité importants. Il doit permettre :

1. de passer en revue les formes d'accès aux objets individuels, l'historique des accès de processus spécifiques et individuels et l'utilisation des différents mécanismes de protection supportés par le système et leur efficacité ;

2. de découvrir quels sont parmi les utilisateurs internes et externes, ceux qui contournent les mécanismes de protection ;
3. de découvrir toute utilisation de privilèges qui peut se produire lorsqu'un utilisateur assume une fonctionnalité avec des privilèges qui sont plus élevés que les siens (ex : utilisateur à administrateur).
4. d'agir comme un outil dissuasif contre les essais habituels et répétés pour contourner les mécanismes de protection système ;
5. d'assurer que les essais de contourner les mécanismes de sécurité soient enregistrés et découverts. Ainsi même si l'essai de contourner le mécanisme de protection réussit, les informations enregistrées aideraient à réparer les dégâts causés.

Un *journal d'audit de sécurité* est un fichier dans lequel sont enregistrées des séquences chronologiques d'activités utilisateurs. Ces séquences d'enregistrement du journal d'audit sont dites *traces d'audit*. Chacune de ces traces décrit des occurrences d'événements de sécurité. Pour [Mounji 1997], ces événements doivent reporter les attributs d'une action exécutée par un utilisateur sur un objet. Ces attributs incluent :

- le type de l'événement,
- l'identité du sujet et de l'objet impliqués,
- la ressource affectée par l'action,
- le résultat de cette action : succès ou échec.

Pour [Mé 1994], ces événements doivent permettre de répondre à six questions :

- quelle est l'opération effectuée?
- quel est l'auteur de l'opération ?
- quelles sont les ressources affectées du système?
- à quelle heure a été effectuée l'opération?
- quel est le lieu où s'est produite l'opération? Dans le cas d'un serveur distant, l'identifiant de ce serveur ?
- dans le cas d'échec, quelle est la raison de cet échec?

L'analyse de ces traces d'audit peut révéler une violation de sécurité, qui peut se traduire par l'envoi d'une *alarme de sécurité* à l'administrateur système ou de sécurité.

Les fonctions d'alarmes et d'audit de sécurité sont dirigées par *une politique de détection d'intrusions* et une *politique d'audit* [Ford 1994]. Cette politique d'audit est définie par l'administrateur de sécurité qui doit spécifier :

- les utilisateurs et activités systèmes à auditer ;
- le moyen d'assurer la collecte des événements spécifiques et de les traiter pour une détection efficace ;
- le cas échéant les actions à entreprendre, comme l'envoi d'alarmes.

2.4.1.1 Spécification des événements

[Mé 1994] fournit une liste non exhaustive, des informations à auditer. Il définit quatre classes d'informations :

- informations sur les accès au système,
- informations sur l'usage fait du système,
- informations sur l'usage fait des fichiers,
- informations relatives à chaque application,
- informations sur les violations éventuelles de la sécurité,
- informations sur les performances du système.

Pour plus de détails, on se référera aux documents [Mé 1994] et [Mé et Alanou 1996].

Le *livre orange* (TCSEC)[Udod 1985], qui définit, dans un ordre croissant, quatre classes de D à A, n'introduit la nécessité de mécanismes d'audit qu'à partir de la classe C2. Dans cette classe, il définit les événements et informations à auditer :

1. *événements auditables* :

- utilisation des mécanismes d'authentification et d'identification (exemple: exécution de la procédure de "login" ainsi que l'identifiant utilisé ou nom de "login") ;
- introduction d'objet dans "l'espace utilisateur" (exemple. ouverture d'un fichier) ;
- destruction d'un objet ;
- actions entreprises par l'utilisateur, l'administrateur ou l'officier de sécurité ;
- tout événement qui indique une violation imminente de la politique de sécurité système ;

Ces événements auditables doivent inclure, en plus des événements de sécurité, tous les événements nécessaires pour rétablir le système après des dommages causés [Gallagher 1987].

2. *informations auditables :*

- heure et date de l'événement ;
- identifiant unique du responsable de l'événement ;
- type d'événement ;
- succès ou échec de l'événement ;
- le nom de l'objet s'il s'agit de l'introduction de celui-ci dans "l'espace utilisateur" ou de sa destruction ;
- origine de la demande (i.e. l'identifiant du terminal) s'il s'agit de l'utilisation de mécanismes d'identification et d'authentification ;
- description des modifications faites par l'administrateur ou l'officier de sécurité sur les bases de données de sécurité système et utilisateur.

Le journal d'audit doit garder une trace de tous les événements auditables.

2.4.1.2 Collecte des événements

La collecte des événements peut être assurée grâce à :

- des sous - systèmes d'audit fournis par la plupart des systèmes d'exploitation. Ces systèmes d'audit permettent de générer certains types d'événements, dont la collecte est assurée par le noyau du système. Nous retrouvons notamment au niveau du système UNIX, un certain nombre de journaux d'audit, qui gardent une trace de tout ce qui se passe sur le système ;
- des sondes réseaux, telles que RMON, ou d'autres outils tels que "TCPDump". Ces outils permettent de récupérer les données du trafic réseau, en d'autres termes, les paquets circulant sur le réseau.

Les événements peuvent également être générés au sein d'applications [Mé 1995]. Pour cela une "boîte à outils de sécurité" devra mettre à disposition du développeur, les primitives de génération et de collecte adéquates.

2.4.1.3 Analyse des événements

L'analyse des journaux d'audit est une activité essentielle dans un environnement où des violations de sécurité sont produites, afin de détecter toute violation des règles de sécurité. Elle permet ainsi à l'officier de sécurité de :

- déterminer les auteurs de ces violations,
- déterminer les vulnérabilités du système,
- estimer les dégâts éventuels et d'assurer leur réparation,
- reconstruire l'historique de ce qui s'est passé dans le système,
- faire une comptabilité.

L'analyse de ces journaux peut se faire soit en temps réel, soit en temps différé. Cependant, afin de détecter le plus de malversations possibles et surtout dans le souci de limiter les dégâts éventuels, il est préférable que les analyses soit suffisamment fréquentes. En d'autres termes, qu'elles se fassent en quasi-temps réel. Nous reviendrons sur cet aspect plus tard (voir 2.4.2.3).

Dans le cas d'un système distribué, i.e. lorsque l'audit doit se faire sur réseau, tous les enregistrements d'audits collectés des différentes machines connectées sur le réseau, constitueront un journal d'audit global.

Afin que l'analyse de ces journaux d'audit soit fiable, il est nécessaire de protéger ces journaux contre toute tentative de modification ou de lecture par des utilisateurs non autorisés. Dans le cas d'un réseau, ce sont les transferts d'informations auditées qui doivent être sécurisés.

L'analyse de ces journaux est un travail très fastidieux. Afin de faciliter la tâche et d'aider l'officier de sécurité dans son analyse, il serait judicieux d'utiliser des outils qui analyseraient ces journaux de manière automatique, dans le but de rechercher des anomalies et de contrer au plus vite les attaques éventuelles [Cheung et al. 1995].

Ainsi, [Hoagland et al. 1995] a conçu une "boîte à outil" VAB qui fournit une interface visuelle pour feuilleter les journaux d'audit BSM Sun. Le but de cette "boîte à outils", n'est pas de détecter les malversations mais d'aider uniquement l'officier de sécurité à analyser ces journaux. Cependant il existe d'autres outils dits, *systèmes de détection d'intrusions*, qui seraient plus efficaces pour réaliser cette tâche. Nous présentons dans la section suivante, les caractéristiques de ces systèmes de détection d'intrusions.

2.4.2 Caractéristiques des systèmes de détection d'intrusions

Un système de détection d'intrusions est un système qui essaye d'identifier les essais d'intrusion aussi bien de la part des utilisateurs non autorisés (externes) que des utilisateurs autorisés (internes) qui abusent de leurs privilèges. Ainsi les systèmes de détection d'intrusions surveillent les activités systèmes pour détecter les actes malveillants et identifier alors les utilisations non autorisées et abusives [Puketza et al. 1997]. Ces systèmes de détection d'intrusions offrent une défense lorsque les vulnérabilités systèmes sont exploitées et cela sans qu'il y ait nécessité de remplacer des équipements très coûteux.

Le système de détection d'intrusions doit s'exécuter constamment sur le système, en travaillant en arrière-plan, et ne notifiant l'administrateur de sécurité que lorsqu'il détecte quelque chose qu'il considère comme suspicieux ou illégal [Price 1998].

Lors de la conception d'un système de détection d'intrusions, plusieurs paramètres sont à prendre en considération :

- l'origine des données qui seront collectées, dans le but d'être analysées ;
- la topologie du système cible à analyser, qui peut être soit centralisée, soit distribuée ;
- et le mode d'analyse, qui peut être soit temps réel, soit a posteriori (i.e. en mode "batch").

Ces paramètres sont discutés dans les paragraphes suivants.

2.4.2.1 Origine des données

Les systèmes de détection d'intrusions sont classifiés en fonction de l'origine des données qui seront exploitées pour détecter des actions intrusives. Ces informations peuvent être :

- soit des données provenant des fichiers d'audit,
- soit des données de trafic réseau.

Ainsi, on distingue trois classes de systèmes de détection d'intrusions :

- a) les systèmes de détection d'intrusions basés sur un host (host-based IDS) : Ces systèmes de détection d'intrusions [Mukherjee et al. 1994] sont désignés pour surveiller un seul host. Ils utilisent les traces d'audit du système d'exploitation de leur propre host. Ainsi ce sont les données d'audit d'un seul host qui sont utilisées pour détecter les intrusions.
- b) les systèmes de détection d'intrusions basés sur plusieurs hosts (multihost-based IDS) :
Ces systèmes de détection d'intrusions [Mukherjee et al. 1994] sont chargés de surveiller plusieurs hosts et utilisent les informations provenant des fichiers d'audit de ces hosts comme la source principale d'entrées pour détecter les intrusions. Dans ce cas c'est la corrélation des données d'audit des différents hosts qui permettra de détecter des comportements intrusifs. Cette analyse est faite sur un host séparé.
- c) les systèmes de détection d'intrusions basés sur le réseau (network-based IDS) :
Ces systèmes de détection d'intrusions [Mukherjee et al. 1994] [White et al.1996] [Price 1998] recherchent des activités suspectes dans le trafic réseau. Pour cela ils collectent et analysent les paquets de données circulant sur le réseau.

Cela dit, certains systèmes de détection d'intrusions [Mukherjee et al. 1994] [White et al.1996] basent leur analyse sur la corrélation des données provenant aussi bien des fichiers d'audit d'un ou plusieurs hosts que du trafic réseau afin de détecter le plus d'intrusions possibles.

2.4.2.2 Comparaison de la détection d'intrusions sur une topologie centralisée par rapport à une topologie distribuée

Un système centralisé unique est un système dont la configuration consiste en un host avec plusieurs terminaux qui lui sont connectés alors qu'un système distribué consiste en plusieurs hosts, la plupart du temps hétérogènes, auxquels sont reliés plusieurs terminaux.

Il existe une différence significative entre la détection d'intrusions sur un système centralisé et la détection d'intrusions dans un environnement distribué [Porras 1992]. La première différence est que dans un système unique centralisé, le processus d'audit ne se fait que sur un seul host alors que dans un environnement distribué, l'audit du réseau implique l'audit de tous les hosts constituant ce réseau [Ilgun 1992]. [Porras 1992] fournit une brève description des différences majeures qui rendent le développement des systèmes de détection en environnement distribué, quelque peu plus complexe que le développement de ceux conçus pour des systèmes centralisés.

- a) Dans un système centralisé unique, la collecte des données est assurée par un seul mécanisme d'audit, ce qui implique une consistance dans le format des enregistrements d'audit. Pour un système distribué, la collecte des données est assurée par plusieurs mécanismes d'audit ce qui entraîne deux problèmes majeurs :
 - la nécessité de comparer les enregistrements d'audit des différents composants et dont le format est loin d'être standard puisqu'il est différent pour chaque mécanisme,
 - et le besoin de coordonner les analyses sur les différents hosts.
- b) Dans un réseau distribué, la collecte, le tri et l'analyse des données, peut se faire soit de manière centralisée, en utilisant un serveur central ou alors de manière distribuée en utilisant des serveurs distribués avec plusieurs analyseurs de données. Dans ce cas, la coordination entre les différents analyseurs est nécessaire afin de produire une information d'audit globale pour le système.
- c) Le côté hétérogène du réseau distribué, ne fait que multiplier les vulnérabilités des différents systèmes, contrairement à un système centralisé unique.
- d) La quantité d'informations collectée est beaucoup plus importante dans un réseau que dans un système unique, ce qui implique l'utilisation d'algorithmes sophistiqués ainsi que des systèmes d'archivage de grande capacité.
- e) De plus, dans un système distribué, l'utilisation de protocoles supportant d'une part les données d'audit des différents composants (différents formats d'audit) et les messages échangés entre les analyseurs d'audit distants, est incontournable. Ceci implique la nécessité de sécuriser les échanges afin de préserver l'intégrité et la confidentialité des données.

2.4.2.3 Analyse temps réel et analyse en mode batch

Comme nous l'avons vu (voir 2.4.1.3), l'analyse des données peut se faire soit en temps réel, soit plus tard (i.e. en mode "batch").

Dans les systèmes de détection d'intrusions fonctionnant en mode "batch", l'analyse des données ne se fait qu'après que les données aient été collectées. Le principal avantage d'une telle analyse est qu'elle peut se faire à un moment où le taux de CPU utilisé est au plus bas et/ou sur un système totalement différent [Ilgun 1992]. Son désavantage est que certaines attaques ne seraient détectées qu'une fois que des dommages considérables auraient été causés.

Les systèmes de détection d'intrusions, fonctionnant en mode "batch", peuvent être très utiles dans des environnements où les résumés périodiques des utilisations suspectes sont suffisants [Porras 1992].

Quant aux systèmes de détection d'intrusions fonctionnant en temps réel, les données d'audit doivent être analysées dès qu'elles sont créées.

Ce mode d'analyse peut se révéler crucial pour des systèmes soucieux d'identifier les comportements utilisateurs suspects pendant qu'ils se produisent et de détecter de façon imminente toute violation de sécurité.

Cependant, l'analyse en temps réel, peut engendrer un problème de performance matérielle et logicielle. En effet, une haute fiabilité, une grande capacité de stockage et un matériel à très haut débit, deviennent des problèmes clés [Porras 1992]. De plus le temps entre le moment où une transaction d'audit se produit et le moment où elle est écrite sur le disque, doit être réduit au maximum.

2.4.3 Intrusions

Une intrusion peut être définie comme tout ensemble d'actions qui essaye de compromettre [Crosbie and Spafford 1995a] [Crosbie and Spafford 1995b] [Crosbie and Spafford 1996] [Balasubramanian et al. 1998] :

- la **confidentialité** de données (exemple vol de données, furetage, fuite d'informations par canal caché),
- l'**intégrité** de données (exemple modification illégitime de fichiers),
- la **disponibilité** d'une ressource ou d'un service (exemple occupation illégitime ou abusive de ressources, destruction illégitime ou abusive de fichiers).

En dépit des différentes formes d'intrusion, elles peuvent être regroupées dans deux classes :

- les **intrusions connues** : ces intrusions sont des attaques bien définies qui généralement exploitent des failles connues du système cible.
- les **intrusions inconnues** ou **anomalies** : ces intrusions sont considérées comme des déviations du profil normal d'un système. Elles sont détectées dès qu'il est observé un comportement anormal du système.

2.4.4 Méthodes de détection d'intrusions

La détection d'intrusions repose sur deux approches de base :

- l'*approche comportementale*,
- l'*approche par scénarios*.

L'approche comportementale, dite aussi *approche de détection d'anomalie*, définit un modèle d'activité normale en construisant un profil de l'activité utilisateur ; et toute déviation de la norme établie est considérée comme anormal [Mounji 1997].

Quant à l'approche par scénario, souvent appelée approche de détection d'abus, elle définit quelles sont les actions utilisateurs spécifiques qui constituent un abus et utilise des règles définies *a priori* pour encoder et détecter des formes d'intrusions connues [Mounji 1997].

Nous présentons dans les paragraphes suivants, les différentes méthodes utilisées dans chacune de ces deux approches.

2.4.4.1 Détection d'intrusions basée sur l'approche comportementale

La détection d'intrusions basée sur cette approche permet de détecter les intrusions *inconnues* et ne nécessite donc aucune connaissance *a priori* des intrusions. Cette approche se base sur le fait qu'un intrus ne se comporte pas de la même manière qu'un utilisateur régulier. Contrairement à l'utilisateur, qui a un comportement normal, l'intrus a un comportement anormal. Ainsi, toutes les activités intrusives sont forcément anormales [Sundaram 1996].

Pour [Kumar 1995], la détection d'anomalie (ou l'approche comportementale) essaie de quantifier le comportement habituel ou jugé acceptable et considère le comportement irrégulier comme une intrusion potentielle.

Quant à [Lane and Brodley 1997a] [Lane and Brodley 1997b] [Lane and Brodley 1997c], elle voit la détection du comportement anormal comme un problème de classification qui peut être valué de manière binaire, et dans lequel les mesures de l'activité système sont utilisées pour produire une classification de l'état du système comme normal ou anormal.

L'approche comportementale consiste à établir le profil d'un comportement normal pour une activité utilisateur particulière et à observer les déviations significatives de l'activité utilisateur courante par rapport à la forme normale établie [Mounji 1997]. Ainsi, un système de détection d'intrusions construit avec cette philosophie apprend les profils de comportement et reporte les anomalies soit à un opérateur humain ou alors à une partie de son système pour une analyse plus détaillée [Frank 1994].

Parmi les travaux développés dans le cadre de cette approche, nous retrouvons les systèmes conçus par [Lane and Brodley 1997a] et [Samfat 1996]. Leur rôle est d'apprendre un *profil utilisateur* et de l'utiliser ensuite pour détecter un comportement anormal.

En résumé, la détection basée sur cette approche se fait en deux phases :

- une phase d'apprentissage : le système apprend le comportement normal d'un sujet (utilisateur ou système). Il crée ainsi "le profil normal" d'un utilisateur à partir des données collectées.
- une phase de détection : le système examine la trace d'audit courante ou l'information réseau et la compare au profil pour vérifier s'il n'y a pas d'activité

intrusive. Si les différences entre le profil et la trace d'audit sont significatives, une alarme est déclenchée.

Pour pouvoir formaliser le comportement normal d'un sujet (utilisateur ou système), il existe plusieurs approches, que l'on peut classer dans deux groupes principaux :

- ***Modélisation du comportement normal.***
- ***Prédiction du comportement normal.***

Les principales approches de ces deux groupes sont abordées ci-après :

2.4.4.1.1 Modélisation du comportement normal

La modélisation du comportement normal peut se faire de deux façons :

- soit en utilisant un modèle statistique et c'est le modèle le plus utilisé,
- soit un système expert.

Vu l'importance de l'approche statistique, nous en présentons une description détaillée dans le paragraphe suivant.

Modèle statistique

L'approche la plus fréquemment utilisée pour la génération d'un modèle de comportement normal d'un utilisateur ou d'un système est une ***approche statistique***. Elle consiste à utiliser des mesures statistiques pour modéliser un profil de comportement et détecter des comportements intrusifs. Ces mesures peuvent être par exemple :

- le temps CPU utilisé,
- le nombre de connexions durant une certaine période de temps,
- les fichiers les plus fréquemment utilisés,
- les entrées/sorties utilisées, etc.

Chacune de ces valeurs est associée à un seuil ou à un intervalle de valeurs, dans lequel une activité est considérée comme anormale. Tout dépassement de seuil ou situation de valeurs à l'extérieur des bornes de l'intervalle indique une activité anormale.

Cette approche se base sur deux techniques [Ilgun 1992] [Porras 1992] :

- la détection basée sur la notion de seuil,
- et la détection basée sur la notion de profil statistique.

Ces deux méthodes sont expliquées dans les deux sous - paragraphes suivants.

a) Détection basée sur la notion de seuil

La détection de seuil est l'une des formes de détection les plus rudimentaires. L'idée de cette approche est d'enregistrer chaque occurrence d'un événement spécifique et lorsque le nombre d'occurrences devient trop élevé durant une certaine période de temps et dépasse un certain seuil, la présence d'un intrus est détectée. Ce seuil ainsi que l'intervalle de temps d'analyse sont définis au préalable par l'officier de sécurité. L'identification et la précision de ces deux paramètres repose entièrement sur l'expérience de l'officier de sécurité. Cette tâche n'est pas très simple car il faut définir des valeurs de sorte à minimiser le nombre de fausses alarmes et à maximiser le nombre de détections d'intrusions. En effet, définir une valeur de seuil trop faible provoquerait un taux élevé de fausses alarmes et à l'inverse une valeur de seuil trop élevée peut laisser passer des intrusions sans qu'elles ne soient détectées.

b) Détection basée sur la notion de profil statistique

Cette approche utilise les mesures statistiques pour identifier le comportement prévu d'un utilisateur ou d'un groupe d'utilisateurs [Ilgun 1992] et construire ainsi un profil utilisateur qualifié de normal. Ceci étant fait pendant la phase d'apprentissage. Durant la phase de détection, la variation entre le profil courant et le profil attendu est mesurée tout le temps. Les comportements intrusifs sont alors détectés dès que la déviation du profil courant par rapport au profil prévu est importante.

c) Le modèle de Denning

Le modèle de détection d'intrusions de Denning est indépendant de tout système et environnement d'application, et des types d'intrusions [Denning 1987]. C'est un modèle qui se veut donc générique. Il se compose de six éléments :

- **les sujets** : ce sont les initiateurs des actions effectuées sur le système. Ce sont donc les utilisateurs, les groupes d'utilisateurs, ou encore les processus agissant pour le compte de ces utilisateurs,
- **les objets** : ce sont les ressources gérées par le système tel que les fichiers, les programmes, les messages, les périphériques, etc. La granularité des objets dépendra de l'environnement d'application et du niveau de sécurité recherché. Par exemple, pour certaines applications de bases de données la granularité au niveau de l'enregistrement sera nécessaire alors que pour d'autres applications le niveau de fichier ou répertoire serait suffisant.
- **les enregistrements d'audit** : ils représentent les actions entreprises par un sujet sur un objet et sont générés par le système. Ils sont composés :
 - ◆ du sujet ayant fait l'action,
 - ◆ du type de l'action (par exemple login, logout, lecture, écriture),
 - ◆ des ressources utilisées,
 - ◆ du résultat de l'action (échec ou succès),
 - ◆ des éléments quantitatifs (par exemple, nombre de lignes ou pages imprimées, nombre d'enregistrements lus ou écrits),
 - ◆ de la date et heure où s'est déroulée l'action.

- **les profils** : ce sont des structures qui caractérisent le comportement des sujets envers des objets, en termes de variables et modèles statistiques de l'activité observée.
- **les enregistrements d'anomalie** : ils sont générés lorsqu'une activité anormale est détectée,
- **les règles d'activités** : elles spécifient les actions à entreprendre lorsque certaines conditions sont satisfaites sur les enregistrements d'audit ou enregistrements d'anomalies générés. Ces règles, qui ont la forme classique conditions - actions, consistent à mettre à jour les profils, détecter les comportements anormaux, produire des rapports ou encore alerter l'officier de sécurité.

Dans ce modèle, un profil est défini par un ensemble de *variables* et de *modèles statistiques*. Ces variables représentent des mesures quantitatives accumulées durant une période de temps, qui peut être soit un intervalle de temps fixé (minute, heure, jour, etc.) ou entre deux événements d'audits (login et logout, connexion et déconnexion, etc.). Il existe trois types de variables :

- **un compteur d'événements** : qui représente le nombre d'enregistrements d'audit se produisant durant une période de temps (par exemple le nombre de mots de passe erronés durant une minute),
- **un compteur de temps** : par exemple le temps écoulé entre deux logins successifs,
- **un compteur de ressources** : qui représente le nombre de ressources utilisées durant une période de temps (par exemple nombre de pages imprimées par utilisateur et par jour)

En ce qui concerne les modèles statistiques, leur but est de définir à partir de n observations x_1, \dots, x_n sur une variable x , si la valeur x_{n+1} de l'observation $n+1$ est anormale. Les modèles proposés par Denning, sont :

- **le modèle opérationnel** : il compare la nouvelle valeur de x avec une limite fixe, par exemple, s'il y a dix mots de passe erronés en une minute, c'est une intrusion,
- **le modèle de déviation standard et moyen** : il utilise la moyenne et l'écart type des n observations pour définir un intervalle de confiance. Si la valeur x_{n+1} est en dehors de cet intervalle alors l'observation $n+1$ est anormale,
- **le modèle de covariances** : il est similaire au précédent mais il se base sur la corrélation de plusieurs variables pour tirer des conclusions, par exemple, le temps de CPU et les entrées/sorties utilisées par un programme,
- **le modèle de processus de Markov** : appliqué aux compteurs d'événements, il considère chaque type d'événement comme une variable d'état et utilise une matrice de transition d'état pour définir la probabilité de passage d'un état, défini par un enregistrement d'audit, à un autre état. Une nouvelle observation est

considérée comme anormale, si sa probabilité, définie par rapport à l'état précédent et à la matrice d'état, est trop faible.

- **le modèle de séries temporelles** : il considère qu'une observation $n+1$ est anormale si sa probabilité d'apparition, est trop faible.

Systemes experts

La différence majeure entre un système expert et un modèle statistique est que ce dernier utilise des formules statistiques pour identifier des comportements dans les données d'audit alors que le système expert utilise un ensemble de règles pour représenter ces comportements [Ilgun 1992] [Porras 1992]. L'inconvénient majeur d'un système expert est que sa base de règles n'est pas facile à créer, ni à maintenir.

2.4.4.1.2 Prédiction du comportement normal

Ces approches essayent de prédire les événements futurs en se basant sur des événements qui se sont déjà produits dans le passé [Sundaram 1996]. Elles se basent sur l'hypothèse que les séquences d'événements ne sont pas aléatoires. Si les données courantes ne sont pas en accord avec celles prédites alors une alarme est déclenchée. Les deux approches les plus importantes sont expliquées dans ce qui suit.

Génération de forme prédictive

Cette approche prédit les formes les plus probables en se basant sur les formes observées. Durant la phase d'apprentissage, cette approche détermine des règles temporelles qui caractérisent le comportement normal des utilisateurs. Un exemple d'une telle règle :

E1 - E2 -- (E3 = 95%, E4 = 5%)

Cette règle, basée sur les données observées auparavant, signifie que pour la forme E1 des événements observés, suivi de E2, la probabilité d'avoir ensuite E3 est de 95% alors qu'elle est de 5% pour E4.

Dans la phase de détection, une déviation est détectée si la séquence d'événements observée correspond à la partie gauche de la règle mais que les événements suivants dévient de manière significative de ceux prédits dans cette même règle [Kumar 1995]. Si l'on considère la règle précédente comme faisant partie d'un profil utilisateur normal alors la séquence E1 - E2 - E5 peut indiquer une intrusion.

Réseaux de neurones

Un réseau de neurones est constitué de plusieurs éléments de traitement simples, ressemblant à des neurones, appelés *unités* et qui interagissent en utilisant des connexions pondérées [Mounji 1997]. L'idée de cette approche est d'essayer de prédire le comportement des utilisateurs. Le réseau constitue le profil normal d'un utilisateur en apprenant les séquences des commandes habituellement utilisées par cet utilisateur. Ainsi, après chaque commande utilisée par cet utilisateur, il essaye de prédire la commande suivante, en tenant compte des n commandes antérieures. Si la commande réelle dévie de celle prédite alors une alarme est envoyée.

2.4.4.2 Détection basée sur l'approche non comportementale

L'approche non comportementale, dite aussi approche par scénario, consiste à détecter des intrusions ou des attaques exploitant des vulnérabilités systèmes *connues*. Elle se base sur le fait que toute attaque connue produit une trace spécifique dans les enregistrements d'audit. Elle nécessite donc une connaissance *a priori* des attaques à détecter. Une alarme est émise lorsque la trace d'une attaque connue est détectée dans les enregistrements d'audit.

L'approche par scénario nécessite plusieurs étapes :

- les scénarios d'attaques sont d'abord collectés,
- puis ces scénarios sont formalisés dans des modèles indépendants du système (tel que des règles ou des formes),
- ces modèles sont enregistrés dans une base de données,
- les traces d'audit ou informations réseaux sont ensuite formalisées pour pouvoir être comparées aux modèles enregistrés dans la base de données,
- si ces informations d'audit ou réseaux contiennent une séquence similaire à un modèle d'attaque, une alarme est émise.

Cette approche utilise plusieurs techniques que l'on peut regrouper dans trois classes :

- les approches à base de règle, c'est-à-dire les systèmes experts,
- les approches basées sur la signature,
- les algorithmes génétiques.

Une brève description de ces trois approches est présentée dans les paragraphes suivants.

2.4.4.2.1 Approches à base de règles : Systèmes experts

La détection des attaques connues se fait très souvent en utilisant des systèmes experts. La base de connaissances de ces systèmes experts contient des règles concernant non seulement les scénarios d'attaques que l'on veut détecter sur le système cible, mais aussi les vulnérabilités connues de ce système. La construction de cette base de règle repose entièrement sur l'expérience et le savoir-faire de l'officier de sécurité [Mé et Alanou 1996].

2.4.4.2.2 Approches basées sur la signature

Les approches basées sur la signature essayent de représenter les attaques sous forme de signatures. Les données courantes sont alors comparées à ces signatures. S'il y a des similitudes, une alarme est déclenchée.

Reconnaissance de formes

La reconnaissance de forme [Kumar and Spafford 1994a] [Kumar and Spafford 1994b] [Kumar and Spafford 1995] consiste à encoder les signatures d'intrusions connues en formes qui peuvent être reconnues dans les données d'audit. Dans son modèle, basé sur la notion d'événement, Kumar essaye d'identifier chaque nouvel événement à des formes représentant des scénarios d'intrusion. [Mé et Alanou 1996] fait une très bonne analogie, pour expliquer cette approche. Il considère, chaque événement d'un scénario d'attaque comme une lettre prise dans un alphabet, qui représente l'ensemble des événements auditable sur le système à surveiller. Il voit alors le fichier d'audit comme une chaîne de caractères principale, dans laquelle il faut localiser les scénarios d'attaque, qui sont vus comme des sous - suites de cette chaîne.

Analyse de transition d'état

Dans cette approche, le système surveillé est représenté comme un diagramme d'états [Sundaram 1996]. Les attaques sont alors représentées comme une séquence de transitions d'états de ce système [Kumar 1995]. Lorsque les données sont analysées, le système établit des transitions d'un état à un autre état. Cette approche modélise les intrusions comme des séries de changements d'états qui partent d'un état initial sûr à un état cible compromis [Porras 1995]. Si une séquence d'états spécifique est observée, une alarme est alors générée.

Surveillance de la frappe

Cette approche, très simple, analyse les frappes (sur clavier) d'un utilisateur pour identifier une attaque.

2.4.4.2.3 Algorithmes génétiques

Les algorithmes génétiques sont utilisés afin d'optimiser la recherche de scénarios d'attaques dans les fichiers d'audit [Mé 1995]. "*L'approche génétique, grâce à son bon équilibre exploration/exploitation, permet d'obtenir en un temps de traitement raisonnable, le sous-ensemble des attaques potentiellement présentes dans les traces d'audit*" [Mé et Alanou 1996].

2.4.4.3 Avantages et inconvénients des deux approches

Chacune des deux approches comportementale et par scénarios présente des avantages et des inconvénients.

2.4.4.3.1 Avantages de l'approche comportementale

Le principal avantage de l'approche comportementale est qu'elle est capable de détecter les attaques sans une connaissance *a priori* de ces attaques. Ainsi elle peut détecter des attaques inconnues.

2.4.4.3.2 Avantages de l'approche par scénarios

Le principal avantage de cette approche est qu'elle peut détecter les attaques qui se sont produites par le passé et qu'elle peut prendre en compte les comportements exacts des attaquants potentiels.

2.4.4.3.3 *Inconvénients de l'approche comportementale*

Les deux inconvénients majeurs de cette approche :

- définir un profil n'est pas une tâche facile. Il faut définir les bonnes variables à comparer entre le profil et les données d'audit ou données réseau. Durant une activité système normale, les valeurs des variables choisies doivent rester stables. S'il y a une attaque, elles doivent changer de manière significative ;
- décider qu'une activité suspicieuse doit déclencher une alarme n'est pas chose facile car il faut minimiser le nombre de fausses alarmes tout en essayant de limiter au maximum le nombre d'attaques non détectées voir même que ce nombre serait nul.

2.4.4.3.4 *Inconvénients de l'approche par scénarios*

Parmi les inconvénients de cette approche, nous en retiendrons trois :

- les attaques non répertoriées ne sont pas détectées ;
- il est difficile, voir impossible de formaliser un scénario d'attaque dans un modèle indépendant du système ;
- dans le cas des systèmes experts, la construction de la base de règle est très délicate car elle repose sur l'expérience de l'officier de sécurité, qui peut parfois être faible dans le domaine de la détection d'intrusions.

2.4.5 *Systèmes de détection d'intrusions existants*

Dans ce paragraphe, nous allons essayer de présenter brièvement quelques systèmes de détection d'intrusions, que nous avons choisi parmi tant d'autres [Mukherjee et al. 1994] [Price 1998].

2.4.5.1 **IDES et NIDES**

IDES (Intrusion-Detection Expert System) a été développé par SRI International. Il représente le modèle de référence pour un grand nombre de systèmes de détection d'intrusions. Il a été conçu pour surveiller un seul host et il traite uniquement les données d'audit fournis par les journaux d'audit.

IDES se veut indépendant du système surveillé. En effet, il fonctionne sur une machine dédiée, reliée au système par un réseau. Dans le but de détecter les violations de sécurité en temps réel, IDES s'appuie aussi bien sur une approche statistique que sur un système expert [Marshall 1991] [Porras 1992] [Mukherjee et al. 1994] [Samfat 1996] [White et al. 1996] [Price 1998]. Ainsi, il est constitué de deux éléments importants :

- **le détecteur d'anomalie** : qui est responsable de la détection des comportements atypiques (i.e. anormaux), en utilisant des méthodes statistiques du modèle de Denning ;

- **le système expert** : qui est chargé de détecter les attaques suspectes en s'appuyant sur une base de connaissances de scénarios d'attaques connus. En plus des règles concernant les intrusions passées, la base de connaissances de ce système expert contient également des règles concernant les vulnérabilités connues du système cible ainsi que des règles concernant la politique de sécurité d'un site spécifique [Mukherjee et al. 1994].

NIDES (Next- Generation IDES) [Lunt and Anderson 1993] [Mukherjee et al. 1994] [Sri 1997a] [Sri 1997b] [Price 1998], est une version améliorée de IDES. Il assure la détection d'intrusions sur plusieurs hosts (i.e distribuée) en se basant toujours sur les données d'audit. il n'y a aucune analyse du trafic réseau. Il utilise les mêmes algorithmes que IDES.

2.4.5.2 NADIR

NADIR (Network Anomaly Detection System) [Marshall 1991] [Porras 1992] [Mukherjee et al. 1994] [White et al. 1996] [Price 1998] est un système expert, qui a été conçu pour le réseau ICN (Integrated Computing Network) du Laboratoire National Los Alamos. Son but est d'analyser les activités réseaux des utilisateurs et de ICN en se basant sur les règles du système expert qui définissent la politique de sécurité et les comportements suspects. L'inconvénient majeur de ce système est qu'il ne peut être porté sur d'autres réseaux, étant donné que les protocoles réseaux d'ICN ne sont pas standards.

2.4.5.3 DIDS

DIDS (Distributed Intrusion Detection System) [Mukherjee et al. 1994] [Samfat 1996] [White et al. 1996] [Price 1998] a été conçu pour opérer sur un réseau distribué. Son architecture se compose de trois entités :

- le "Host Monitor" : Il collecte les données du host surveillé, fait une première analyse simple sur ces données puis transmet les informations pertinentes au "DIDS Director". Il en existe un par host ;
- le "LAN Monitor" : Il surveille le trafic sur le LAN, collecte les informations réseaux et reporte au "DIDS Director" les activités suspectes et non autorisées qui se sont produites sur le réseau. Il en existe un pour chaque segment LAN ;
- le "DIDS Director" : il analyse les rapports reçus du "LAN Monitor" et des "Host Monitor" afin de détecter les attaques potentielles.

La nature centralisée de DIDS peut se révéler être un inconvénient majeur dans le cas de réseaux WAN où les communications avec le "DIDS Director" risqueraient de submerger le réseau.

2.4.5.4 CSM

CSM (Cooperating Security Manager) [White et al. 1996] [Price 1998] est un système de détection d'intrusions, qui a été développé à l'université A&M du Texas, pour être utilisé dans un environnement de réseau distribué. Son principal objectif est

de détecter les activités intrusives, de façon non centralisée car utiliser un directeur central, qui coordonnerait toutes les activités, limiterait la taille du réseau. Pour cela, CSM doit s'exécuter sur chaque host connecté au réseau. Ainsi, au lieu de reporter les activités anormales à un directeur central, les CSMs communiquent entre eux pour détecter de manière coopérative les intrusions réseaux.

Le CSM est composé :

- d'un système de détection d'intrusions local (IDS) : qui, d'une part, assure la détection d'intrusions pour un host local et d'autre part est responsable de la détection proactive sur un autre host ;
- d'un gestionnaire de sécurité (SECMGR) : qui coordonne la détection d'intrusions distribuée entre les CSMs. La coordination entre les CSMs permet également de garder une trace des utilisateurs lorsqu'ils se déplacent d'un host à un autre host ;
- d'un gestionnaire d'intrus (IH : intruder handling component) : dont le rôle est d'entreprendre les actions qu'il faut lorsqu'un intrus est détecté ;
- d'une interface utilisateur graphique (GUI) : elle permet aux administrateurs et officiers de sécurité de communiquer avec les CSMs individuels afin de surveiller l'état du système ;
- d'un contrôleur de commande (CMNDMON : command monitor) : son but est d'intercepter les commandes exécutées par les utilisateurs et de les envoyer au IDS pour être analysées ;
- d'un gestionnaire de communication (TCPCOM : TCP communication handler) : il gère les communications TCP entre les différents CSMs.

La principale caractéristique de CSM est qu'il utilise une approche de détection d'intrusions *proactive*, au lieu de *réactive*. Dans une approche réactive centralisée, les hosts individuels reporteraient, par exemple, les occurrences de logins erronés au gestionnaire central. Alors que dans une approche proactive coopérative, le host à partir duquel l'intrus essaye de se connecter, contacterait les hosts cibles (i.e. ceux que l'intrus essaye d'attaquer). Cette approche proactive est possible car CSM s'exécute sur tous les hosts (ou du moins la plupart) connectés au réseau.

CSM peut fonctionner dans un environnement hétérogène car les messages échangés entre les CSMS sont indépendants du système. Cependant, CSM présente l'inconvénient de ne pas être facilement portable d'un système à un autre à cause du module gestionnaire d'intrus (IH) qui est très spécifique au système. Ceci dit, sa méthodologie peut être portée sur n'importe quel système.

2.4.5.5 GrIDS

GrIDS (Graph-Based Intrusion Detection System) [Staniford-Chen et al. 1996] [Staniford-Chen 1997] [Cheung et al. 1997] a été conçu pour détecter des attaques à grande échelle. Il peut fonctionner aussi bien sur des réseaux LAN que sur des réseaux plus grands.

GrIDS collecte les données concernant l'activité des hosts et le trafic réseau entre ces hosts. Toutes ces informations sont ensuite rassemblées dans des graphes d'activité qui révèlent la structure causale de l'activité réseau. Les nœuds d'un graphe d'activité correspondent aux hosts constituant le réseau, alors que les arêtes représentent l'activité réseau entre les différents hosts.

L'objectif de GrIDS est non seulement de détecter les attaques réseaux mais aussi les violations d'une politique d'accès réseau. En effet, GrIDS permet aussi à l'officier de sécurité de définir une politique de sécurité, spécifiant quels objets peuvent être accédés par quels sujets, puis de reporter les violations de sécurité.

En analysant les caractéristiques des graphes d'activité, GrIDS détecte et reporte les violations de la politique de sécurité établie par l'officier de sécurité. Durant la phase de détection, GrIDS analyse les caractéristiques des graphes d'activité et compare ces graphes à des formes intrusives connus. S'il y a des similitudes entre ces graphes et des attaques connues, il en informe l'officier de sécurité.

2.4.5.6 AAFID

Le système AAFID (Autonomous Agent for Intrusion Detection) traite le problème de la détection d'intrusions sous un autre angle [Crosbie and Spafford 1995a] [Crosbie and Spafford 1995b] [Crosbie and Spafford 1996] [Balasubramaniyan et al. 1998]. Au lieu de concevoir un système de détection d'intrusions monolithique, il propose une architecture distribuée, où plusieurs petits processus indépendants opèrent de manière coopérative pour assurer la surveillance du système cible. Ses principaux avantages sont : son efficacité, sa tolérance aux fautes, sa résistance aux dégradations et son extensibilité.

L'architecture de AAFID est composée de trois entités essentielles :

- les *agents*,
- les "*transceivers*",
- les "*monitors*".

Un *agent* est une entité d'exécution indépendante qui surveille certains aspects d'un host et reporte les comportements anormaux ou suspects au "*transceiver*" approprié. Sur un host, il peut y avoir un ou plusieurs agents. Les agents n'ont aucune autorité pour générer les alarmes et ne peuvent pas communiquer entre eux.

Les "*transceivers*" représentent l'interface de communication externe d'un host. Il y en a un pour chaque host surveillé. Un "*transceiver*" a deux rôles importants :

- ◆ un rôle de *contrôle* : il doit lancer et stopper les agents s'exécutant sur son host, garder une trace de ces agents et répondre aux commandes envoyées par ses "*monitors*" ;
- ◆ un rôle de *traitement de données* : il doit recevoir et analyser les rapports générés par les agents s'exécutant sur son host, puis envoyer les résultats de son analyse à un ou plusieurs "*monitors*".

Quant aux "*monitors*", ils représentent l'entité de plus haut niveau dans l'architecture de AAFID. Ils ont eux aussi un rôle de contrôle et un rôle de traitement de données similaires à ceux des "*transceivers*". La différence entre les "*monitors*" et les "*transceivers*" est qu'un "*monitor*" peut contrôler des entités qui s'exécutent sur différents hosts alors qu'un "*transceiver*" ne peut contrôler que les agents locaux. Dans leur rôle de contrôle, les "*monitors*" peuvent :

- ◆ recevoir des instructions d'autres "*monitors*",
- ◆ contrôler des "*transceivers*" et d'autres "*monitors*".

Par contre, pour ce qui est de la tâche de traitement de données, les "*monitors*" font une corrélation des informations reçues des "*transceivers*" qu'ils contrôlent. Ils détectent ainsi des événements qui impliquent différents hosts. De plus, ils peuvent communiquer avec une interface utilisateur et fournir ainsi un point d'accès au système AAFID.

2.4.5.7 Autres systèmes et outils de détection d'intrusions

En plus des cinq outils que nous venons d'étudier, de nombreux autres outils ont été développés. [Mukherjee et al. 1994], [Mé et Alanou 1996] et [Price 1998] présentent une courte description de différents outils. Voici une liste, non exhaustive, de certains de ces outils :

- ASAX [Mé et Alanou 1996] [Habra et al. 1992] [Habra et al. 1994] [Mounji 1997] [Price 1998],
- ComputerWatch (Computer Watch Audit Trail Analysis Tool) de AT&T [Marshall 1991] [Mukherjee et al. 1994] [Price 1998],
- Discovery développé par TRW [Marshall 1991] [Mukherjee et al. 1994] [Price 1998],
- EMERALD (Event Monitoring Enabling Response to Anomalous Live Disturbances) [Price 1998] [Porras 1998] développé par SRI International,
- GASSATA [Mé et Alanou 1996] [Price 1998],
- HAYSTACK , développé par les laboratoires Haystack pour le compte du Los Alamos National Laboratory [Marshall 1991] [Mukherjee et al. 1994] [Samfat 1996] [Mounji 1997] [Price 1998],
- Hyperview [Mé et Alanou 1996] [Price 1998],
- IDIOT [Crosbie et al. 1996] [Price 1998],
- ISOA (Information Security Officer's Assistant) [Marshall 1991] [Porras 1992] [Mukherjee et al. 1994],

- MIDAS (Multics Intrusion Detection and Alerting System) [Marshall 1991] [Porras 1992] [Mukherjee et al. 1994] [Mounji 1997] [Price 1998],
- NSM a été conçu à l'université de Californie-Davis [Marshall 1991] [Mukherjee et al. 1994] [Price 1998],
- USTAT [Ilgun 1992] [Porras 1992] [Kemmerer 1997].

2.4.6 *Caractéristiques d'un bon système de détection d'intrusions*

Les systèmes de détection d'intrusions existants ne sont pas parfaits. Si nous reprenons par exemple, DIDS, sa nature centralisée représente un désavantage majeur, dans le cas de réseaux WAN où les communications avec l'entité gestionnaire, peuvent congestionner le réseau. Quant à CSM, bien qu'il ait été développé pour un environnement distribué, il ne peut pas être facilement portable vers un autre environnement. En règle générale, les systèmes de détection d'intrusions existants sont mal adaptés à la complexité croissante des réseaux et des attaques auxquels ils sont sujets. Traditionnellement, ils utilisent des méthodes basées sur des modèles de système expert, de modèles statistiques, réseaux de neurones, etc. Ces systèmes sont généralement développés pour des réseaux et systèmes bien définis et ne sont pas adaptés à des environnements dynamiques. En effet, les paramètres des modèles utilisés sont prédéfinis. Ainsi, si une nouvelle attaque doit être détectée, il est très difficile de modifier le système de détection d'intrusions. Globalement, l'architecture des systèmes existants est monolithique. Dans cette architecture, l'analyse des données, collectées par une ou plusieurs entités distribuées, n'est effectuée que par un seul module (DIDS). Cette approche présente deux inconvénients majeurs :

1. d'une part, elle présente un point de rupture, dans le cas où l'entité centrale serait attaquée ;
2. d'autre part le déploiement de ce type de système à grande échelle est limité.

Dans d'autres systèmes, tel que CSM, l'analyse des données est effectuée sans l'utilisation d'une entité centralisée ce qui résout les problèmes engendrés par l'approche monolithique. Cependant, il existe encore certains inconvénients tel que :

1. la difficulté de s'adapter aux changements qui peuvent se produire dans le réseau et aux comportements des utilisateurs qui varient considérablement ;
2. la difficulté de mise à jour de ces systèmes, lorsque l'on veut améliorer ou rajouter de nouvelles méthodes de détection.

Pour [Crosbie and Spafford 1995a] [Crosbie and Spafford 1995b] [Crosbie and Spafford 1996] [Balasubramaniyan et al. 1998] [Price 1998], un bon système de détection d'intrusions doit présenter les caractéristiques suivantes :

- ***fonctionnement continu*** du système et ***visibilité du fonctionnement*** : le système doit s'exécuter continuellement sans aucune supervision humaine. Cependant il ne doit pas être une "boîte noire", son fonctionnement interne doit être examinable de l'extérieur ;

- **tolérance aux fautes** : le système doit survivre à un "crash" système sans qu'il ait à reconstruire sa base de connaissances lors du redémarrage ;
- **résistance aux subversions** : le système doit s'auto-surveiller afin de détecter dès qu'il y a un attaquant qui essaye de le corrompre ;
- **minimisation de la surcharge système** : un système de détection d'intrusions qui ralentit trop le fonctionnement du système surveillé serait difficilement utilisable ;
- **observation des déviations** du comportement normal ;
- **adaptabilité** : il doit facilement s'ajuster au système cible ;
- **reconfiguration dynamique** : il doit être capable de s'adapter aux changements qui peuvent se produire au niveau du système à surveiller (par exemple l'ajout de nouvelles applications) ;
- **minimisation des erreurs fausses positives, fausses négatives et de subversion** : une erreur **fausse positive** survient lorsque le système classifie une action comme une anomalie (une intrusion possible) alors que c'est une action légitime. Une **fausse négative** se produit lorsqu'une action intrusive se produit mais que le système la laisse passer comme un comportement non intrusif. Ces erreurs sont bien entendu plus sérieuses que les fausses positives puisque cela porte atteinte à la sécurité du système.
Une **erreur de subversion** se produit lorsque qu'un intrus modifie des opérations du détecteur d'intrusions en forçant les fausses négatives à se produire.
Les occurrences de ces types d'erreurs doivent être minimisées au maximum.

Quant à [Mounji 1997], il identifie quatre besoins fondamentaux auxquels doit répondre un système de détection d'intrusions :

- **puissance** : dûe à la complexité des scénarios d'attaques, le système doit utiliser un modèle de représentation et de détection des attaques assez puissant ;
- **réutilisabilité** : il doit pouvoir être réutilisé dans des environnements différents avec un minimum d'effort d'adaptation ;
- **efficacité** : il doit non seulement être capable de détecter les scénarios d'attaques les plus complexes lorsqu'ils se manifestent dans les traces d'audit mais aussi de pouvoir le faire en temps réel ;
- **adaptabilité** : le système doit facilement s'adapter à des politiques de sécurité spécifiques. En d'autres termes, l'officier de sécurité doit pouvoir d'une part activer et désactiver une ou plusieurs règles en fonction des politiques de sécurité en vigueur ; et d'autre part, il doit avoir la possibilité de rajouter de nouvelles règles de détection.

En ce qui nous concerne, sans ignorer les caractéristiques précédentes, nous nous sommes aussi posé la question, de savoir quelles seraient les propriétés que devrait avoir un bon système de détection d'intrusions. Nous pensons que pour qu'un système de détection d'intrusions soit *efficace*, il est très important de :

- **distribuer** les fonctions de détection à plusieurs entités *autonomes*, qui surveillent différents points du réseau afin de détecter les attaques réseaux se caractérisant très souvent par des comportements sur plusieurs éléments différents distribués dans le réseau ;
- de plus, ces entités doivent pouvoir se *communiquer* leurs analyses et *coopérer* afin de détecter efficacement les attaques coordonnées ;
- **réagir** rapidement lorsqu'une attaque se produit afin de limiter les dommages qui peuvent être causés ;
- enfin, un système de détection d'intrusions doit pouvoir *s'adapter* aux changements se produisant dans le réseau, notamment lors de l'application de nouvelles politiques de sécurité.

Nous reviendrons plus en détail sur ces propriétés et leur justification dans le Chapitre 3.

2.5 Conclusion

Nous avons vu, que la majorité des systèmes de détection d'intrusions existants [Price 1998] sont basés sur des techniques d'intelligence artificielle. Cependant, ces systèmes sont généralement développés pour des environnements bien définis et n'offrent pas une solution à certaines caractéristiques des réseaux telles que la variation des comportements utilisateurs et des services offerts, la complexité et l'évolution croissante des types d'attaques auxquels ils peuvent être sujets, la rapidité des attaques qui peuvent survenir simultanément sur plusieurs machines, etc. La gestion de la sécurité en général et la détection d'intrusions en particulier s'avère donc être un domaine assez complexe. Pour essayer de simplifier cela, nous nous sommes intéressés aux technologies actuellement émergentes qui s'orientent de plus en plus vers des approches agents intelligents. Ces technologies nous laissent, a priori un bon espoir pour apporter une solution au problème de la gestion de la sécurité. De nombreux travaux ont été menés et ont prouvé qu'une approche agents intelligents et système multi-agents (SMA) était bien adaptée pour gérer des problèmes complexes, particulièrement dans le domaine de la gestion de réseau [Oliveira 1998] [Alagha and Labiod 1999]. Dans notre cas, nous pensons qu'une solution SMA a une forte probabilité d'être une excellente solution pour répondre à cela. Nous proposons donc d'étendre l'application de cette technologie au domaine de la gestion de la sécurité et spécialement la détection d'intrusions. Cela nous a amené à investiguer ce domaine que nous décrivons dans le chapitre suivant.

Chapitre 3 Une approche multi-agents pour la gestion de sécurité

3.1 Introduction

Nous avons introduit (voir Chapitre 2) que les SMAs pouvaient apporter une solution à la complexité du problème de gestion de sécurité. Nous verrons que certaines des caractéristiques requises pour les entités de réseau (distribution, autonomie, communication, coordination, adaptabilité, réactivité) sont importantes et même nécessaires pour répondre aux nouveaux besoins de gestion de sécurité. Nous pensons et nous le prouverons (voir 3.4), qu'il existe une similitude entre ces caractéristiques et les propriétés des systèmes multi-agents. Une nouvelle solution basée sur des techniques de SMAs semble ainsi avoir une forte probabilité de résoudre notre problème de gestion de sécurité.

Comme nous l'avons déjà dit dans le Chapitre 2, l'apport d'une solution intelligente pour la résolution de problèmes complexes, a été prouvé à plusieurs reprises [Rao et al 1993] [Rao and Georgeff 1995], en particulier dans le domaine de la gestion de réseaux [Oliveira 1998] [Alagha and Labiod 1999] [Conti 2000]. Dans ce chapitre, nous allons essayer de montrer que la nature intelligente et distribuée des SMAs peut répondre aux nouveaux besoins de sécurité.

Dans ce chapitre, nous décrivons brièvement la technique de systèmes multi-agents. Puis nous discutons de l'évolution des besoins de sécurité et de l'importance de certaines caractéristiques pour :

- répondre aux nouveaux besoins de sécurité,
- assurer une détection d'intrusions efficace.

Enfin nous présenterons les grandes lignes de notre modèle de gestion de sécurité.

3.2 Les systèmes multi-agents et la technologie agent intelligent

Dans cette section, nous allons donc investiguer ce domaine en commençant par définir les concepts d'agents intelligents et systèmes multi-agents. Puis nous décrirons leurs caractéristiques. Ensuite, nous présenterons les différents modèles d'agents

existants. Enfin, nous présenterons les méthodologies existantes de conception de SMAs, sur lesquelles nous nous baserons pour concevoir notre SMA.

3.2.1 Concepts de base

Le concept *agent* a été influencé par une large variété de disciplines et d'expériences. Cependant, les origines de ce terme proviennent de l'intelligence artificielle (IA) et plus particulièrement de l'intelligence artificielle distribuée (IAD) où il est utilisé depuis les années quatre vingt pour exprimer l'idée "d'objets qui pensent" [Magendaz 1995]. L'IAD est née de la difficulté d'intégrer dans une même base de connaissances, l'expertise, les compétences et la connaissance de différentes entités qui communiquent et collaborent pour réaliser un but commun [Erceau et Ferber 1991]. L'IAD consiste à distribuer l'expertise au sein d'une société d'entités, appelées *agents* dont le contrôle et les données sont distribués [Guessoum 1996]. Ces agents, qui sont relativement indépendants et autonomes interagissent dans des modes simples ou complexes de coopération pour accomplir un objectif global, notamment la résolution de problèmes complexes.

[Skarmas 1998] définit, l'IAD comme étant un domaine concerné par les systèmes ouverts et distribués dont les entités présentent une sorte d'intelligence et qui essaient d'accomplir des buts qui peuvent être implicites ou explicites. L'IAD a donné naissance à deux principaux domaines :

- la **résolution des problèmes distribués** (DPS), qui s'intéresse à la décomposition d'un problème complexe en sous-problèmes et à sa résolution par des entités distribuées logiques et physiques ;
- les **systèmes multi-agents** (SMA) qui sont concernés par la coordination du comportement des agents, qui peuvent être vus comme des entités intelligentes et autonomes [Skarmas 1998].

L'IAD était supposée apporter une solution à des problèmes spécifiques tels que [Labidi et Lejouad 1993] [Oliveira 1998] :

- la modélisation et distribution de la connaissance parmi plusieurs agents ;
- la génération de plans d'actions où la présence d'autres agents doit être considérée ;
- la résolution de conflits entre agents et la maintenance de la cohérence des décisions et plans d'actions ;
- la résolution des problèmes de communication pour permettre les interactions entre agents ;
- la résolution des problèmes spécifiques concernant l'organisation des SMAs.

Par ailleurs, le terme *agent* est aussi utilisé, depuis plusieurs années, dans le domaine de l'informatique distribuée pour décrire des entités spécifiques (client/serveur) dédiées à la résolution de tâches spécifiques tel que, Agents Utilisateurs, Agents de

Transfert de Messages et Agents de Systèmes d'Annuaire, dans les normes X.400 et X.500. Le concept *agent* est beaucoup utilisé dans le domaine de la gestion de réseau, où l'on parle d'Agents SNMP et d'Agents CMIP, qui sont des modules logiciels collectant et enregistrant des informations de gestion concernant un ou plusieurs éléments réseaux. Ces agents communiquent ensuite ces informations à la demande d'une entité supérieure (manager) en utilisant les protocoles SNMP et CMIP [Cheikhrouhou et al. 1998].

Etant donné les origines diverses du concept agent, nous ne pouvons pas donner une seule définition au terme *agent*. En effet, plusieurs définitions ont été proposées par différents auteurs pour clarifier ce concept. Ces définitions sont citées dans la section suivante.

3.2.1.1 Définitions

Nous avons vu dans la section précédente que le terme *agent* a été introduit dans différents domaines, ce qui ne facilite pas sa définition. Bien au contraire, jusqu'à présent, il n'existe pas une définition universelle d'un agent et encore moins d'un agent intelligent. A ce propos, Carl Hewitt fit remarquer (lors du troisième international workshop sur l'IAD), que la question *qu'est ce qu'un agent ?* est aussi embarrassante pour la communauté informatique que la question *qu'est ce que l'intelligence ?* est embarrassante pour la communauté d'intelligence artificielle. Cependant, différentes définitions ont été données à ce terme :

Pour N.Skarmas [Skarmas 1998], les agents peuvent être vus comme des unités intelligentes et autonomes.

J. Ferber [Ferber 1995] [Ferber and Ghallab 1998] définit un agent comme une entité physique ou virtuelle :

- qui est capable d'agir dans son environnement ;
- qui peut communiquer avec d'autres agents ;
- qui est capable de percevoir son environnement dont elle ne dispose qu'une représentation partielle, voir même aucune représentation ;
- qui possède des compétences, offre des services et peut éventuellement se reproduire ;
- qui poursuit un objectif individuel ;
- dont le comportement tend à satisfaire ses objectifs en tenant compte de ses compétences, de sa perception, de ses représentations et des communications qu'elle reçoit.

Pour Y.Demazeau et J-P. Müller [Demazeau and Müller 1990] [Demazeau and Müller 1991], le terme agent représente une entité intelligente qui agit de manière rationnelle et intentionnelle en fonction de ses buts et de l'état courant de ses connaissances.

Quant à Shoham [Shoham 1993], Cohen et Levesque [Cohen and Levesque 1988], Wooldridge et Jennings [Wooldridge and Jennings 1995], ils définissent un agent comme une entité dotée d'un *état mental*, qui représente ses connaissances, croyances, intentions et engagements vis-à-vis de lui-même et des autres agents. Nous reviendrons sur cette notion d'*état mental*, dans le paragraphe 3.2.2.1.1.

Pour revenir à la définition d'un SMA, il peut être défini comme une société d'agents qui interagissent pour satisfaire un ensemble de buts [Ferber 1996].

Bien que nous n'ayons pas pu donner une seule définition du concept agent, nous allons, dans les deux paragraphes suivants, présenter les propriétés qui caractérisent un agent et un SMA.

3.2.1.2 Propriétés d'un agent intelligent

Plusieurs propriétés caractérisent un agent, cependant, dans cette section, nous avons sélectionné celles qui nous paraissent appropriées pour répondre aux besoins de gestion de la sécurité.

3.2.1.2.1 Autonomie

Wooldridge et Jennings [Wooldridge and Jennings 1995] définissent l'autonomie comme étant la capacité pour un agent d'opérer de manière autonome sans une intervention directe d'humains ou d'autres agents et de contrôler ses actions et son état interne. Quant à Castelfranchi [Castelfranchi 1995], il définit un agent autonome comme étant un agent qui a ses propres buts et qui est capable de décider des buts à poursuivre, comment les atteindre et résoudre les conflits internes relatives aux buts choisis. Pour Cohen [Cohen and Levesque 1988], Shoham [Shoham 1993], un agent est capable :

- d'agir selon ses intentions ;
- d'adopter les buts qu'il croit réalisables et d'abandonner ceux qu'il croît irréalisables ;
- de planifier ses propres actions et de tenir compte de celles des autres ;
- de raisonner non seulement sur ses connaissances mais aussi sur celles des autres.

3.2.1.2.2 Réactivité

Les agents perçoivent leur environnement et réagissent aux changements qui s'y produisent [Wooldridge and Jennings 1995]. La réactivité signifie aussi la capacité qu'a un agent de modifier son comportement lorsque les conditions environnementales changent [Oliveira 1998].

3.2.1.2.3 Proactivité

Les agents n'agissent pas simplement en réponse à leur environnement, ils sont capables d'exhiber un comportement guidé par des buts en *prenant des initiatives* [Wooldridge and Jennings 1995]. La proactivité est la capacité d'un agent d'anticiper des situations et de changer son cours d'action pour les éviter [Oliveira 1998].

3.2.1.2.4 Adaptabilité

L'adaptabilité est la capacité d'un agent de s'adapter à l'environnement dans lequel il est situé [Oliveira 1998]. Un agent adaptatif est un agent capable de contrôler ses aptitudes (communicationnelles et comportementales) selon l'environnement dans lequel il évolue et selon l'agent avec lequel il interagit [Guessoum 1996].

L'adaptabilité est une propriété très importante pour un agent qui évolue dans un environnement dynamique.

3.2.1.2.5 Sociabilité

La sociabilité est la capacité d'un agent de s'intégrer dans un environnement peuplé d'agents avec qui il échange des messages pour accomplir un but [Oliveira 1998]. Nous reviendrons sur cette propriété lorsque nous aborderons les propriétés d'un SMA car elle intègre d'autres propriétés tel que la communication, la coopération et la délégation.

3.2.1.2.6 Apprentissage

L'apprentissage est une propriété assez particulière car un agent n'est pas forcément une entité capable d'apprendre. Les agents peuvent avoir à apprendre lorsqu'ils réagissent et/ou interagissent avec leur environnement externe [Nwana 1996]. L'apprentissage est une propriété qui fournit aux systèmes la capacité d'acquérir la compréhension de certains comportements au cours du temps, sans nécessiter que ces comportements soient programmés manuellement [Mitchell 1992].

Un agent a la faculté d'apprendre s'il est capable d'utiliser de nouvelles connaissances pour modifier son comportement [Cheikhrouhou et al. 1998].

Bien que l'apprentissage soit un attribut clé de l'intelligence [Nwana 1996], il existe très peu d'agents qui ont cette capacité.

3.2.1.2.7 Sécurité

La sécurité est une propriété importante, notamment dans le contexte de ce travail, car elle permet de garantir que lorsque l'on interagit avec un agent, que cet agent n'a pas été corrompu par un virus, par de fausses croyances ou par des connaissances qui n'ont pas de sens [Oliveira 1998].

3.2.1.3 Propriétés des systèmes multi-agents

Nous avons abordé la *sociabilité* dans la section précédente mais en réalité cette propriété est centrée sur les interactions entre agents [Guessoum 1996]. Nous allons donc voir maintenant les différentes propriétés qui sont impliquées dans la sociabilité.

3.2.1.3.1 Coopération

Pour Ferber [Ferber 1995], pour que plusieurs agents soient dans une situation de coopération, il faut que l'une des deux conditions suivantes soit vérifiée :

1. l'ajout d'un nouvel agent permet d'accroître différenciellement les performances du groupe ;
2. l'action des agents sert à éviter ou à résoudre des conflits potentiels ou actuels.

La coopération entre agents peut être [Oliveira 1998] :

- soit *implicite* et dans ce cas les agents ont un but commun implicite qu'ils doivent atteindre en exécutant des actions indépendantes. Dans ce type de coopération, la communication entre les agents est facultative ;
- soit *explicite* et dans ce cas, les agents exécutent des actions qui leur permettent d'achever non seulement leurs propres buts mais aussi les buts des autres agents. Ce type de coopération nécessite une communication entre les agents pour prendre connaissance des buts des autres agents.

3.2.1.3.2 *Coordination*

Dans un système multi-agents, la coordination des actions des différents agents permet d'assurer une cohérence du système [Guessoum 1996]. Il existe plusieurs mécanismes de coordinations parmi lesquels, nous retrouvons : l'organisation, la planification et la synchronisation [Guessoum 1996] [Oliveira 1998]. Nous définissons ces trois mécanismes.

L'organisation

Une organisation représente un groupe d'agents qui travaillent ensemble afin de réaliser une ou plusieurs tâches [Guessoum 1996]. De nombreux travaux sur l'organisation ont été proposés, néanmoins T. Bourron [Bourron 1992] les a regroupés en deux classes où l'organisation est définie :

- soit comme une structure externe aux agents et elle est représentée par un objet ou un agent,
- soit comme un objet abstrait dont la représentation est distribuée parmi les membres de l'organisation [Guessoum 1996].

Parmi les travaux existants, nous pouvons citer le *réseau contractuel* introduit par R. Smith [Smith 1980] [Smith and Davis 1981]. Le *réseau contractuel* est un mécanisme d'allocation de tâches fondé sur la notion d'appel d'offre [Ferber 1995]. Dans ce modèle, un agent peut avoir deux rôles par rapport à une tâche [Guessoum 1996] : *manager* ou *contractant*. L'appel d'offre s'effectue en quatre étapes :

- ***appel d'offre*** : le *manager* décompose une tâche en sous-tâches. Il recherche ensuite des *contractants* pour les réaliser en faisant une annonce de tâches à tous les agents du système ;
- ***envoi de propositions*** : les agents élaborent une proposition et font une offre au *manager* ;
- ***attribution du marché*** : le *manager* évalue les propositions, attribue la tâche à un ou plusieurs agents et informe les autres agents de son choix ;
- ***établissement du contrat*** : l'agent ou les agents sélectionnés deviennent *contractants* et informent le *manager* qu'ils s'engagent à réaliser la tâche.

La planification

La *planification multi-agents* est une autre approche de coordination dans les systèmes à base d'agents. Afin d'éviter des actions conflictuelles ou inconsistantes, les agents construisent un plan multi-agents qui détaille toutes les actions futures et les interactions nécessaires pour atteindre leurs buts [Oliveira 1998].

La planification dans les SMAs, se décompose en trois étapes [Ferber 1995] :

- la construction de plans,
- la synchronisation et coordination des plans,
- l'exécution de ces plans.

Il existe deux types de planification, dans les SMAs :

- la *planification centralisée* pour agents multiples qui suppose l'existence d'une vue globale du plan [Guessoum 1996]. Dans cette approche, il existe un seul agent capable de planifier et d'organiser les actions pour l'ensemble des agents [Ferber 1995] ;
- la *planification distribuée* où chaque agent planifie individuellement ses actions en fonction de ses propres buts [Ferber 1995]. Les différents agents se communiquent ensuite leurs plans partiels afin de détecter et d'éviter les conflits éventuels.

La synchronisation

La synchronisation est le "bas niveau" de la coordination où sont implémentés les mécanismes de base permettant aux différentes actions de s'articuler correctement [Ferber 1995]. Elle permet de synchroniser l'enchaînement des actions des différents agents. J. Ferber [Ferber 1995] répertorie deux types de synchronisation :

- la *synchronisation par mouvement* utilisée lorsque plusieurs éléments doivent se déplacer ensemble. Dans ce type de synchronisation, il s'agit de coordonner le rythme et le positionnement dans le temps d'actions en fonction des événements qui se produisent ;
- la *synchronisation d'accès à une ressource* utilisée lorsque plusieurs agents doivent partager une ressource.

3.2.1.3.3 Délégation

La délégation est la capacité d'un agent à exécuter des tâches pour le compte d'un tiers. Elle permet ainsi à un agent, qui ne peut pas atteindre ses buts par manque de ressources, de compétences ou dans le but d'alléger sa tâche, de demander à d'autres agents d'achever des buts pour son compte. Cette propriété est cruciale lorsque la coordination est supportée par une structure organisationnelle [Oliveira 1998] et que l'environnement du SMA varie et nécessite par conséquent une redistribution et mise à jour des tâches que doivent accomplir les différents agents (ce qui se traduit par une délégation de nouveaux buts). Dans une organisation hiérarchique par exemple,

lorsque le SMA doit atteindre de nouveaux objectifs, elle permet à une entité gestionnaire de déléguer des sous-tâches à des agents sous-jacents et de les modifier. La délégation est beaucoup utilisée dans la gestion de réseau où elle permet une flexibilité du système qui l'utilise [Magendaz 1995].

3.2.1.3.4 *Communication*

Dans un SMA, les agents communiquent entre eux en s'échangeant des informations via un *langage de communication agent* [Wooldridge and Jennings 1995]. Le langage, le plus utilisé aujourd'hui est KQML (Knowledge Query and Manipulation Language) qui est un langage de haut niveau utilisant une liste de types de messages, appelés performatifs [Ferber 1995]. Les agents peuvent également communiquer en utilisant d'autres mécanismes tel que le mécanisme de tableau noir [Guessoum 1996].

3.2.2 *Les différents modèles d'agents*

Les SMAs existants sont généralement décomposés en deux classes principales :

- les systèmes *délibératifs*, dits aussi *cognitifs*,
- les systèmes *réactifs*.

Cependant, dans la littérature, sont répertoriés quatre modèles d'agents : les *agents délibératifs*, les *agents réactifs*, les *agents interactifs* et les *agents hybrides*. Dans cette section, nous allons présenter brièvement les différents types de modèles d'agents.

3.2.2.1 *Les agents délibératifs*

Les *agents délibératifs* ont la capacité de résoudre des problèmes complexes. Ils sont ainsi capables de raisonner sur une base de connaissances, de traiter des informations diverses liées au domaine d'application et des informations relatives à la gestion des interactions avec d'autres agents et avec l'environnement [Guessoum 1996]. Ces agents maintiennent une représentation interne de leur monde et un état mental explicite qui peut être modifié par un raisonnement symbolique [Müller 1996]. Les SMAs délibératifs ont deux problèmes majeurs [Guessoum 1996] :

- la traduction de l'univers de l'agent en une description symbolique ;
- la représentation symbolique des informations de l'univers complexe des entités et des processus ainsi que la manière de raisonner sur ces informations.

L'architecture d'agents délibératifs la plus importante est l'architecture BDI (Belief, Desire, Intention) que nous décrivons dans le paragraphe suivant.

3.2.2.1.1 *Le modèle BDI*

L'idée de base de l'approche BDI est de décrire l'état interne d'un agent en termes d'*attitudes mentales* et de définir une architecture de contrôle grâce à laquelle l'agent peut sélectionner le cours d'action de ses attitudes mentales. Il a été défini trois attitudes mentales de base qui sont : les *croyances* (beliefs) les *désirs* (desires) et les *intentions* (intentions). Dans des approches BDI plus pratiques tel que IRMA par

exemple [Müller 1996], il a été montré que ces trois attitudes mentales n'étaient pas suffisantes, une extension a alors été proposée en rajoutant la notion de *buts* (goals) et de *plans* (plans). Nous allons maintenant définir de manière informelle ces différents concepts :

- les ***croyances*** décrivent l'état de l'environnement du point de vue d'un agent [Ferber 1995]. Elles expriment ce que l'agent croit sur l'état courant de son environnement [Müller 1996] ;
- les ***désirs*** sont une notion abstraite qui spécifie les préférences sur l'état futur de l'environnement d'un agent. Une caractéristique importante des *désirs* est qu'un agent peut avoir des *désirs* inconsistants et qu'il n'a donc pas à croire que ses *désirs* sont réalisables [Müller 1996] ;
- les ***buts*** représentent les engagements d'un agent pour atteindre un ensemble d'états de l'environnement [Müller 1996]. A partir de la définition précédente, on peut dire qu'un *désir* est une étape dans le processus de création d'un *but*. Si un *désir* d'un agent est poursuivi de manière consistante, il devient l'un des *buts* qui indiquent les options de gestion qu'a un agent [Oliveira 1998]. Cependant, il n'y a, à ce moment là, aucun engagement pour l'exécution de cours d'actions spécifiques. La notion d'engagement d'atteindre un *but* décrit la transition des *buts* aux *intentions*. De plus, il est nécessaire, que l'agent croit que ses *buts* peuvent être atteints [Müller 1996] ;
- les ***intentions*** représentent les actions que l'agent s'engage à exécuter. A partir du moment où les agents sont limités par leurs ressources, ils peuvent leur arriver de ne pas pouvoir poursuivre tous leurs *buts*. Même si l'ensemble des *buts* créés est consistant, il est nécessaire que l'agent choisisse un certain nombre de *buts* pour lesquels il s'engage. C'est ce processus qui est appelé la formation des *intentions*. Ainsi, les *intentions* courantes d'un agent sont décrites par un ensemble de *buts* sélectionnés avec leur état de traitement [Müller 1996] ;
- les ***plans*** jouent un rôle très important pour une implémentation pragmatique des *intentions*. Les *intentions* représentent des *plans* partiels d'actions que l'agent s'engage à exécuter pour atteindre ses buts. Par conséquent, il est possible de structurer les intentions en plans plus étendus, et de définir les intentions d'un agent comme les *plans* qui sont couramment adoptés [Müller 1996].

Dans le paragraphe suivant, nous allons voir le formalisme de cette théorie BDI.

3.2.2.1.2 Sémantique du modèle formel BDI

Hintikka [Hintikka 1962] considère que les *croyances* peuvent être modélisées par la sémantique des *mondes possibles* où un ensemble d'états d'environnement possibles est associé à chaque situation. Rao et Georgeff [Rao and Georgeff 1991a] [Rao and Georgeff 1991b] [Rao and Georgeff 1992] [Rao and Georgeff 1995] ont cette sémantique où ils considèrent que les *buts* et *intentions* ont également une sémantique de mondes possibles. Dans chaque situation, il y a un ensemble de *mondes accessibles de croyances, de buts et d'intentions* (belief-accessible world, goal-accessible world, intention-accessible world) qui caractérisent les mondes (états d'environnement) que

l'agent croît être possible, les buts à atteindre et les buts qu'il s'engage à atteindre. Les mondes possibles multiples résultent du manque de connaissances de l'agent en ce qui concerne l'état du monde. Ces mondes sont modélisés par des relations d'accessibilité pour les croyances **B**, les buts **G** et les intentions **I**, qui permettent le passage entre les mondes. Cohen et Levesque [Cohen and Levesque 1990] traitent chaque monde possible comme une ligne temporelle représentant une séquence d'événements qui s'étend temporellement entre le passé et le futur. Quant à Rao et Georgeff [Rao and Georgeff 1991a] [Rao and Georgeff 1991b] [Rao and Georgeff 1992] [Rao and Georgeff 1995], ils considèrent chaque monde possible comme un arbre temporel, qui dénote les cours d'actions (ou d'événements) que l'agent peut choisir dans un monde particulier. L'agent passe d'un *monde accessible de croyances* à un *monde accessible de buts* en désirant les chemins futurs et d'un *monde accessible de buts* à un *monde accessible d'intentions* en s'engageant pour certains chemins futurs. Le passage entre les mondes revient à éliminer successivement les branches de l'arbre temporel. Sémantiquement, cela nécessite un *sous-monde accessible de buts* pour chaque *monde accessible de croyances* et de même un *sous-monde accessible d'intentions* pour un *monde accessible de buts*. Cette sémantique de monde possible a été également reprise par Halpern et Moses 1992 [Halpern and Moses 1992] et Wooldridge [Wooldridge 1992].

3.2.2.2 Les agents réactifs

L'idée d'architectures d'*agents réactifs* a été essentiellement introduite par Brooks [Brooks 1991], dans les années 80. Les architectures *réactives*, dites *aussi comportementales*, se caractérisent par des agents qui ont la capacité de réagir rapidement à des problèmes simples, qui ne nécessitent pas un haut niveau de raisonnement. En effet, leurs décisions sont essentiellement basées sur un nombre très limité d'informations et sur des règles simples de type *situation – action* [Müller 1996]. Ce type d'architectures ne nécessite aucune représentation symbolique de l'environnement réel. En fait, les agents réactifs réagissent aux changements de leur environnement ou aux messages provenant des autres agents. Ces agents ne disposent que d'un protocole de communication et d'un langage de communication réduits [Guessoum 1996]. Dans un système à base d'agents réactifs, l'intelligence émerge de l'interaction des agents avec leur environnement.

L'inconvénient majeur de ce type d'architectures est l'absence de formalisme, ce qui ne facilite pas la compréhension et la prédiction du comportement des agents, et la quasi-impossibilité de vérifier leur cours d'action [Oliveira 1998].

3.2.2.3 Les agents interactifs

Les architectures d'*agents interactifs* se caractérisent par des agents conçus spécialement avec des capacités de coordination et de coopération. Ce type d'architectures n'est pas suffisant pour construire des agents [Oliveira 1998]. Les principaux mécanismes utilisés dans ces architectures sont la *communication*, la *résolution des problèmes distribués (DPS)* et la *planification multi-agents* [Müller 1996].

3.2.2.4 Les agents hybrides

Les architectures précédentes présentent certaines faiblesses :

- les architectures purement réactives ont un comportement assez simpliste,
- alors que les architectures délibératives utilisent des mécanismes de raisonnement qui ne sont pas faciles à manipuler et qui ne sont pas suffisamment réactifs.

Afin d'apporter une réponse à ces imperfections, des *architectures hybrides en couches* ont été proposées [Müller 1996]. L'idée principale est de structurer les fonctionnalités d'un agent en deux ou plusieurs couches hiérarchiques qui interagissent entre elles afin d'atteindre un état cohérent de l'agent. Une approche hybride structurée en couche présente plusieurs avantages [Müller 1996] :

- elle permet de modulariser un agent ; ainsi les différentes fonctionnalités sont clairement séparées et reliées par des interfaces bien définies ;
- elle permet une conception de l'agent plus compacte, ce qui augmente sa robustesse et facilite son "debuggage" ;
- elle accroît les capacités de l'agent car les différentes couches peuvent s'exécuter en parallèle ;
- elle augmente la réactivité de l'agent car il peut raisonner dans un monde symbolique tout en surveillant son environnement et en réagissant en conséquence ;
- elle réduit les connaissances nécessaires à une couche individuelle pour prendre ses décisions. Par exemple, une couche réactive n'aurait à utiliser que les informations concernant l'état courant de l'environnement alors qu'une couche délibérative aurait à utiliser des informations plus complexes relatives aux attitudes manipulées (croyances, buts, intentions, etc.).

3.2.3 Méthodologies de conception d'un système multi-agents

Afin de guider le processus de conception des SMAs, plusieurs méthodologies de conception ont été proposées. Bien qu'en réalité, il existe très peu de travaux sur ce sujet, nous pouvons en retenir trois : [Gutknecht et Ferber 1999] [Kinny et al.1996] et [Wooldridge et al. 1999].

J. Ferber [Gutknecht et Ferber 1998] [Gutknecht et Ferber 1999] propose un cadre méthodologique centré sur trois concepts organisationnels :

1. l'**agent** qui est défini comme une entité autonome communicante qui joue des rôles au sein de différents *groupes*. L'architecture interne de l'agent n'est pas définie afin de permettre au concepteur de choisir le modèle le plus adapté à son application ;
2. le **groupe** qui est défini comme un moyen d'identifier par regroupement un ensemble d'agents. Un groupe peut être fondé par n'importe quel agent et un agent peut être un membre d'un ou plusieurs groupes ;

3. le **rôle** qui est une représentation abstraite d'une fonction, d'un service ou d'une identification d'un agent au sein d'un groupe. Un rôle peut être attribué à plusieurs agents et un agent peut avoir plusieurs rôles.

Ce modèle est volontairement restreint afin de pouvoir s'intégrer à d'autres méthodologies. Au-dessus de ces trois concepts de base, [Gutknecht et Ferber 1999] spécifie un modèle structurel, utilisé comme outil de conception, où il définit les notions de :

1. **structure de groupe** qui est une description abstraite d'un groupe qui définit les rôles et leurs interactions et qui est représentée par un tuple : $S = \langle R, G, L \rangle$ où :
 - R est l'ensemble des rôles identifiés dans le groupe,
 - G est le graphe d'interactions entre les rôles,
 - L est le langage d'interaction.
2. **structure organisationnelle** qui est l'ensemble des groupes qui définit un modèle d'organisation multi-agents. Elle est représentée par un couple $O = \langle S, Rep \rangle$ où :
 - S est un ensemble de structure de groupes,
 - Rep est le graphe des représentants où chaque arc réunit deux rôles appartenant à deux structures de groupe différentes. Un représentant entre deux structures de groupes est un agent qui aurait simultanément un rôle dans un groupe et un second rôle dans un autre groupe.

M. Wooldridge [Wooldridge et al. 1999] propose une méthodologie, appelée *Gaia*, qui traite les niveaux *macro* (social) et *micro* (agent) de conception. Tout comme [Gutknecht et Ferber 1999], la méthodologie est indépendante de l'architecture interne de l'agent. Le processus méthodologique de Gaia implique l'utilisation d'un ensemble de modèles définis dans deux phases : *analyse* et *conception*.

Phase d'analyse

Durant la phase d'analyse, sont définies les entités abstraites qui définissent le modèle organisationnel du système. Ce dernier comprend deux modèles :

1. le **modèle de rôles** qui définit les rôles clés du système. Un rôle est caractérisé par deux types d'attributs :
 - les *permissions* ou *droits* identifiant les ressources qui peuvent légitimement être utilisées pour remplir un rôle. Elles définissent également les limites d'utilisation de ces ressources ;
 - les *responsabilités* qui définissent les fonctionnalités d'un rôle.
2. le **modèle d'interactions** qui définit les relations de dépendances entre les différents rôles. Dans ce modèle, pour chaque type d'interaction inter-rôle, est

identifié un ensemble de *définitions de protocoles*. Les *définitions de protocoles* sont composées des attributs suivants :

- le *but* : une brève description de la nature de l'interaction,
- l'*initiateur* : le ou les rôles responsables de l'initiation de l'interaction,
- le *correspondant* : le ou les rôles avec qui l'initiateur interagit,
- les *entrées* : les informations utilisées par l'initiateur,
- les *sorties* : les informations fournies par le correspondant durant l'interaction,
- le *traitement* : une brève description de tout traitement que l'initiateur du protocole exécute durant l'interaction.

Phase de conception

Durant la phase de conception, trois modèles, contenant les entités concrètes du système, sont générés :

1. le **modèle d'agent** qui identifie :
 - les *types d'agents* qui seront utilisés pour l'implémentation du système,
 - les *instances d'agents* qui traduiront ces types d'agents à l'exécution ;
2. le **modèle de services** qui définit les principaux services associés à chaque type d'agent. Un service est un bloque cohérent d'activités que l'agent s'engage à accomplir ;
3. le **modèle de communication** qui définit les liens de communication entre les types d'agent. Ce modèle est un simple graphe où les nœuds représentent les types d'agents et les arcs les chemins de communication.

Contrairement à la méthodologie de [Gutknecht et Ferber 1999], la notion de *structure organisationnelle* n'est pas définie dans Gaia.

L'approche de D. Kinny [Kinny et al.1996] [Kinny and Georgeff 1997] ne fait pas de distinction entre les phases d'analyse et de conception. Elle propose un ensemble de modèles pour définir d'un *point de vue externe* et d'un *point de vue interne* des systèmes multi-agents construits sur l'*architecture BDI*.

Point de vue externe

Du point de vue externe, un SMA est défini par deux modèles qui sont indépendants de l'architecture BDI :

1. le **modèle d'agent** qui décrit les relations hiérarchiques entre les différentes classes d'agents abstraites et concrètes. Il identifie également les instances d'agents qui peuvent exister dans le système ainsi que leur multiplicité ;

2. le *modèle d'interactions* qui décrit les responsabilités d'une classe d'agent, les services fournis et les interactions entre les classes d'agents. Ce modèle définit la syntaxe et la sémantique des messages utilisés pour :

- les communications inter-agents,
- les communications entre les agents et les autres éléments du système tels que les interfaces utilisateurs.

Point de vue interne

Du point de vue interne, sont décrites les attitudes mentales de chaque classe d'agent par l'intermédiaires de trois modèles :

1. le *modèle de croyances* qui définit les informations sur l'environnement et l'état interne d'un agent et les actions qu'il peut exécuter ;
2. le *modèle de buts* qui décrit les buts qu'un agent doit atteindre et les événements auxquels il peut répondre ;
3. le *modèle de plans* qui décrit les plans possibles qu'un agent peut utiliser pour atteindre ses buts.

3.3 Evolution des besoins de sécurité

La dynamique et l'évolution des besoins de sécurité, dues à l'évolution des systèmes et des réseaux, accroissent le besoin de systèmes flexibles et adaptatifs afin de pouvoir ajouter de nouvelles capacités et de les reconfigurer facilement. Un système de détection d'intrusions basé sur une technologie d'agents intelligents semble approprié pour répondre aux nouveaux besoins de sécurité tels que la détection de nouvelles attaques et la mise en place de nouvelles politiques de sécurité. Grâce à leurs propriétés, les agents intelligents peuvent remplir ces besoins :

- l'*adaptabilité* : les politiques de sécurité qui sont appliquées dans une entreprise peuvent changer. Par conséquent, l'administrateur doit modifier ou ajouter de nouvelles politiques afin de réadapter les tâches de détection et les fonctions de surveillance. Quand une nouvelle politique de sécurité est ajoutée ou une politique existante est modifiée, un ensemble de nouveaux buts est dérivé et envoyé aux agents intelligents concernés. Ces agents sont capables ensuite d'atteindre ces nouveaux buts ;
- la *flexibilité* : le réseau surveillé peut changer et varier avec le temps. En effet, de nouveaux services peuvent être fournis et de nouvelles applications et ressources peuvent être ajoutées. Ainsi, les systèmes de détection d'intrusions à développer doivent tenir compte de ces modifications sans qu'ils aient besoin d'être réinitialisés. En fait, dans une approche d'agents intelligents, l'administrateur, via un ensemble de nouveaux buts, spécifie ces modifications aux agents intelligents. Cela ne nécessite pas que les agents intelligents soient relancés parce que leur comportement change en fonction des buts spécifiés qu'ils doivent atteindre ;

- l'**apprentissage** : c'est une caractéristique fondamentale pour détecter de nouvelles attaques. En fait, les scénarios d'attaque évoluent continuellement dans le temps. Aussi, il est très important de pouvoir apprendre les nouvelles attaques afin de les détecter quand elles se produisent. Une nouvelle attaque peut être apprise de deux manières :
 1. la plus simple est réalisée par l'administrateur qui indique dans un ensemble de buts comment détecter les nouvelles attaques. L'agent intelligent peut alors modifier sa base de connaissances ;
 2. la seconde, est la capacité d'un agent à collecter la connaissance basée sur ses expériences passées et par conséquent modifier son comportement en réponse aux nouvelles situations. En fait, quand un agent intelligent observe une liste d'événements qui ne correspond pas à un scénario connu d'attaque, il peut suspecter que c'est un comportement intrusif en se basant sur les cas et expériences précédentes ainsi que les modèles (statistiques...) et techniques (algorithmes génétiques, réseaux neurologiques...) appropriées. Il peut également communiquer et coopérer avec d'autres agents pour obtenir plus d'informations sur ce scénario. S'il détecte que c'est un comportement intrusif, il peut ajouter le nouveau scénario d'attaque dans sa base de connaissances.

3.4 Efficacité de la détection des attaques de sécurité et les propriétés des agents intelligents

Dans cette section, nous discutons de l'importance de certaines caractéristiques pour une détection d'intrusions efficace :

- la **distribution** : un élément commun à un grand nombre d'attaques réseau est qu'un utilisateur essaye très souvent de s'attaquer à plusieurs ressources du réseau [Ko et al.1993]. Ce genre d'attaques se caractérise par des comportements anormaux à différents éléments du réseau (hosts, serveurs de fichier, routeurs, etc.). Par exemple, un intrus peut essayer d'attaquer le serveur de fichier, de rendre un service TCP indisponible sur un host spécifique ou de congestionner le réseau à un nœud particulier. Au niveau de chaque élément du réseau, des événements de sécurité caractérisant des activités anormales peuvent être observés. Détecter ce type d'attaques par un seul système, s'exécutant sur un seul élément, est fastidieux, trop compliqué et exige l'échange d'un grand nombre de messages. Ainsi, il serait plus facile de distribuer les tâches de surveillance et de traitement parmi un certain nombre d'entités qui peuvent surveiller le réseau à différents points. Cet aspect important est fourni par la plupart des systèmes de détection d'intrusions existants (comme NIDES, DIDS et CSM.). Par exemple, dans NIDES et DIDS, la collecte des données est assurée par plusieurs entités mais l'analyse est exécutée par un directeur centralisé, qui communique avec les entités de surveillance. Cependant, dans CSM, il n'y a pas de gestionnaire centralisé établi mais les différents gestionnaires ont la responsabilité de détecter les intrusions locales.
- l'**autonomie** : afin de simplifier la détection d'intrusions, la distribution des tâches de détection parmi différentes entités est nécessaire. Cependant, ce n'est pas suffisant car distribuer la collecte des événements peut également poser des

problèmes de congestion de trafic dans le réseau entre les diverses entités. D'ailleurs, une attaque qui surgit à un élément spécifique du réseau (host, serveur, etc.), se caractérise par un ensemble d'événements de sécurité qui peuvent être observés au niveau de cet élément. Par exemple, un intrus qui lance une attaque du type "TCP scan", envoie des "telnets" successifs sur chaque port TCP d'un host spécifique. Ces "telnets" seraient observés par l'entité surveillant ce host. Par conséquent, il serait plus judicieux de laisser cette entité détecter ce comportement anormal. Ainsi, les entités de gestion doivent être autonomes afin d'assurer une analyse locale et de détecter les comportements intrusifs qui se produisent sur les hosts surveillés. Les approches CSM et DIDS ont montré la nécessité d'utiliser des entités autonomes. Elles proposent différentes techniques dans le sens où la décision finale dans le système DIDS est prise par un gestionnaire centralisé, tandis que dans CSM certaines décisions peuvent être directement prises par l'entité elle-même.

- la **délégation** : un autre aspect, lié à l'autonomie, qui doit être considéré est la fonction de délégation. En fait, la forte dynamique des réseaux informatiques exige de pouvoir modifier, à tout moment, les fonctions de gestion de sécurité afin de les adapter aux changements qui se produisent dans les réseaux surveillés. Le modèle basé sur la distribution de la fonctionnalité de délégation entre diverses entités de gestion, permet de remplir ce besoin. En effet, les tâches de gestion déléguées, qui sont exécutés localement, peuvent être modifiées dynamiquement, et d'une manière flexible, à tout moment et n'importe où. Par exemple, si un administrateur veut détecter une nouvelle attaque, il devra envoyer de nouvelles tâches de surveillance et de traitement aux différentes entités autonomes. La délégation des tâches de détection d'intrusions parmi les entités de gestion semble nécessaire.

Les tâches déléguées sont structurées sous forme de buts. Dérivé d'un but global, un ensemble de sous-buts est généré et délégué aux entités autonomes. A chaque opération de dérivation, de nouveaux sous-buts sont générés. Chaque entité autonome aura à atteindre un but local, qui peut être différent du but d'une autre entité. Ainsi, chaque entité devra surveiller un degré de suspicion spécifique afin d'atteindre son but. Afin d'obtenir une vue globale de l'état de sécurité de réseau, il est alors nécessaire de corréler les informations locales en communiquant les différents niveaux suspicieux entre les diverses entités autonomes.

Cette caractéristique critique n'est retrouvée dans aucun des systèmes de détection d'intrusions existants.

- la **communication** et la **coopération** : très souvent, les attaques de sécurité réseau se composent d'attaques coordonnées qu'il n'est pas facile de détecter. En fait, la complexité des attaques de sécurité rend leur détection plus difficile par une entité individuelle. Puisque chaque entité a une vue restreinte locale du réseau, les attaques coordonnées qui se produisent à différents points du réseau ne peuvent pas être détectées par une seule entité autonome. Par exemple, si un intrus lance une attaque "doorknob rattling", il essaye plusieurs "logins" sur chaque host. Les entités autonomes, qui surveillent individuellement ces hosts, ne peuvent pas détecter cette attaque en raison du niveau bas du nombre de "logins" répétés. En fait, la surveillance du comportement local, tel que des "logins" répétés n'est pas suffisante dans ce cas-ci. Dans ce genre d'attaques, il est nécessaire de corréler les différentes analyses faites, à différents points du réseau, par les diverses entités

autonomes. Ainsi, une communication de ces analyses et une coopération entre les diverses entités de détection d'intrusions est nécessaire afin de détecter les comportements intrusifs coordonnés. Les opérations de délégation, de communication et de coopération utilisent un ensemble d'informations structurées (buts, sous-buts, niveau de suspicion, etc.) stocké dans la base de connaissances des agents.

Le système CSM a montré la nécessité de la coopération entre les gestionnaires de sécurité afin de détecter les attaques de sécurité qui ne peuvent pas être détectées par les CSMs individuels. Chaque CSM détecte les intrusions locales et communique avec les autres CSMs en s'échangeant des informations pour détecter coopérativement les activités intrusives.

- la **réactivité** : le but d'une détection d'intrusions efficace est de réagir rapidement à une attaque avant que des dommages sérieux ne puissent être causés. Par exemple, dans le cas d'une attaque d' "*ICMP flooding*", si le système de détection d'intrusions attend que soit observée une congestion du trafic dans le réseau, il sera trop tard. Ainsi, il est important de détecter une telle attaque avant qu'elle ne congestionne le réseau. L'efficacité d'un système de détection d'intrusions peut être mesurée par rapport à sa rapidité dans la détection d'une attaque avant qu'elle n'endommage sérieusement le réseau.

Les six caractéristiques (distribution, autonomie, délégation, communication, coopération et réactivité) décrites ci-dessus sont considérées comme des conditions nécessaires pour détecter des attaques efficacement et répondre aux intrusions en réduisant la période de détection. Si nous regardons ces caractéristiques et les différentes propriétés des agents intelligents décrites dans ce chapitre, il apparaît qu'une approche basée sur les agents intelligents serait appropriée pour détecter les attaques de sécurité, en particulier les attaques réseaux. Le système CSM fournit cinq de ces caractéristiques (distribution, autonomie, communication coopération et réactivité), excepté la fonction de délégation. En effet, les gestionnaires de CSM ne peuvent pas facilement adapter leurs tâches de détection aux changements qui peuvent se produire dans le réseau, tels que de nouvelles politiques de sécurité à appliquer. De plus, ils n'ont pas la capacité d'apprendre de nouvelles attaques.

Dans les sections précédentes, nous avons analysé les propriétés qui nous paraissent incontournables pour la nouvelle génération de systèmes de gestion de sécurité. Nous allons maintenant, dans la section suivante essayer d'identifier les différentes entités qui vont intervenir pour modéliser la gestion de la sécurité d'un réseau.

3.5 *Modèle du système de gestion de sécurité*

Pour modéliser le problème de gestion de sécurité, nous devons, rappelons le, prendre en compte les besoins suivants :

1. la nature distribuée du réseau à sécuriser ;
2. la variation du réseau et l'accroissement de sa complexité en termes d'applications et de services offerts ;
3. l'augmentation du nombre d'utilisateurs et la diversité de leurs profils qui changent constamment en particulier dans le cas de la mobilité ;

4. la diversité et complexité des attaques de sécurité ;
5. la variation des politiques de sécurité à appliquer dans le réseau.

Pour répondre à cela, la solution que nous proposons est à base d'agents intelligents. Par ailleurs, pour gérer la sécurité d'un réseau nous devons :

1. d'une part, gérer les *politiques de sécurité* spécifiées par l'*administrateur* ;
2. d'autre part, analyser les *événements* caractérisant les attaques de sécurité et se produisant dans le *réseau*

A partir de ces éléments, nous proposons de décomposer la gestion de la sécurité en trois plans différents (voir Figure 1)

1. *plan utilisateur*
2. *plan kernel*
3. *plan intelligence*.

Le *plan utilisateur* est constitué de deux entités :

- l'*administrateur*,
- les *politiques de sécurité*.

C'est dans ce plan que l'administrateur définira et spécifiera les politiques de sécurité à appliquer dans le réseau. Il pourra également les modifier, lorsque la configuration du réseau aura changée ou lorsqu'il voudra détecter de nouvelles attaques. En plus de la gestion des politiques, l'administrateur recevra et analysera les rapports de sécurité. Il pourra également demander des informations particulières, comme par exemple, l'état de sécurité du réseau.

Le *plan intelligence* symbolise la partie "intelligente" de notre système. Il est représenté par :

- le *système multi-agents*,
- le modèle d'information manipulé par le SMA. Ce modèle est basé sur le concept *BDI* que nous avons vu dans ce chapitre et pour lequel nous avons opté. Nous justifierons l'utilisation de ce modèle dans le Chapitre 6.

Etant donné que l'intelligence de notre système repose essentiellement sur le modèle d'information (ce que nous verrons plus tard), nous avons jugé important de le dissocier du SMA.

Le *plan kernel* représente le niveau le plus bas de notre système, à savoir :

- le *réseau* à sécuriser,
- les *événements de sécurité* se produisant dans le réseau. Ces événements, qui représentent un élément important dans le système, seront analysés pour détecter les attaques et veiller ainsi à ce que les politiques de sécurité, définies par l'administrateur, soient respectées.

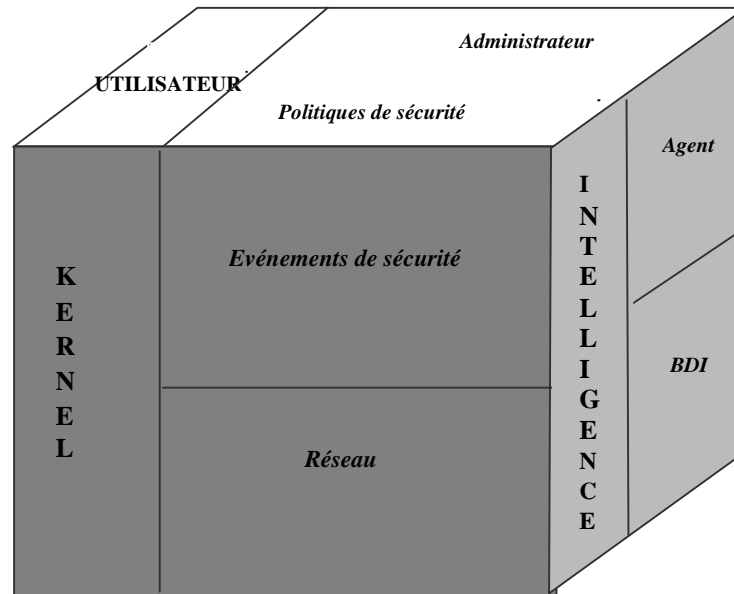


Figure 1 : Les différents points de vue de gestion de sécurité

A partir de ces trois points de vues, nous pouvons identifier les éléments d'information que devra manipuler notre système de gestion de sécurité. Si nous reprenons les plans précédents, nous pouvons d'ores et déjà identifier les éléments d'information suivants :

- les politiques de sécurité,
- les événements de sécurité
- le modèle d'information agent.

Essayons de voir comment vont interagir ces éléments d'information. En fait, lorsqu'une politique est définie, elle va permettre de spécifier les attaques que le système doit détecter. Ces attaques sont caractérisées par des événements, qui devront être analysés par le SMA et plus précisément par des éléments d'information plus abstraits.

A partir de cela, nous avons donc décomposé le modèle d'information manipulé par notre système en trois couches (voir Figure 2) :

1. les *politiques de sécurité* manipulées par l'administrateur,

2. le *modèle BDI* manipulé par le SMA,
3. le *modèle événementiel*, qui permet de formaliser les attaques de sécurité en utilisant un ensemble d'opérateurs. Ces opérateurs seront ensuite utilisés pour sélectionner les événements qui se seront produits dans le réseau.

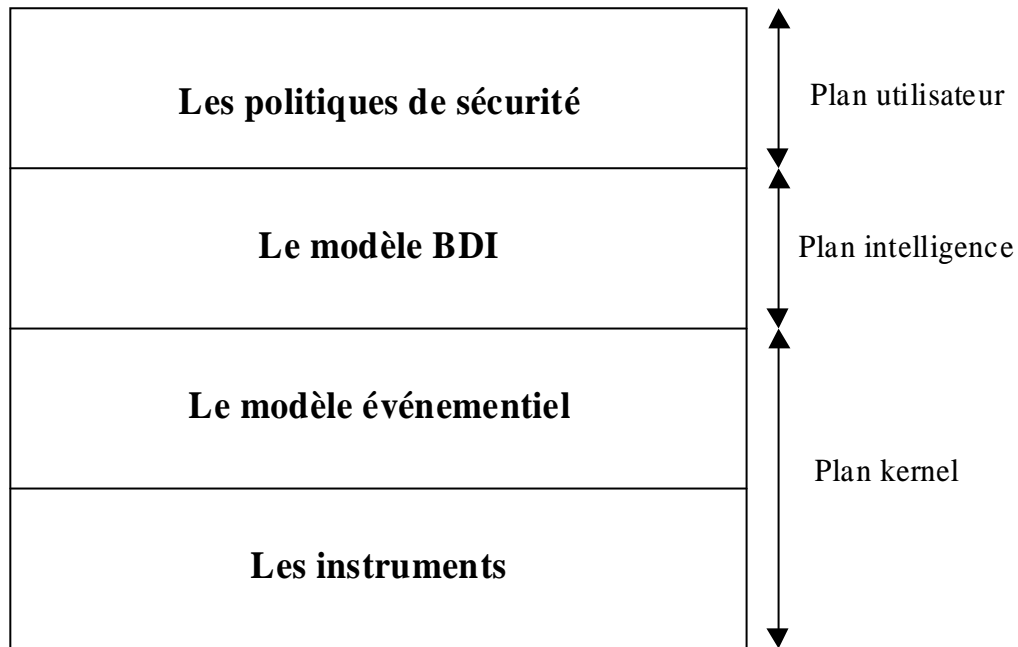


Figure 2 : La représentation en couches du modèle d'information pour la gestion de la sécurité réseau

Les politiques de sécurité représentent l'entité directrice du modèle BDI. Ce modèle va piloter à un niveau plus abstrait le modèle sous-jacent qui est un modèle concret. En réalité, nous faisons une correspondance entre les entités du modèle abstrait et celles du modèle événementiel. Ce dernier va devoir collecter les événements du réseau, en utilisant un ensemble d'outils d'instrumentation (sonde RMON, TCP Dump, etc.) que nous proposons de regrouper dans une quatrième couche, dite *instrumentation*. Cette couche contient non seulement les outils de collecte mais aussi les outils que le système peut utiliser pour agir sur le réseau (par exemple, un firewall).

Après avoir défini les différents modèles, nous allons dans les chapitres suivants décrire en détail chacun de ces modèles.

3.6 Conclusion

Dans ce chapitre, nous avons présenté nos motivations quant à l'utilisation d'une approche multi-agents pour la gestion de la sécurité, notamment la détection des attaques portant atteinte à la sécurité d'un réseau. Nous voulions montrer qu'une nouvelle solution basée sur des techniques de SMAs est prometteuse et appropriée pour répondre aux nouveaux besoins de gestion de sécurité, en raison des similitudes entre les propriétés des agents intelligents et SMAs et les caractéristiques requises pour les entités de gestion de sécurité.

Nous avons commencé par présenter les différents concepts relatifs à la technique multi-agents. Nous nous sommes ainsi intéressés :

- aux propriétés des SMAs,
- aux différentes architectures d'agents,
- ainsi qu'aux méthodologies de conception des SMAs.

En ce qui concerne les propriétés des agents et systèmes multi-agents, nous avons sélectionné celles qui nous paraissent les plus appropriées pour répondre aux besoins de gestion de sécurité, particulièrement la détection des attaques portant atteinte à la sécurité d'un réseau.

Pour ce qui est des architectures d'agents, nous avons présenté les principales architectures existantes. Nous verrons (voir Chapitre 6), que pour que notre SMA puisse :

- détecter les attaques complexes
- détecter rapidement les attaques simples,

il faudra qu'un agent de sécurité soit doté de capacités délibératives et réactives. Notre choix se portera alors sur une architecture d'agent hybride.

Enfin pour concevoir notre SMA, il nous a fallu opter pour une méthodologie de conception. Pour cela, nous avons présenté les principales méthodologies existantes. Nous verrons également (voir Chapitre 6) que la méthodologie retenue pour la conception de notre SMA sera en réalité une méthodologie hybride des trois approches retenues.

Dans une deuxième partie, nous avons analysé les caractéristiques nécessaires pour répondre aux nouveaux besoins de sécurité et pour détecter efficacement les attaques.

Finalement, nous avons présenté une vue globale de notre modèle de gestion de sécurité.

Dans les chapitres suivants nous allons présenter chacun des plans du modèle. Nous commencerons par décrire le *plan kernel* puis le *plan utilisateur* et enfin le *plan intelligence*.

Chapitre 4 Gestion des événements de sécurité

4.1 Introduction

Dans ce chapitre, nous décrivons le *plan kernel* de notre modèle de gestion de sécurité (voir Figure 1 p51). Dans ce plan, nous traitons les *événements de sécurité* qui permettent de détecter les attaques portant atteinte à la sécurité d'un système et d'un réseau. Pour détecter ces attaques, qui se caractérisent par des *suites d'événements*, nous avons besoin de :

1. définir les *schémas d'attaque* à détecter
2. traiter les événements caractérisant ces schémas d'attaque. Le traitement de ces événements se décompose en trois phases (voir Figure 3) :
 - ◆ filtrer les événements à l'aide de *critères* définis par les schémas d'attaque ;
 - ◆ regrouper les événements filtrés à partir des schémas d'attaque pour construire des suites d'événements corrélés ;
 - ◆ analyser les suites d'événements corrélés pour identifier des schémas d'attaque.

Pour représenter les schémas d'attaque et traiter les événements, nous définissons un modèle de gestion des événements de sécurité, nommé *modèle événementiel* (voir Figure 2 p52). Dans ce modèle, nous décrivons différents éléments (événement, suite d'événements, *filtre d'événements* et schéma d'attaque) et des opérateurs sur ces éléments par le biais d'un *langage descriptif*. Pour faciliter son implémentation, nous proposons un langage sous-jacent, que nous appelons *langage symbolique*. Ce langage fournit un moyen de compréhension plus simple pour la représentation des éléments et le traitement des différents opérateurs.

Dans ce chapitre, nous décrivons le modèle conceptuel de données. Puis, nous présentons le modèle conceptuel de traitement des données où nous décrivons les

opérateurs du *langage descriptif*, le *langage symbolique* et l'algorithme de traitement des données.

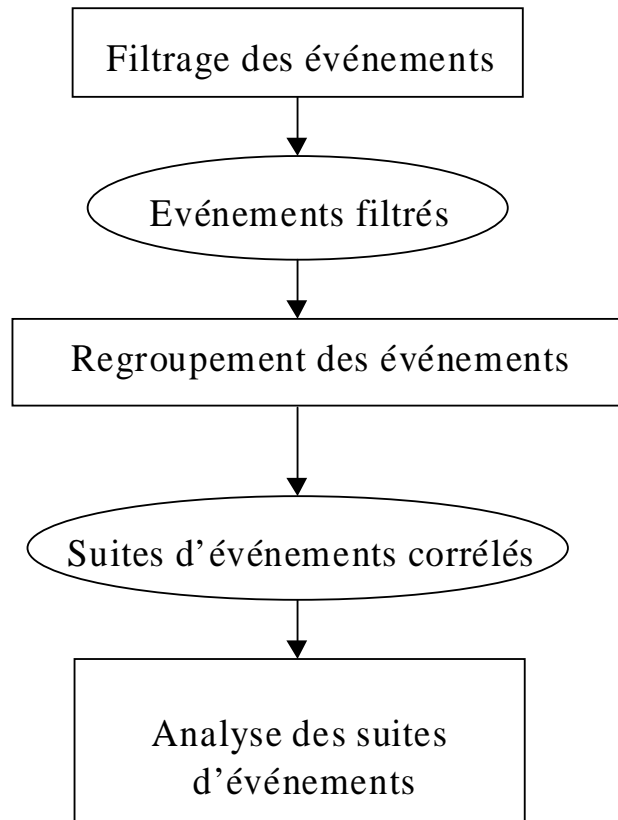


Figure 3 : Traitement des événements de sécurité

4.2 *Modèle conceptuel de données*

Dans notre modèle conceptuel de données, nous distinguons les éléments d'information suivants (voir Figure 4) :

1. les messages et les événements,
2. les filtres d'événements
3. les suites d'événements ;
4. les schémas d'attaque.

Les filtres d'événements permettent de sélectionner les messages pertinents en vue d'obtenir les événements à analyser.

Les schémas d'attaque permettent :

- de définir les filtres d'événements
- de regrouper les événements de sécurité en suites d'événements.

Dans cette section, nous décrivons les différents éléments du modèle.

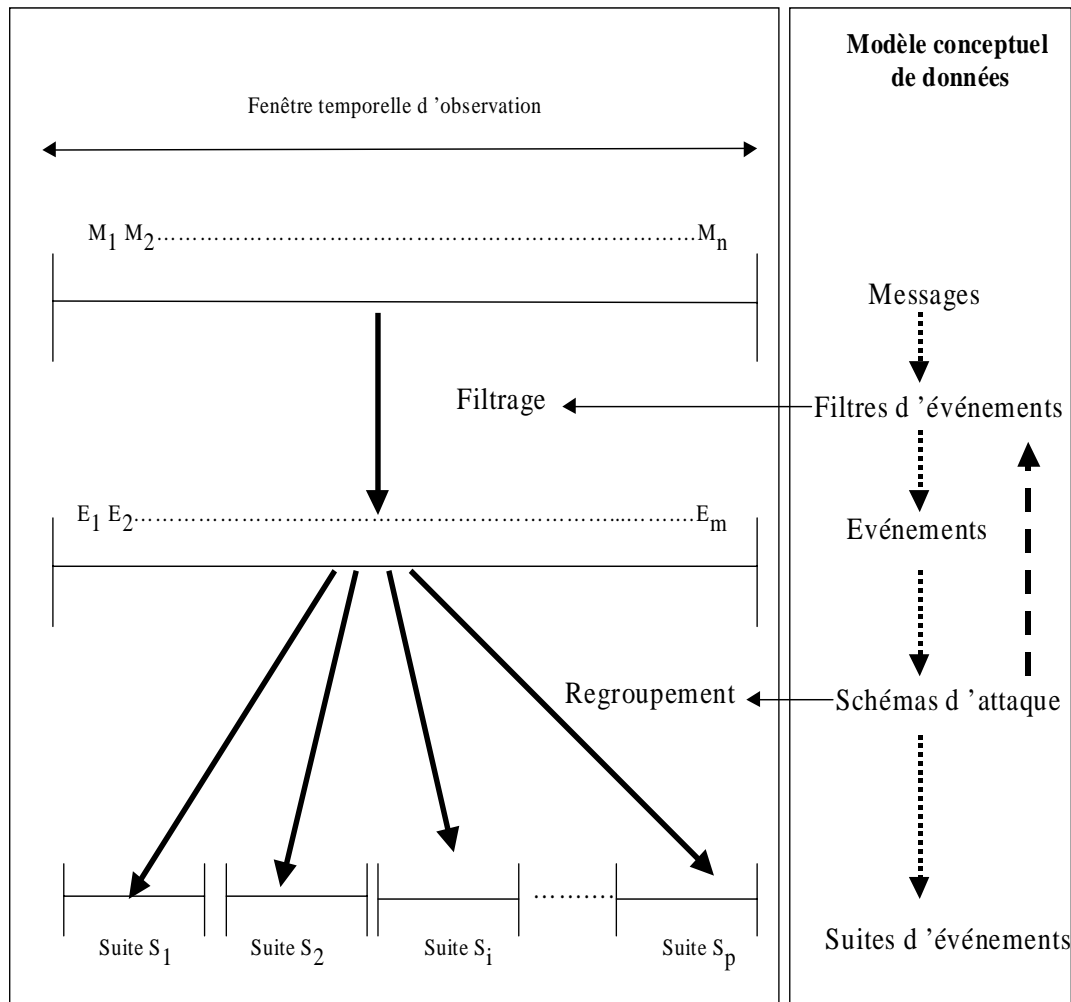


Figure 4 : Sélection des événements et des suites d'événements

4.2.1 Les événements

Nous nous intéressons au trafic réseau et aux fichiers logs afin d'analyser les *messages réseau* (message ICMP, message TCP, etc.) et les *messages systèmes*. Soit e , un *événement de sécurité*. e est défini par un critère de sélection :

- **simple** ou **mono-message** pour un critère de sélection portant sur les attributs d'un seul message (exemple la source et/ou destination d'un paquet réseau). Nous associons à ce message, une notion temporelle, qui est l'instant (le temps) où il est capturé et enregistré ;
- **multiple** ou **multi-messages** pour un critère de sélection portant sur plusieurs messages distincts. Dans ce cas e serait le résultat du couplage de deux ou plusieurs messages (exemple une requête et sa réponse).

D'un point de vue théorique, nous avons besoin de retenir une sélection *multi-messages*. Cependant, nous limiterons son utilisation à cause de la difficulté de sa mise en œuvre. En effet, en terme d'implémentation, nous ne pouvons pas collecter des messages de manière illimitée. Nous devons définir une fenêtre temporelle d'observation ; ce qui nous empêche de voir tous les messages. Pour cela, dans la pratique, nous nous intéresserons soit à une requête soit à sa réponse.

Exemple

Supposons que nous voulions :

- collecter des événements de type "échec de login",
- analyser "le nombre d'échec de login".

En théorie, il nous faut coupler deux messages : la demande de connexion ("login") et le résultat de la demande (échec). En pratique, nous pouvons :

- soit ne prendre en compte que les demandes de "logins" (répétition d'une même demande de "login" laisse supposer que les précédents ayant échoué). Dans ce cas, nous ne comptons que le nombre de "logins" (pour un utilisateur ou une origine, par exemple) ;
- soit prendre en compte les réponses de "login" et dans ce cas la réponse est suffisante.

La sélection des messages se fait sur une fenêtre temporelle d'observation. Une fois les critères de sélection définis, un ensemble d'événements de sécurité est collecté dans cette fenêtre. Nous définissons cet ensemble comme une succession ordonnée et datée d'événements de sécurité. Ces événements sont caractérisés par un ensemble d'attributs que nous allons voir dans la prochaine sous-section.

4.2.1.1 Attributs d'un événement de sécurité

Un événement peut être collecté :

- soit à partir des fichiers logs (systèmes ou firewall),
- soit à partir du trafic réseau en utilisant un outil de capture (TCP Dump, TCP Wrapper, une sonde RMON, etc.).

Ces sources de collecte et les définitions de [Udod 1985] [Mé 1994] [Mounji 1997], nous conduisent à définir les attributs suivants :

- ***point d'observation*** de l'événement (fichier logs ou trafic réseau),
- type de l'événement,
- date et heure de l'événement,

- auteur de l'événement,
- source de l'événement,
- destination de l'événement,
- ressources utilisées/objet sur lequel a porté l'événement,
- résultat de l'événement,
- *type de danger* de l'événement,
- *type d'activité* qui a généré l'événement.

Un événement de sécurité est donc caractérisé par son point d'observation, son type, un attribut temporel qui représente l'instant où s'est produit l'événement et un ensemble d'attributs atemporels (auteur, source, destination, ressources utilisées, résultat, type de danger et activité). Revenons sur la description des deux derniers attributs *type de danger* et *activité*.

Description de l'attribut *type de danger*

Nous introduisons l'attribut *type de danger* afin de caractériser, après analyse, la nature de danger d'un événement. Cet attribut peut prendre trois valeurs possibles : *normal*, *dangereux* ou *suspicieux*.

- *normal* s'il ne représente aucun danger pour le réseau et/ou système ;
- *dangereux* s'il représente une menace pour le réseau et/ou système, i.e. une attaque (exemple : accès au fichier password par un utilisateur non autorisé) ;
- *suspicieux* s'il n'est ni normal ni dangereux.

Description de l'attribut *type d'activité*

Nous avons vu (voir Chapitre 2) qu'un événement reportait les attributs de l'activité d'un utilisateur [Mounji 1997]. Par l'intermédiaire de l'attribut *activité*, nous voulons typer les activités utilisateurs en fonction du type de réseau source et destination. Par rapport au réseau que nous voulons sécuriser, nous distinguons quatre types de réseaux :

1. le *réseau externe*, qui est l'Internet
2. le *réseau Intranet*, qui est le réseau d'entreprise,
3. les *réseaux locaux* qui sont constitués d'un ensemble de domaines ;
4. les *domaines*, qui sont des sous-réseaux locaux. Un domaine est représenté par un ensemble de machines regroupées :
 - soit suivant la structure de l'entreprise ;

- soit suivant d'autres critères tels que le niveau de sensibilité des machines, la situation géographique des machines, etc.

En fonction de ces types de réseaux, nous identifions alors quatre types d'activités (voir Figure 5) :

- les **activités extranets** : activités externes au réseau d'entreprise. Ce sont les activités entre le réseau d'entreprise et l'extérieur. Il y a typiquement les activités de connexion et de transfert vers/de l'extérieur ;
- les **activités intranets** : activités internes au réseau d'entreprise mais externes au réseau local. Ce sont les activités entre le réseau local et le réseau distribué ;
- les **activités internes** : activités internes au réseau local. Ce sont les activités entre plusieurs domaines d'un même réseau local ;
- les **activités locales** : activités internes à un domaine.

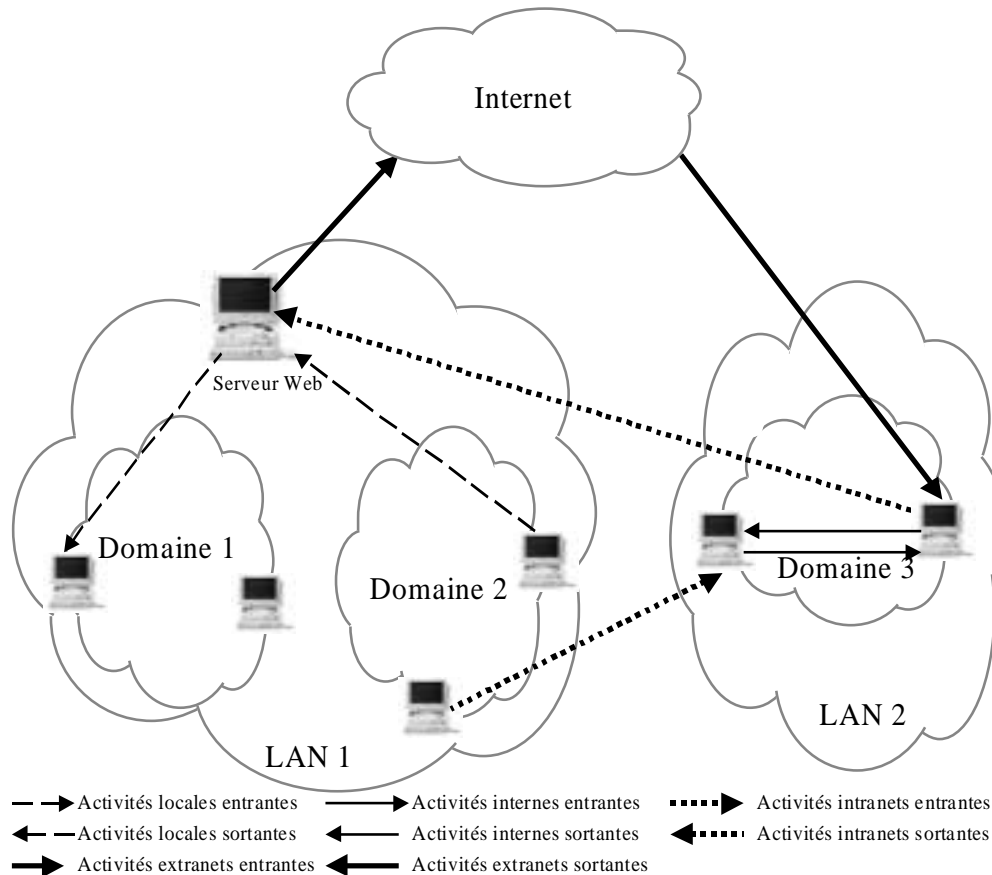


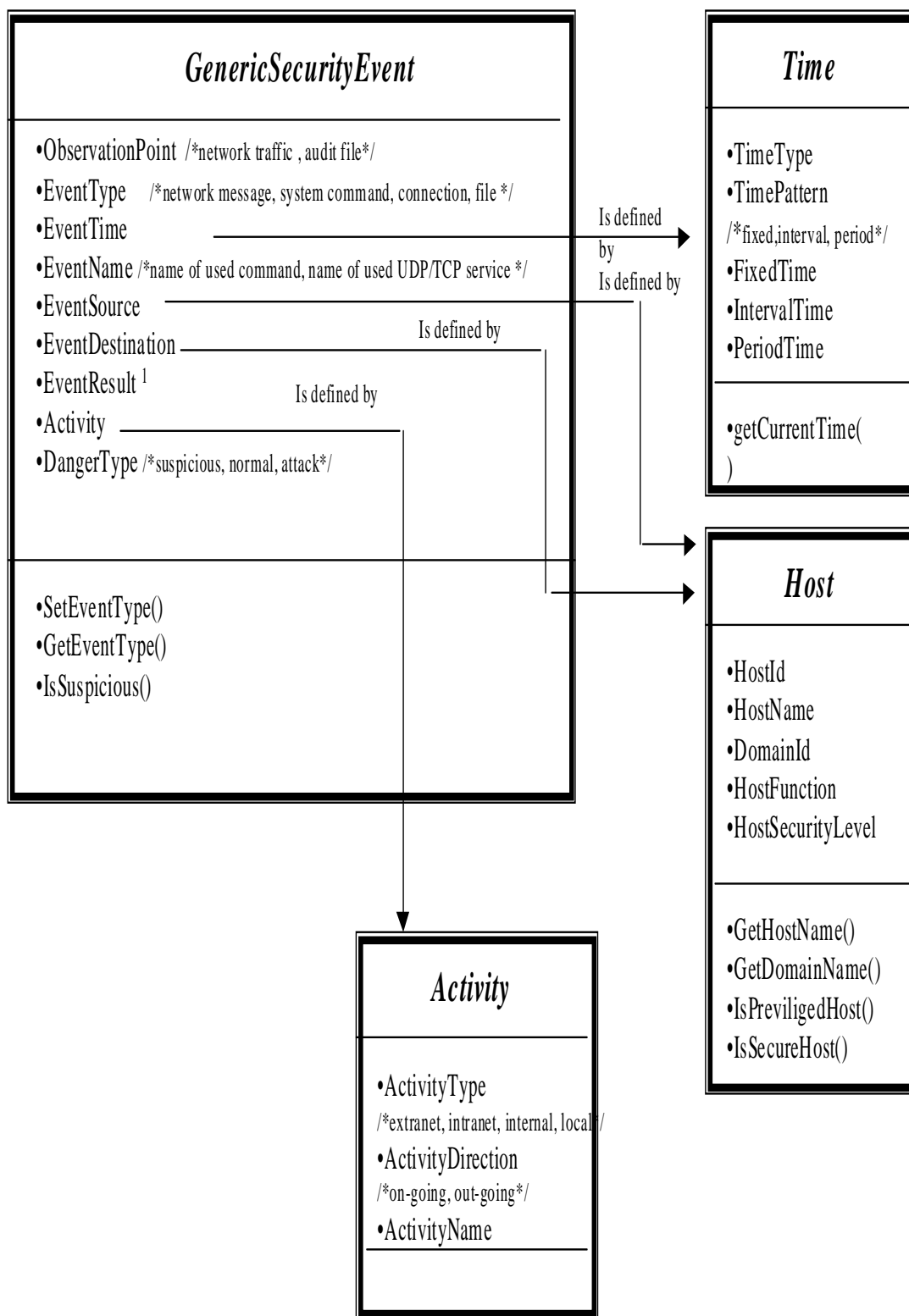
Figure 5 : Les différents flux d'activité dans un réseau d'entreprise

De plus, en fonction de l'origine de ces activités, elles peuvent être : soit *sortantes*, soit *entrantes* (voir Figure 5). Par conséquent, suivant le type et le sens de l'activité, nous distinguons :

- Pour les *activités extranets* :
 - ◆ les *activités extranets entrantes* : la source est externe au réseau d'entreprise,
 - ◆ les *activités extranets sortantes* : la destination est externe au réseau d'entreprise.
- Pour les *activités intranets* :
 - ◆ les *activités intranets entrantes* : la source est externe au réseau local et interne au réseau d'entreprise,
 - ◆ les *activités intranets sortantes* : la destination est externe au réseau local et interne au réseau d'entreprise.
- Pour les *activités internes* :
 - ◆ les *activités internes entrantes* : la source est interne au réseau local,
 - ◆ les *activités internes sortantes* : la destination est interne au réseau local
- Pour les *activités locales* :
 - ◆ les *activités locales entrantes* : la source est interne au domaine,
 - ◆ les *activités locales sortantes* : la destination est interne au domaine.

Le type et le sens d'une activité sont définis dans la classe "*Activity*" (voir Figure 6), à partir de laquelle est instancié l'attribut *activité*.

Ces différents attributs permettent de filtrer les événements de sécurité à traiter. Ces événements sont filtrés en fonction de leur type et de la valeur de certains attributs atemporels. Ces attributs peuvent être différents suivant le type de l'événement. Pour cela, nous proposons de classer les événements de sécurité en fonction de leur type. Ce qui nous a amené à définir plusieurs classes (voir 4.2.1.2). Néanmoins, il existe un ensemble d'attributs qui sont communs à tous les types d'événements et qui sont regroupés dans la classe "*GenericSecurityEvent*" (voir Figure 6).



¹ authentication failure, authentication success, access control failure, access control success, illegitimate access, legitimate access, illegitimate modification, legitimate modification, illegitimate transfer, legitimate transfer

Figure 6 : La classe principale d'événement de sécurité

4.2.1.2 Classification des événements de sécurité

Suivant le type de l'événement, nous pouvons avoir plusieurs classes d'événements possibles. Chaque classe hérite de la classe principale "*GenericSecurityEvent*" et se différencie des autres par des attributs atemporels supplémentaires qui lui sont propres. Dans le cadre de ce travail, nous proposons les classes d'événements décrites dans la Figure 7 : "*AuditEvent*", "*NetworkEvent*", "*FileEvent*", "*SystemEvent*", "*ConnectionEvent*", "*UDPEvent*", "*TCPEvent*" et "*ICMPEvent*".

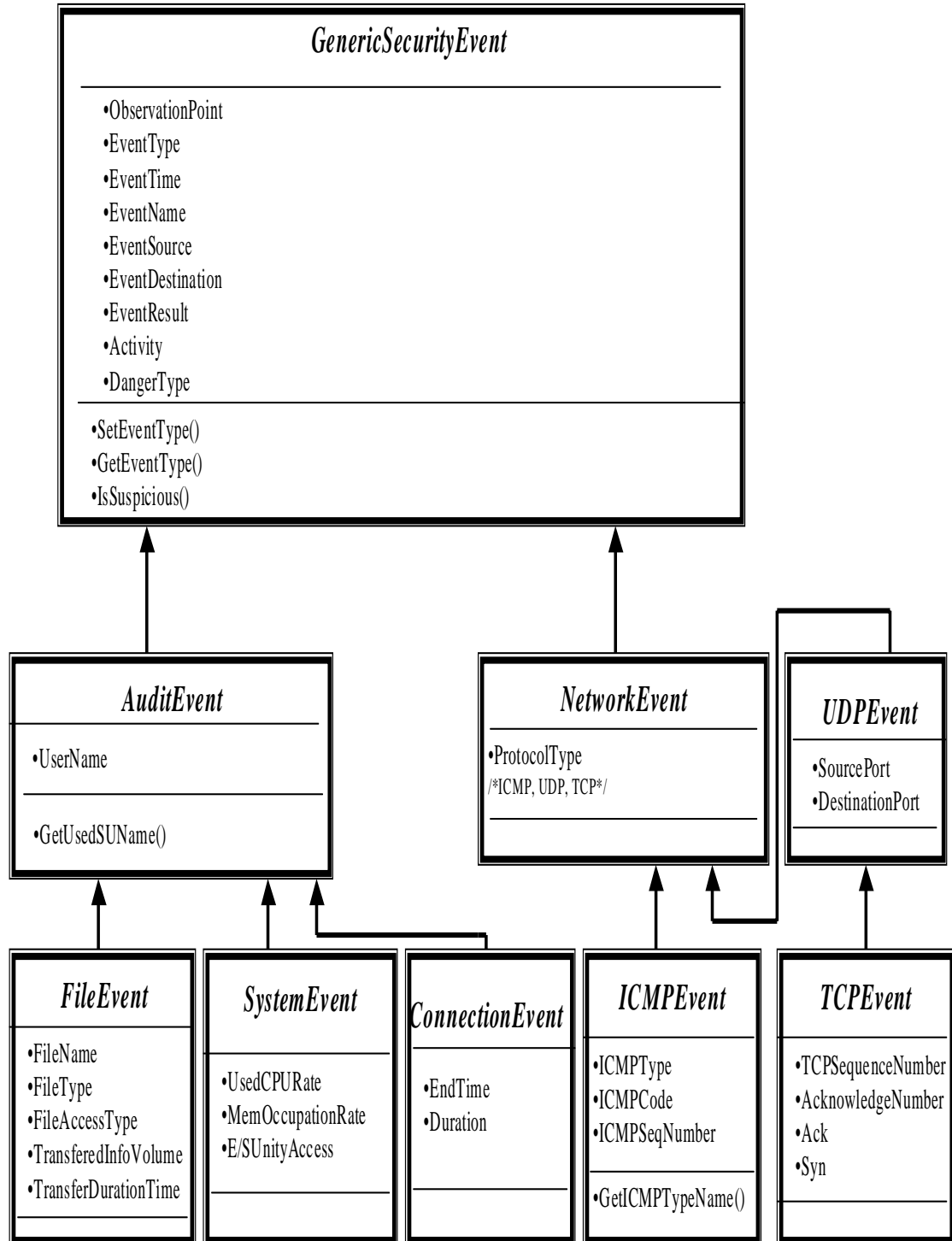


Figure 7 : Les classes d'événements de sécurité

Les événements de sécurité (messages) sont filtrés en fonction de la classe à laquelle ils appartiennent. Chaque classe implémente une fonction de filtrage qui lui est propre.

A l'issue de ce filtrage, nous obtenons un ensemble d'événements. Pour traiter ces événements, nous devons les regrouper en fonction des profils d'attaque que nous cherchons à détecter. A l'issue de ce regroupement, nous obtenons des suites d'événements.

4.2.2 Les filtres d'événements

Pour spécifier les critères de sélection mono-message et multi-messages, nous distinguons un nouvel élément d'information, que nous nommons *filtre d'événement*.

Un *filtre d'événement* est défini par la classe "*EventDescription*" (voir Figure 8) qui caractérise l'événement à filtrer en spécifiant :

- des valeurs d'attributs qui concernent essentiellement le point d'observation de l'événement, son type et certains attributs atemporels (exemple la source et/ou la destination de l'événement).
- des opérateurs de comparaison sur ces valeurs d'attributs. Ces opérateurs spécifient les critères de sélection des événements.

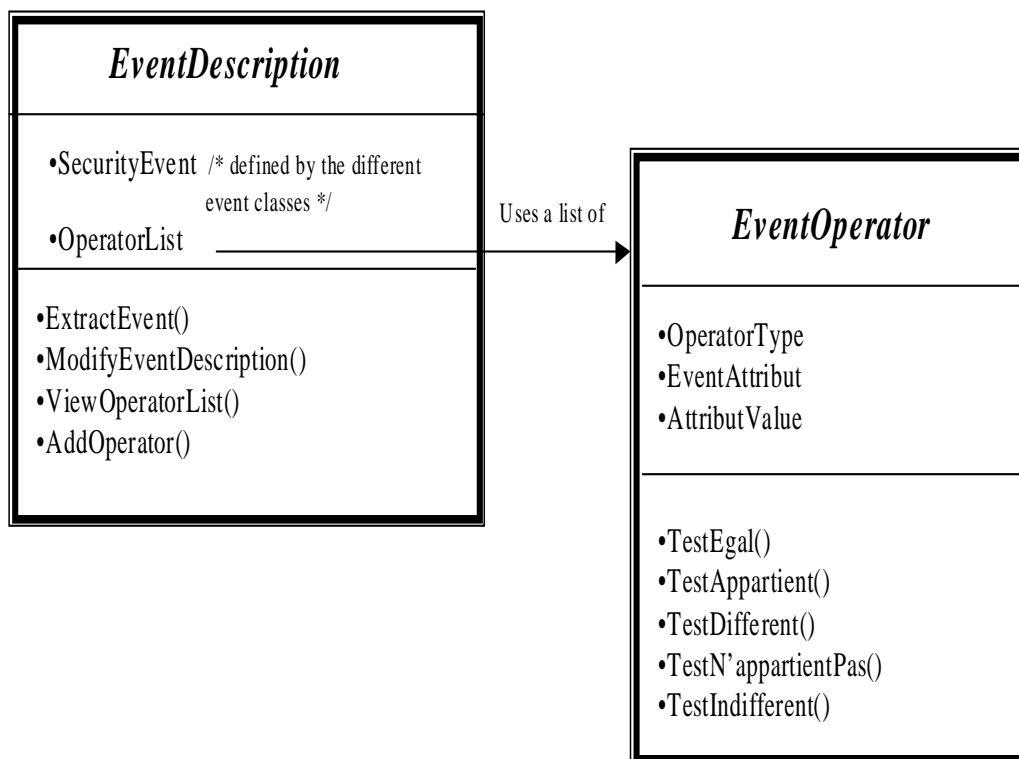


Figure 8 : La classe filtre d'événement

Pour décider si un événement doit être collecté ou non, une évaluation des différents opérateurs est nécessaire. Pour l'évaluation de ces opérateurs nous définissons une méthode "*extractEvent()*" (voir Figure 9), qui est la fonction principale de la classe

"*EventDescription*". Cette fonction est un ET logique entre les résultats d'évaluation des différents opérateurs de comparaison. Autrement dit, cette fonction retourne "vrai" si les résultats d'évaluation des différents opérateurs sont "vrai". Dans ce cas, cela signifie que l'événement évalué correspond au *filtre d'événement* et qu'il peut être collecté.

Pour le traitement interne de la classe "*EventDescription*", nous définissons les méthodes suivantes :

- "*modifyEventDescription()*" pour modifier des valeurs d'attributs d'événement,
- "*addOperator()*" pour ajouter un opérateur de comparaison,
- "*viewOperatorList()*" pour visualiser les différents opérateurs de comparaison.

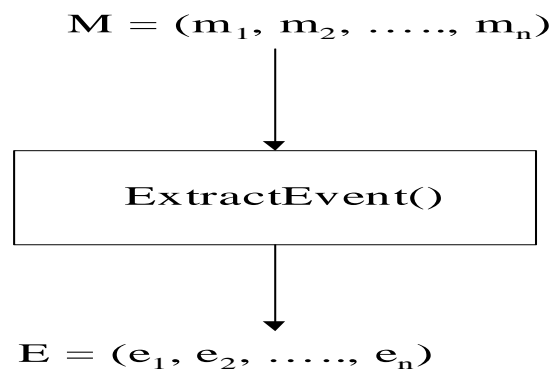


Figure 9 : Sélection des événements

4.2.3 Les suites d'événements

Dans le cadre de ses activités, un utilisateur ou éventuellement un groupe d'utilisateurs génère des suites d'événements corrélés. Nous définissons une *suite d'événements* comme étant un *sous-ensemble d'événements de sécurité* extrait de l'ensemble total des événements collectés dans une fenêtre temporelle d'observation. L'extraction de ce sous-ensemble se fait en utilisant un *critère de sélection des suites d'événements*. Ce critère permet une corrélation des événements par rapport à des attributs pour identifier les événements qui appartiennent à la même suite. Cette corrélation peut porter sur le nom de l'utilisateur, l'adresse source, l'adresse destination,...

Les paramètres caractérisant une suite d'événements et définis par la classe "*EventSeriesFeatures*" (voir Figure 10) sont :

- la fenêtre temporelle d'observation,
- la *suite d'événements* extraite de l'ensemble total d'événements,
- le *type de la suite*,

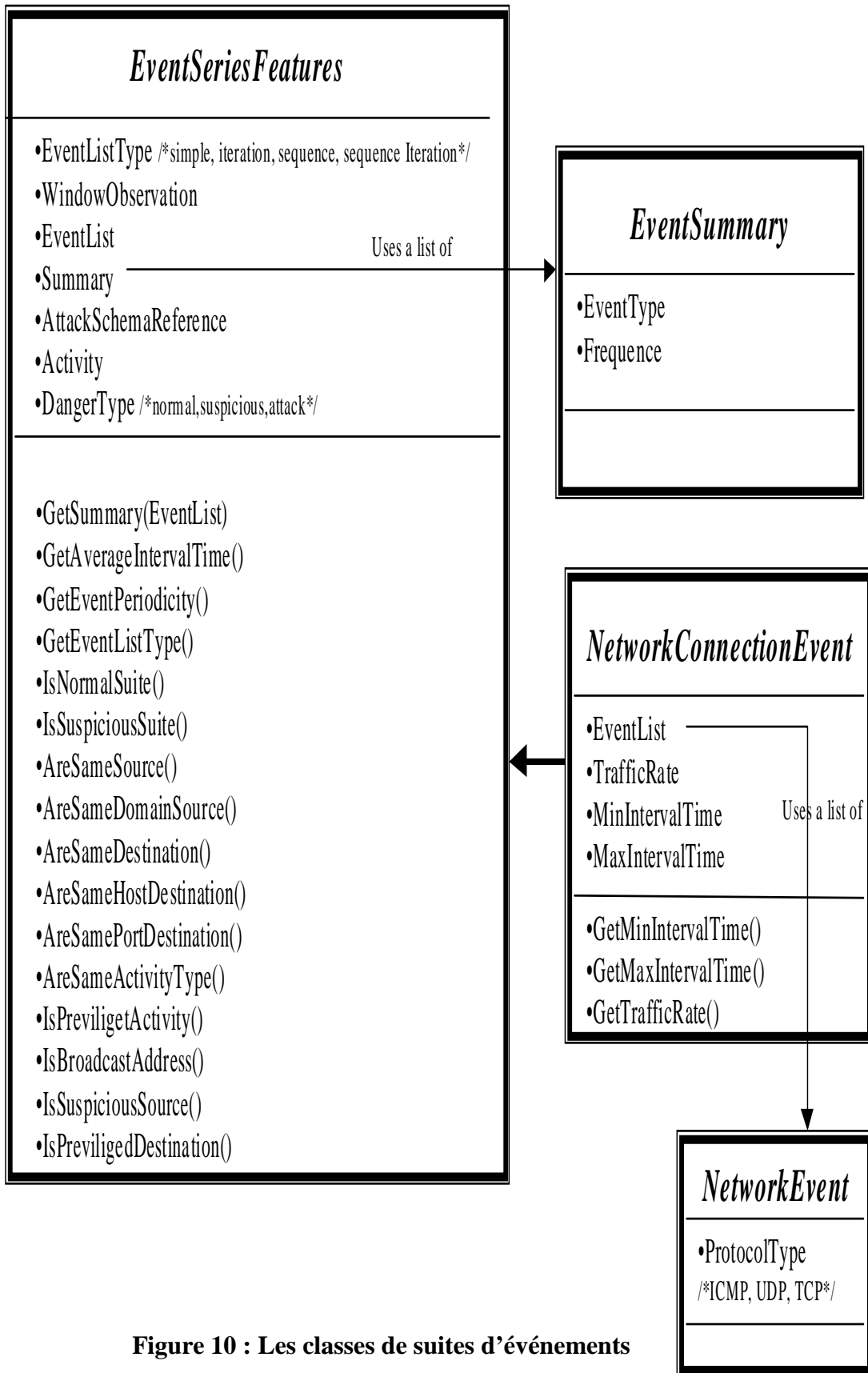


Figure 10 : Les classes de suites d'événements

- le *résumé de la suite*,
- la *référence du schéma d'attaque*,
- le *type d'activité*
- le *type de danger*.

Les cinq derniers attributs sont décrits dans les paragraphes suivants.

Description de l'attribut *type de la suite*

Une suite d'événements peut se traduire par :

- un seul événement simple ;
- l'itération d'un événement : un même type d'événement qui se répète un certain nombre de fois dans la suite (ce qui introduit la notion de seuil d'événements) ;
- une séquence d'événements : des événements qui se succèdent (de manière ordonnée ou non) ;
- la répétition d'une séquence d'événements (ce qui introduit la notion de seuil de séquence d'événements).

Description de l'attribut *résumé de la suite*

Un grand nombre d'attaques se traduit par la répétition d'un même (ou plusieurs) type(s) d'événements. Dans ce cas, par souci de facilité de traitement ultérieur, nous introduisons le *résumé de la suite*. Ce *résumé* est en fait une version simplifiée de la suite qui contiendra les types d'événements et la fréquence de répétition de ces événements.

Description de l'attribut *référence du schéma d'attaque*

Les critères d'extraction des suites sont définis par rapport à des schémas d'attaque (élément d'information que nous verrons plus en détail dans la sous-section 4.2.4) à détecter. Pour cela, nous introduisons l'attribut *référence du schéma d'attaque* auquel est associée la suite d'événements observée.

Description de l'attribut *type d'activité*

L'attribut *type d'activité* a la même signification que pour un événement (voir 4.2.1.1 p59).

Description de l'attribut *type de danger*

L'objectif de cette étude est de pouvoir caractériser des risques d'attaque. Nous prévoyons pour cela un attribut *type de danger* qui permettra, après analyse, de caractériser la suite. La façon d'analyser, de traiter et de détecter une attaque sera vue plus tard. Nous avons introduit ce même attribut lors de la description des

caractéristiques d'un événement. Tout comme un événement de sécurité, une suite d'événements peut être :

- *normale* si elle ne représente aucune menace pour le réseau et/ou système ;
- *dangereuse* si elle correspond à un profil d'attaque ;
- *suspicieuse* si elle n'est ni normale ni dangereuse.

Il existe une relation entre le *type de danger* d'un événement et celui d'une suite d'événements (contenant cet événement). En effet, le niveau de danger d'une suite est au moins égal au niveau de danger le plus élevé des événements de la suite. La valeur de cet attribut sera supérieure ou égale au plus grand danger des valeurs de l'attribut *type de danger* des événements constituant la suite.

Cet attribut a un rôle très important dans le processus de gestion des attaques. En effet, suivant le niveau de danger de la suite d'événements, des actions différentes seront prises :

- si la suite est *normale*, il n'y aura rien à faire ;
- si la suite est *suspicieuse*, il faudra modifier le comportement du processus de détection ;
- si la suite correspond à une attaque, il faudra exécuter un plan d'actions défini par les politiques de sécurité.

En résumé, les événements de sécurité sont filtrés à deux niveaux. Le premier filtrage permet de sélectionner les événements par rapport aux messages. Quant au second filtrage, il permet de sélectionner les suites d'événements par rapport à un ensemble d'événements. Nous reviendrons sur le filtrage dans la section 4.3.3.

4.2.4 Les schémas d'attaque

Nous définissons un ensemble de schémas d'attaque qui représentent des types d'attaques connues. Ces schémas sont instanciés par l'activation d'une politique de sécurité (nous le verrons plus en détail dans le chapitre suivant). Lors de l'instanciation de ces schémas, sont également instanciés les *filtres d'événements*.

Un *schéma d'attaque* est une classe qui caractérise un type d'attaque de sécurité. Une *instance de schéma d'attaque* est une description d'attaque paramétrée. Prenons l'exemple du type d'attaque : "échecs répétés de login" et de l'attaque "échecs répétés de logins provenant du Site A". La première est représentée par un schéma d'attaque et la seconde est représentée par le même schéma d'attaque mais avec une valeur de l'attribut source spécifiée et donc instanciée.

Nous avons vu, dans la section précédente, qu'à partir de l'ensemble des messages réseaux et systèmes, les *filtres d'événements* définissent des *critères* pour sélectionner un ensemble d'événements pertinents *E*. Le but des schémas d'attaque est d'extraire,

à partir de E , des sous-ensembles d'événements qui correspondent à des suites d'événements corrélés pour pouvoir ensuite les analyser (voir Figure 11).

La classe schéma d'attaque a un double objectif :

1. le premier est de sélectionner les événements pertinents,
 2. le second est de sélectionner les suites à partir des événements filtrés ;
- afin de pouvoir reconnaître des scénarios d'attaques.

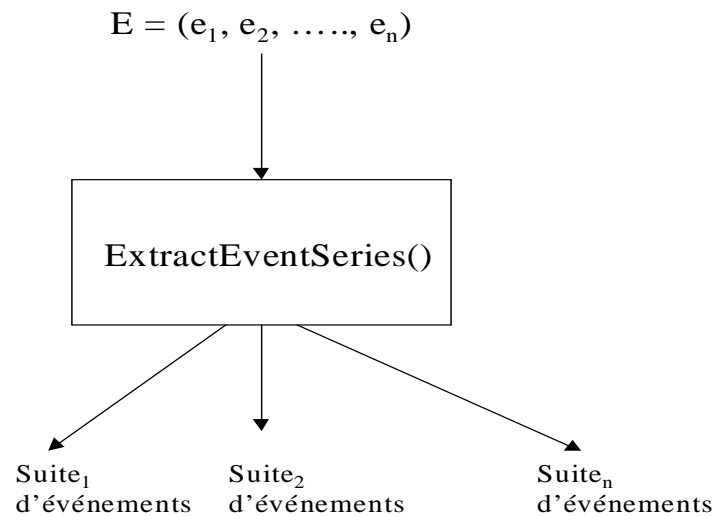


Figure 11 : Sélection des suites d'événements

Afin de pouvoir décrire un schéma d'attaque, puis l'instancier, des opérateurs sont nécessaires pour :

- exprimer des conditions et donner des valeurs aux attributs d'événements ;
- exprimer des contraintes temporelles et des critères de corrélation des événements ;
- reconnaître des similarités entre une suite d'événements et un schéma d'attaque.

Les paramètres d'un schéma d'attaque, définis dans la classe "*AttackSchema*" (voir Figure 12) sont :

- la fenêtre temporelle dans laquelle sont observés les événements ;
- une liste de références à la classe "*EventDescription*" qui caractérise les événements à filtrer et définit les critères de sélection des événements ;
- des opérateurs de comparaison entre les attributs des événements. Ces opérateurs définissent les *critères de sélection* qui permettront de corréler les événements et de générer des sous-ensembles d'événements corrélés ;

- des opérateurs qui permettront d'analyser les suites d'événements pour identifier des attaques.

Afin de traiter les opérateurs, nous définissons deux fonctions principales pour :

- réaliser l'extraction des suites d'événements,
- analyser les suites extraites.

Extraction des suites d'événements

L'extraction des suites d'événements est réalisée par la méthode "*extractEventSeries()*". Cette méthode permet, en fonction des opérateurs de comparaison définis dans la classe, de regrouper les événements qui référencent la même instance de schéma d'attaque. Cette fonction a :

- comme entrée l'ensemble d'événements observé dans la fenêtre temporelle ;
- comme sortie, une instance de la classe "*EventSeriesFeatures*", qui est une suite d'événements corrélés.

Analyse des suites d'événements

Une fois l'extraction terminée, il faut maintenant analyser les suites pour voir si elles correspondent à une instance de schéma d'attaque. Pour cela, nous avons défini une autre méthode "*evaluateEventList()*". Tout comme la fonction d'extraction des suites, cette fonction est assez complexe car :

- l'ensemble des événements filtrés se construit au fur et à mesure que les événements arrivent,
- et de ce fait l'évaluation des suites doit se faire "au fil de l'eau".

Afin de simplifier le traitement des fonctions d'extraction et d'analyse, nous proposons une façon de le réaliser dans la section 4.3.2

Pour le traitement interne de la classe, nous définissons les méthodes suivantes :

- "*getEventDescription()*" pour lire le contenu d'un filtre d'événement,
- "*addEventDescription()*" pour rajouter une référence de filtre d'événement,
- "*modifyEventDescriptionList()*" pour modifier la liste des références de filtres d'événements,
- "*addOperatorList()*" pour ajouter un opérateur,
- "*viewOperatorList()*" pour visualiser les différents opérateurs en fonction de leur type.

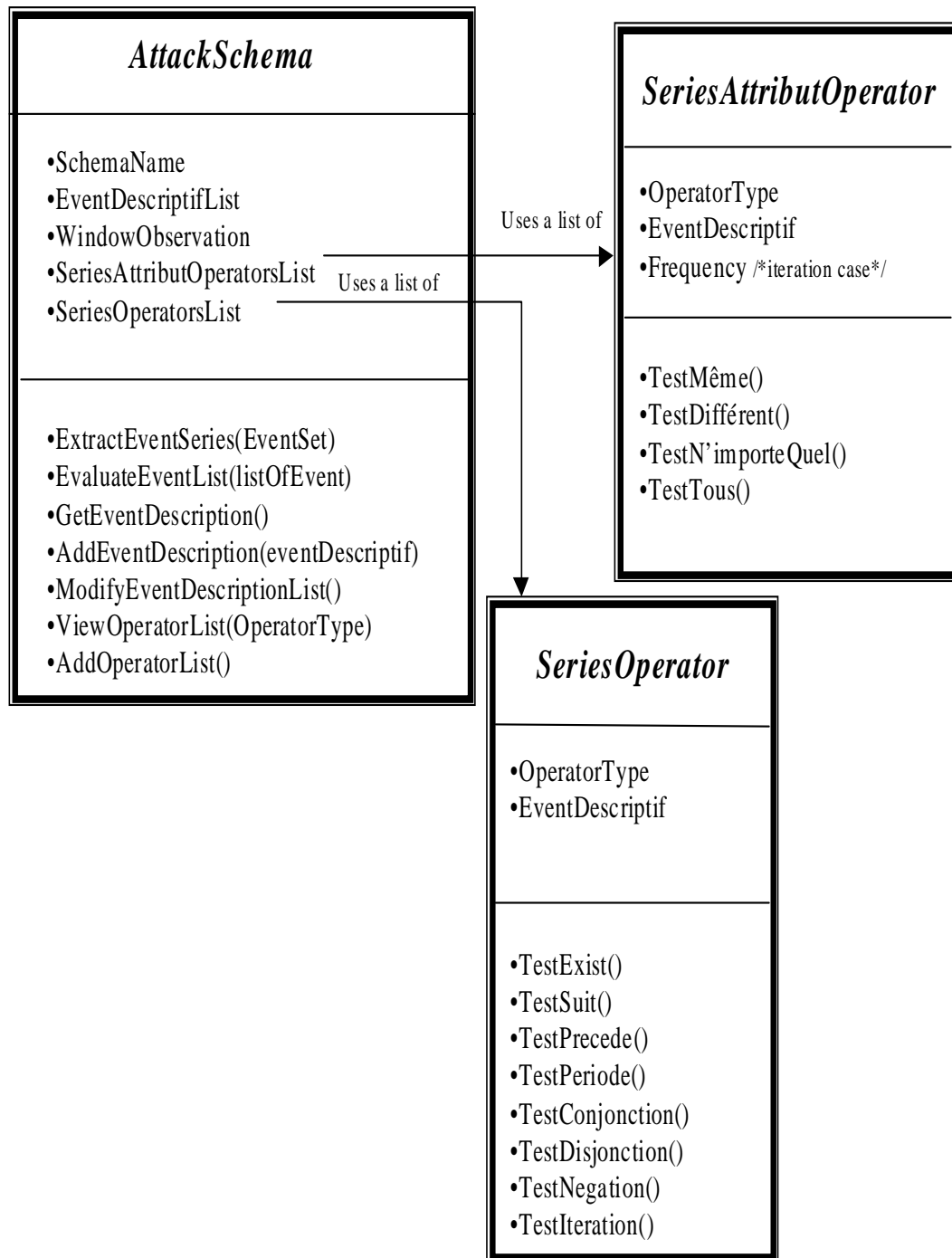


Figure 12 : La classe schéma d'attaque

4.2.5 Phénomène de l'oubli

Les événements de sécurité sont observés dans une fenêtre temporelle et stockés en mémoire. La taille de la mémoire n'est pas illimitée, ce qui signifie qu'il faut gérer l'âge des événements. Pour cela, nous proposons deux méthodes de gestion :

- avoir une fenêtre temporelle glissante (Figure 13) mais avec un certain degré de complexité en terme d'implémentation ;
- ou simuler la fenêtre temporelle en gérant des périodes temporelles de la manière suivante (voir Figure 14) :
 - nous choisissons une période de telle sorte à ce qu'il y ait plusieurs périodes dans la fenêtre temporelle ;
 - à chaque période, nous associons une table contenant les événements produits dans cette période ;
 - à la fin de chaque période, nous déterminons la table la plus ancienne et nous l'éliminons. Par conséquent, tous les événements qui ne sont pas dans la période sont oubliés.

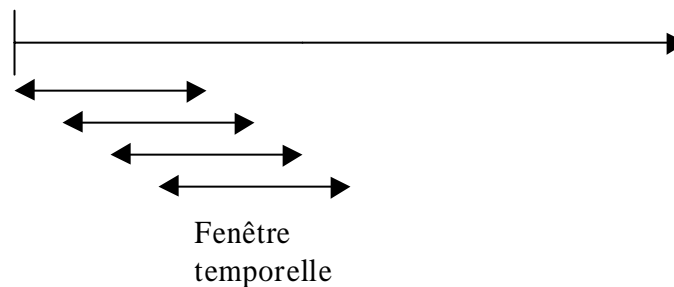


Figure 13 : Fenêtre temporelle glissante

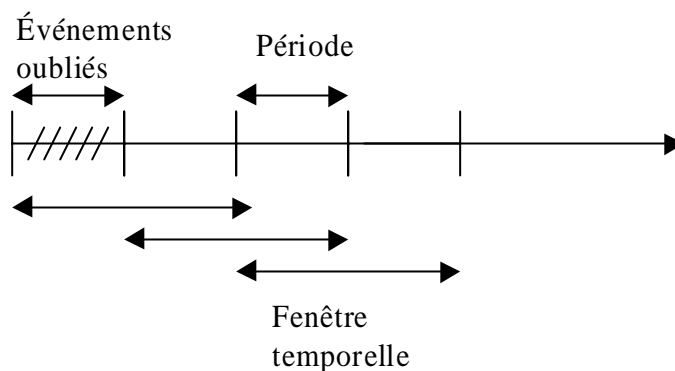


Figure 14 : Simulation de la fenêtre temporelle

4.3 *Modèle conceptuel de traitement des données*

Dans le modèle conceptuel de traitement de données, nous décrivons le langage descriptif, le langage symbolique et l'algorithme de traitement des événements de sécurité.

4.3.1 Le langage descriptif

Dans cette section, nous décrivons les différents opérateurs du langage descriptif. Nous définissons un ensemble d'opérateurs afin d'exprimer les critères de sélection des événements et suites d'événements. Nous distinguons deux types d'opérateurs :

1. les *opérateurs atemporels* qui portent sur :
 - ◆ un *événement* pour exprimer les critères de sélection de l'événement. Ces opérateurs permettent de spécifier des valeurs d'attributs de l'événement ;
 - ◆ une *suite d'événements* pour
 - exprimer les critères de corrélation des événements pour la sélection des suites d'événements. Ce sont des opérateurs de comparaison entre les valeurs d'attributs de plusieurs événements ;
 - exprimer les conditions de reconnaissance d'un schéma d'attaque. Ces opérateurs permettent d'exprimer l'*existence*, l'*itération*, la *conjonction*, la *disjonction* et la *négation* des événements de la suite ;
2. les *opérateurs temporels* qui portent sur une suite d'événements. Ce sont des opérateurs qui permettent d'exprimer des contraintes temporelles sur une suite d'événements tels que l'*ordre temporel* et la *périodicité*.

Comme nous l'avons dit plus haut, à chaque opérateur est associée une fonction d'évaluation. Cette fonction retourne deux valeurs *Vrai* ou *Faux*. Il y a deux types d'évaluation :

1. dans le cas des opérateurs portant sur un événement, chaque opérateur porte sur un ou plusieurs attributs. Si l'attribut remplit la condition relative à l'opérateur alors la fonction retourne *Vrai* sinon elle retourne *Faux*. Supposons que nous voulons collecter des "logins" dont la source est égale à "Site A". L'opérateur portera sur l'attribut source, qui lui spécifiera la valeur "Site A". Dans ce cas la fonction d'évaluation évaluera l'attribut source de chaque *message* pour tester si elle est égale à " Site A ". Si c'est le cas, elle retournera *Vrai* et le message sera collecté ;
2. dans le cas des opérateurs portant sur une suite d'événements, l'évaluation se fait différemment. Prenons l'exemple de l'opérateur d'*itération* que nous appliquons sur une suite d'événements de type "logins" provenant d'une source "Site A". La fonction d'évaluation retournera *Vrai* si le nombre d'événements de la suite est égal au nombre spécifié par l'opérateur d'*itération*.

4.3.1.1 Les opérateurs atemporels portant sur un événement

Ces opérateurs permettent de spécifier des valeurs d'attributs qu'un événement devrait avoir. Lorsqu'il n'existe pas d'opérateur de ce type sur un attribut, cela signifie qu'aucune valeur n'est spécifiée et que l'attribut peut prendre n'importe quelle valeur. Il existe quatre opérateurs : *égal*, *appartient*, *différent* et *n'appartient pas*.

1. *Égal* permet de spécifier une valeur précise à un attribut **a** d'un événement **e**.

<attribut a >**égal** <valeur V> : signifie $e.a = V$

2. *Appartient* permet de spécifier un ensemble de valeurs possibles pour un attribut.

<attribut a >**appartient** <ensemble de valeurs V_{ens} > : signifie $e.a \in V_{ens}$

3. *Différent* est un opérateur de négation qui permet d'exclure une valeur précise d'attribut.

<attribut a >**différent** <valeur V> : signifie $e.a \neq V$

4. *N'appartient pas* permet d'exclure un ensemble de valeurs pour un attribut.

<attribut a >**n'appartient pas**<ensemble de valeurs V_{ens} > : signifie $e.a \notin V_{ens}$

5. *Indifférent* signifie que l'attribut peut prendre n'importe quelle valeur.

<attribut a > **indifférent** : signifie $e.a =$ n'importe quelle valeur.

4.3.1.2 Opérateurs portant sur une suite d'événements

Comme nous l'avons dit précédemment, ces opérateurs permettent d'exprimer l'ordre temporel ou non des événements. Nous définissons la notion de contraintes temporelles qui permettent de spécifier la durée d'observation d'une suite d'événements, dite fenêtre temporelle d'observation et qui est notée ∇_s

4.3.1.2.1 Opérateurs atemporels

Nous distinguons deux types d'opérateurs atemporels : les *opérateurs atemporels portant sur les attributs de plusieurs événements* et les *opérateurs portant sur une suite d'événements*.

Les opérateurs atemporels portant sur les attributs de plusieurs événements

Ces opérateurs définissent les critères de corrélation des événements. Ils permettent d'exprimer l'égalité ou non entre les attributs de plusieurs événements. Nous définissons quatre opérateurs : **même**, **différent_{att}**, **n'importe quel** et **tous**.

1. **Même** permet d'exprimer l'égalité par rapport à un attribut **a** de plusieurs événements.

Même (e_1, \dots, e_n, a) : signifie $e_1.a = e_2.a = \dots = e_n.a$

2. **Différent_{att}** permet d'exprimer la différence par rapport à un attribut **a** de deux événements.

Différent_{att} (e_1, \dots, e_n, a) : signifie $e_1.a \neq e_2.a \neq \dots \neq e_n.a$

3. **N'importe quel** signifie qu'un attribut **a** de deux événements peut être le même ou différent.

N'importe quel $(e_1, \dots, e_n, a) : \equiv \text{même}(e_1, \dots, e_n, a) \vee \text{différent}_{\text{att}}(e_1, \dots, e_n, a)$

Les opérateurs atemporels portant sur une suite d'événements

Ces opérateurs permettent d'exprimer l'existence, l'itération, la conjonction, la disjonction et la négation.

Existence

L'opérateur **Exist** permet d'exprimer l'existence d'une occurrence d'événement décrit par un filtre d'événement d_e . Il est défini par la fonction **Exist** (e, d_e, ∇_s) qui signifie :

sachant $d_e, \exists e \in E \mid$

$e.\text{eventType}$	égal	$d_e.\text{securityEvent.eventType}$	\wedge
$e.\text{eventName}$	égal	$d_e.\text{securityEvent.eventName}$	\wedge
$e.\text{eventSource}$	égal	$d_e.\text{securityEvent.eventSource}$	\wedge
$e.\text{eventDestination}$	égal	$d_e.\text{securityEvent.eventDestination}$	\wedge
$e.\text{eventResult}$	égal	$d_e.\text{securityEvent.eventResult}$	\wedge
$e.\text{eventActivity}$	égal	$d_e.\text{securityEvent.eventActivity}$	\wedge
$e.\text{securityEvent.eventTime} \in \nabla_s$			

E est l'ensemble des événements se produisant dans le réseau. Le filtre d'événement d_e est défini par : $d_e.a_1 = V, d_e.a_2 \in V_{\text{ens1}}, d_e.a_3 \in V_{\text{ens2}}, \text{etc.}$

Itération

L'opérateur d'itération permet d'exprimer la fréquence de répétition d'une occurrence d'événement dans la fenêtre temporelle ∇_s . Cet opérateur est défini par la fonction **Itération** (e_j, d_e, n, ∇_s) qui signifie :

sachant $d_e, \exists e_1, e_2, \dots, e_n \in E \mid$

$\forall j=1..n$	$e_j.\text{eventType}$	égal	$d_e.\text{securityEvent.eventType}$	\wedge
	$e_j.\text{eventName}$	égal	$d_e.\text{securityEvent.eventName}$	\wedge
	$e_j.\text{eventSource}$	égal	$d_e.\text{securityEvent.eventSource}$	\wedge
	$e_j.\text{eventDestination}$	égal	$d_e.\text{securityEvent.eventDestination}$	\wedge
	$e_j.\text{eventResult}$	égal	$d_e.\text{securityEvent.eventResult}$	\wedge
	$e_j.\text{eventActivity}$	égal	$d_e.\text{securityEvent.eventActivity}$	\wedge
	$e_j.\text{eventTime} \in \nabla_s$			

Conjonction

La conjonction permet d'exprimer l'existence de deux événements sans préciser l'ordre temporel. Elle est définie par la fonction **Conjonction** $(e_1, e_2, d_{e1}, d_{e2}, \nabla_s)$ qui signifie :

$$\text{exist}(e_1, d_{e1}, \nabla_s) \wedge \text{exist}(e_2, d_{e2}, \nabla_s)$$

Disjonction

La disjonction permet d'exprimer l'existence de l'un ou l'autre de deux occurrences d'événements sans préciser l'ordre temporel. Elle est définie par la fonction **Disjonction** $(e_1, e_2, d_{e1}, d_{e2}, \nabla_s)$ qui signifie :

$$\text{exist}(e_1, d_{e1}, \nabla_s) \vee \text{exist}(e_2, d_{e2}, \nabla_s)$$

Négation

La négation permet d'exprimer, dans une suite d'événements, l'existence de l'occurrence d'un événement défini par d_{e_1} mais pas de l'occurrence d'un événement défini par d_{e_2} . Il n'y a donc pas d'ordre temporel. Elle est définie par la fonction **Négation** ($e_1, e_2, d_{e_1}, d_{e_2}, \nabla_s$) qui signifie :

$$\text{exist}(e_1, d_{e_1}, \nabla_s) \wedge \neg \text{exist}(e_2, d_{e_2}, \nabla_s)$$

Opérateurs temporels

Ces opérateurs permettent d'exprimer l'ordre temporel à l'aide d'opérateurs de séquence et de périodicité.

Séquentialité

L'opérateur de séquentialité permet d'exprimer la succession ou la précédence de deux occurrences d'événements dans la fenêtre temporelle ∇_s . Cet opérateur est défini par les fonction **Suit** ($e_1, e_2, d_{e_1}, d_{e_2}, \nabla_s$) et **Précède** ($e_1, e_2, d_{e_1}, d_{e_2}, \nabla_s$).

Suit ($e_1, e_2, d_{e_1}, d_{e_2}, \nabla_s$) signifie :

$$\text{exist}(e_1, d_{e_1}, \nabla_s) \wedge \text{exist}(e_2, d_{e_2}, \nabla_s) \wedge e_1.\text{eventTime} > e_2.\text{eventTime}$$

Précède ($e_1, e_2, d_{e_1}, d_{e_2}, \nabla_s$) signifie :

$$\text{exist}(e_1, d_{e_1}, \nabla_s) \wedge \text{exist}(e_2, d_{e_2}, \nabla_s) \wedge e_1.\text{eventTime} < e_2.\text{eventTime}$$

Périodicité

Cet opérateur permet d'exprimer la répétition d'une occurrence d'événement toutes les t_x périodes. Il est défini par la fonction **Tous les** ($e_j, d_e, n, t_x, \nabla_s$) qui signifie :

sachant d_e et sachant $t_x, \exists e_1, e_2, \dots, e_n \in E$ |

$\forall j \in \{1, \dots, n\}$	$e_j.\text{eventType}$	égal	$d_e.\text{securityEvent.eventType}$	\wedge
	$e_j.\text{eventName}$	égal	$d_e.\text{securityEvent.eventName}$	\wedge
	$e_j.\text{eventSource}$	égal	$d_e.\text{securityEvent.eventSource}$	\wedge
	$e_j.\text{eventDestination}$	égal	$d_e.\text{securityEvent.eventDestination}$	\wedge
	$e_j.\text{eventResult}$	égal	$d_e.\text{securityEvent.eventResult}$	\wedge
	$e_j.\text{eventActivity}$	égal	$d_e.\text{securityEvent.eventActivity}$	\wedge
	$e_j.\text{eventTime} \in \nabla_s$			\wedge
$\forall j \in \{1, \dots, n-1\}$	$e_{j+1}.\text{eventTime}$	égal	$e_j.\text{eventTime} + t_x$	

En terme d'implémentation, t_x n'est pas une valeur précise mais une valeur qui varie dans un intervalle $[t_x - \epsilon, t_x + \epsilon]$.

4.3.1.2.2 Opérateurs arithmétiques

= : permet de formaliser le nombre d'itérations d'un événement et le niveau de suspicion

< > : permet de borner les seuils d'alarmes

- : est utilisé pour les relations temporelles

!= : permet d'exprimer une différence entre deux attributs X_i

% : permet d'exprimer des taux (occupation mémoire, congestion, paquets, ...)

() : permet d'encadrer des expressions

[] : permet d'exprimer un intervalle de valeurs

4.3.1.2.3 Opérateurs logiques

non, et, ou : sont utilisés pour les relations entre attributs

{ } : sont utilisés pour formaliser un ensemble

$\in, !\in$: sont utilisés pour exprimer l'appartenance ou non à un ensemble

4.3.1.2.4 Opérateur conditionnel

If : permet d'exprimer des conditions sur les attributs et événements.

4.3.2 Le Langage symbolique

Nous avons souligné, lors de la description de la classe schéma d'attaque, que la fonction d'extraction et d'analyse des suites étaient assez complexes. Pour cela, nous avons pensé définir un langage, dit *symbolique*, pour faciliter le traitement et l'implémentation de notre modèle de gestion des événements de sécurité.

Le langage symbolique permet de représenter de manière simple et visuelle les différents éléments et opérateurs du langage décrit dans les sections précédentes. Ce langage utilise un certain nombre de notations et symboles qui sont : *, >, <, \cap , \cup , $\bar{}$, V_{SP} et V_{EX} . Chacun de ces symboles sera explicité au fur et à mesure qu'il sera introduit pour représenter un élément ou opérateur du langage. Nous utiliserons également la notion de vecteur pour représenter une séquence d'événements dans une fenêtre temporelle V_s .

4.3.2.1 Tableau de représentation

Une implémentation possible des différents éléments et opérateurs est d'utiliser un tableau de représentation à trois dimensions. Initialement, nous avons utilisé ce tableau pour représenter une instance de schéma d'attaque. Il a fallu d'abord représenter un filtre d'événement avec ses opérateurs. Puis un schéma d'attaque avec les opérateurs atemporels et temporels.

Une ligne du tableau d_{ei} représente un filtre d'événements. Une colonne a_i représente un attribut du filtre d'événement. L'intersection entre une colonne et une ligne représente une valeur spécifiée ou non d'un attribut d'un filtre d'événement. Pour ce qui est de la troisième dimension, elle permet de représenter la fréquence de l'opérateur d'itération mais nous le verrons dans le paragraphe 4.3.2.2.2.

Dans ce tableau, nous distinguons :

- une *partie statique* pour représenter un schéma d'attaque. Elle est définie par les lignes d_{ei} , dev_x et dev_z (qui représentent les différents opérateurs et que nous définirons dans la section 4.3.2.2.3),
- une *partie dynamique* pour représenter les instances de ce schéma d'attaque. Elle est définie par les lignes dev_{zi} , que nous définirons dans la section 0).

Attributs	a ₁	a ₂	a ₃	a ₄
Filtres d'événements				
d _{e1}	op _{att}	op _{att}	op _{att}	op _{att}
d _{e2}	op _{att}	op _{att}	op _{att}	op _{att}
dev _x	expr	expr	expr	expr
dev _z	op _{ev}	op _{ev}	op _{ev}	op _{ev}
dev _{z1}	V _{SP1}		V _{SP2}	
dev _{z2}	V _{SP3}		V _{SP4}	

4.3.2.2 Représentation des différents opérateurs

Lors de la conception de ce langage symbolique, nous nous sommes posés la question de savoir comment exploiter le tableau précédent pour représenter les différents opérateurs, dans le but :

- d'instancier les schémas d'attaque,
- et par conséquent de pouvoir traiter les événements et suites d'événements.

4.3.2.2.1 Représentation des opérateurs atemporels portant sur un événement

Les opérateurs atemporels sur événement sont représentés, à l'intersection entre un filtre d'événement et un attribut, à l'aide de trois symboles : V_{SP}, V_{EX} et *.

V_{SP} est utilisé pour les opérateurs : *égal* et *appartient*

V_{EX} est utilisé pour les opérateurs : *différent* et *n'appartient pas*

* est utilisé pour l'opérateur : *indifférent*.

Les deux symboles V_{SP} et V_{EX} sont suivis des valeurs spécifiées ou à exclure, qui peuvent être soit une valeur précise ou un ensemble de valeurs.

Attributs	a ₁	a ₂	a ₃	a ₄
Filtres d'événements				
d _{e1}	op_{att}	op_{att}	op_{att}	op_{att}
d _{e2}	op_{att}	op_{att}	op_{att}	op_{att}

op_{att} := V_{SP} | V_{EX} | *

4.3.2.2.2 Représentation des opérateurs d'itération et de périodicité

L'opérateur d'itération est un opérateur de comptage. Pour le représenter, nous rajoutons une autre dimension dans le tableau pour représenter le nombre de répétitions d'un événement N_{ev} et d'une séquence d'événements N_{seq}.

Une séquence d'événements se produisant dans le réseau sera représentée par un vecteur V_{ect} qui contient le nombre d'occurrences par type d'événement de la

séquence. Puis sur ce vecteur est effectuée une opération de comptage. Nous verrons cela un peu plus tard dans la section 4.3.3.3

	Attributs	a_1	a_2	a_3	a_4	t_x	
	Filtres d'événements						
N	d_{e1}	op _{att}	op _{att}	op _{att}	op _{att}	x_1	N
	d_{e2}	op _{att}	op _{att}	op _{att}	op _{att}	x_2	N

Pour ce qui est de l'opérateur de périodicité, nous ajoutons une colonne pour représenter la période de répétition d'un événement.

4.3.2.2.3 Représentation des opérateurs atemporels portant sur une suite d'événements et des autres opérateurs temporels

Pour représenter les opérateurs de *conjonction*, *disjonction* et *négation* nous utiliserons les symboles suivants : \cap , \cup , \neg .

- \cap est utilisé pour représenter l'opérateur : *et*
- \cup est utilisé pour représenter l'opérateur : *ou*
- \neg est utilisé pour représenter l'opérateur : *non*

Pour ce qui est des opérateurs qui font référence à l'ordre temporel, nous allons utiliser deux symboles : $<$, $>$.

- $<$ est utilisé pour représenter l'opérateur : *précède*
- $>$ est utilisé pour représenter l'opérateur : *suit*

Pour l'ensemble de ces opérateurs, nous rajoutons dans la table de l'instance d'un schéma, une ligne qui caractérise une séquence. Nous créons un nouveau filtre d'événement de type dev_x . Par exemple $dev_x: 2 >(1 \cup 3)$, cela signifie que l'événement 2 doit se produire après l'événement 1 ou l'événement 3

	Attributs	a_1	a_2	a_3	a_4
	Filtres d'événements				
	d_{e1}	op _{att}	op _{att}	op _{att}	op _{att}
	d_{e2}	op _{att}	op _{att}	op _{att}	op _{att}
	dev_x	<i>expression</i>			

expression := $d_{e1} op d_{e2}$
op := $\cap | \cup | \neg | < | >$

4.3.2.2.4 Représentation des opérateurs portant sur les attributs de plusieurs événements

Pour représenter les opérateurs sur attributs entre événements (*même*, *différent_{att}*, et *n'importe quel*), nous utilisons les symboles \neq , $=$, $*$, V_{SPi} (pour une valeur spécifiée). Nous rajoutons une ligne dans le tableau et créons un nouveau filtre d'événement noté dev_z . Cette ligne sera modifiée dynamiquement et il peut y en avoir plusieurs. La dynamique de cette ligne est due au fait qu'elle sera modifiée en fonction des événements observés qui auront des valeurs d'attributs réelles lorsque celles-ci n'auront pas été spécifiées dans l'instance de schéma. Par exemple, si l'instance de schéma est de reconnaître tous les événements de type "telnet" ayant même adresse source, cette adresse source ne sera, dans un premier temps, pas spécifiée. Lorsque les événements seront collectés, ils seront regroupés suivant la même adresse source et dans ce cas nous aurons plusieurs adresses source. L'attribut "adresse source" sera alors spécifié et nous aurons à remplir plusieurs lignes du tableau de type dev_{zi} . Dans ce cas, nous aurons plusieurs instances de schémas, une instance par ligne dev_{zi} .

$=$ est utilisé pour l'opérateur : *même*

\neq est utilisé pour l'opérateur : *différent_{att}*

$*$ est utilisé pour l'opérateur : *n'importe quel*.

V_{SPi} est utilisé pour instancier une valeur d'attribut.

Attributs	a_1	a_2	a_3	a_4
Filtres d'événements				
d_{e1}	op_{att}	op_{att}	op_{att}	op_{att}
d_{e2}	op_{att}	op_{att}	op_{att}	op_{att}
dev_x	expression			
dev_z	op_{ev}	op_{ev}	op_{ev}	op_{ev}
dev_{z1}	V_{SP1}	$*$	V_{SP2}	\neq
dev_{z2}	V_{SP3}	$*$	V_{SP4}	\neq

$op_{ev} := * | = | \neq$

V_{SP1} est égale par exemple à l'adresse source "Lip6" et V_{SP2} à l'adresse destination "Eurécom"

4.3.3 Traitement des événements pour la reconnaissance de schémas d'attaque

Dans cette section, nous allons voir comment les différents opérateurs seront utilisés pour le filtrage et la reconnaissance de schémas d'attaque. A partir des observations effectuées sur les événements de sécurité qui se produisent dans le réseau, une opération de reconnaissance de schémas d'attaque est réalisée afin de voir si l'ensemble des événements vus dans la fenêtre temporelle V_{ts} correspond à un schéma d'attaque.

La reconnaissance des schémas d'attaque est une opération globale qui se décompose en quatre opérations (voir Figure 15) :

- opération de *sélection d'événements*,
- opération de *sélection de suites d'événements*,
- opération de *comptage*,
- opération de *reconnaissance de schéma d'attaque*.

Ces quatre couches de traitement d'événements sont présentées dans les sous-sections suivantes.

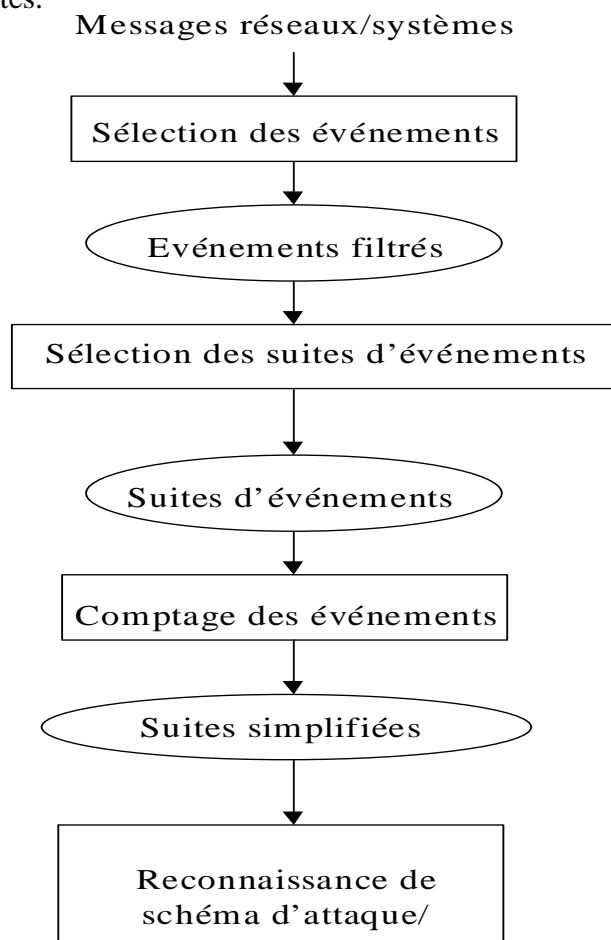


Figure 15 : Processus de traitement des événements

4.3.3.1 Sélection des événements

Dans ce premier niveau de filtrage, nous utilisons les opérateurs de comparaison définis dans les instances de filtres d'événements de chaque schéma d'attaque. Ces opérateurs, rappelons-le, définissent les critères de sélection des événements. Cette étape de filtrage permet ainsi la sélection des événements par rapport aux instances de filtres d'événements. En effet, cette première opération sélectionne pour une instance de schéma donné, les événements ayant les valeurs V_{SP} spécifiées dans le tableau

représentatif du schéma. Ce niveau de filtrage utilise les critères de sélection d'événements pour extraire les événements parmi l'ensemble des messages réseaux et systèmes.

L'opération de sélection porte sur les types des événements et valeurs d'attributs spécifiés dans les filtres d'événements. Le filtrage est effectué attribut par attribut pour des raisons de simplicité. Le but de cette opération est de limiter le nombre d'événements et de simplifier le filtrage pour les autres couches.

Opération de sélection = \exists une séquence de $\{ev_i, \text{fenêtre temporelle d'observation}\}$ dont les attributs ont des valeurs égales à celles spécifiées dans l'instance de schéma.

Pour réaliser ce filtrage, nous avons deux options possibles :

1. définir un seul filtre (voir Figure 16) qui est déterminé par tous les critères de sélection de tous les schémas d'attaque. Soit C_i , les critères de sélection des événements correspondant à un schéma d'attaque. Nous définissons un filtre F dont le critère de filtrage C est défini par : $C = \cup C_i$.
A la sortie du filtre, nous obtenons un seul ensemble d'événements. La sélection des suites d'événements se fera ensuite sur cet ensemble.
2. des filtres séparés (voir Figure 17) et donc un filtre par schéma d'attaque. A la fin de ce filtrage, nous aurons n ensembles d'événements. La sélection des suites d'événements se fera sur tous les ensemble d'événements.

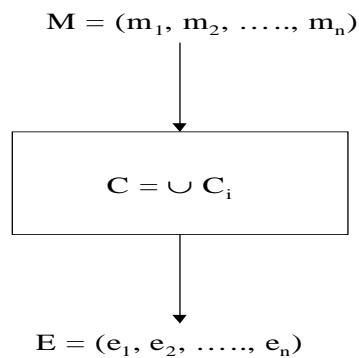


Figure 16 : Première option de filtrage - un seul filtre

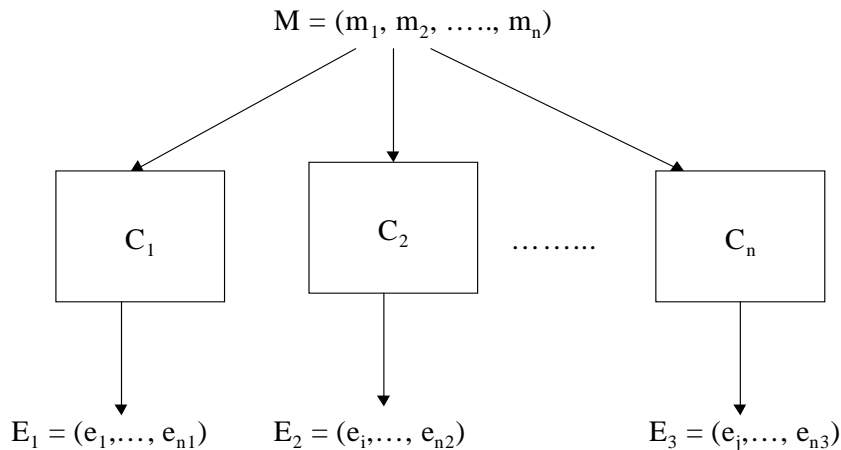


Figure 17 : Deuxième option de filtrage – des filtres séparés

Solution retenue

La première solution a l'avantage de définir un seul filtre qui a comme entrée tous les messages et en sortie un seul ensemble d'événements. L'inconvénient est que pour l'étape suivante, il faudrait, d'abord filtrer les sous-ensembles d'événements qui référencent le même schéma d'attaque.

En ce qui concerne la deuxième solution, l'avantage concerne la couche suivante de traitement où nous n'avons pas besoin de régénérer des sous-ensembles d'événements. Ces sous-ensembles ayant déjà été obtenus en sortie des différents filtres. L'inconvénient est que nous aurions à gérer plusieurs filtres.

Notre souci est de simplifier le traitement des événements pour les couches suivantes, en particulier celle de sélection des suites d'événements. Par conséquent, notre choix s'est porté sur la deuxième option.

4.3.3.2 Sélection des suites d'événements

Ce niveau de filtrage utilise les critères de sélection des suites d'événements pour extraire les suites d'événements corrélées parmi l'ensemble des événements collectés dans la fenêtre temporelle d'observation.

Pour cette étape, nous partons de l'hypothèse que nous avons choisi la deuxième solution, à l'étape précédente. A partir de chaque ensemble d'événements référencant la même instance de schéma d'attaque, sont sélectionnées des suites d'événements.

Dans cette étape, nous utilisons les opérateurs de comparaison entre attributs des différents événements. Ces opérateurs permettent de corréler les événements par rapport à des attributs spécifiés par les opérateurs.

Prenons l'exemple de l'opérateur *<même>* appliqué à l'attribut *source* pour exprimer que nous voulons tous les événements qui ont la même source. Dans ce cas, nous n'avons pas de valeur pour l'attribut source mais nous allons regrouper les événements par source identique (voir Figure 18). Supposons que nous avons des événements provenant de la source "SITE A" et de la source "SITE B". Nous aurons une suite avec tous les événements provenant de " SITE A " et une autre suite avec ceux provenant de " SITE B". Le traitement de cet opérateur entraîne la création de nouvelles instances de schémas d'attaque ayant des valeurs de l'attribut source différentes. Pour traduire cela au niveau du tableau représentant le schéma d'attaque, nous allons rajouter de nouvelles lignes dev_{zi} (voir 4.3.2.2.4). Chaque ligne correspondra à une instance de schéma d'attaque.

Opération de sélection des suites : générer des suites d'événements {événements du 1er filtrage} par instance de schéma d'attaque après avoir spécifié les valeurs d'attributs réelles non spécifiées dans l'instance de schéma.

Une opération de comptage est ensuite déclenchée sur les suites d'événements.

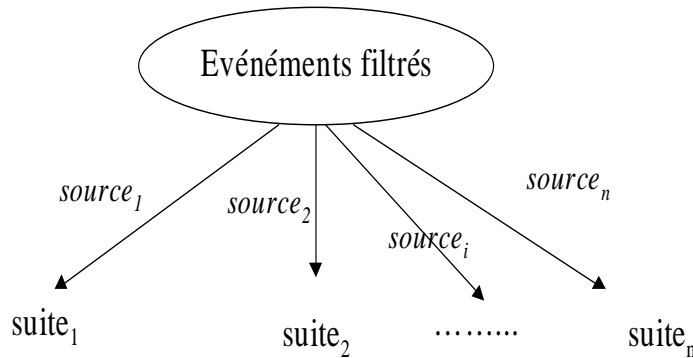


Figure 18 : Filtrage des suites d'événements par rapport à l'attribut *source*

4.3.3.3 Comptage des événements

Cette étape a pour but de simplifier le traitement des opérateurs d'itération et de périodicité. Ce que nous voulons faire c'est compter le nombre d'événements par type d'événements pour générer des suites simplifiées ; d'où le champ *<résumé>* introduit dans 4.2.3. Chaque suite d'événements est synthétisée par un vecteur V_{ect} , qui compte le nombre d'événements par type.

$$V_{\text{ect}} = (n_{e1}, n_{e2}, \dots, n_{en}).$$

Lors de cette phase, sont créés plusieurs vecteurs. Il y aura un vecteur par instance de schéma d'attaque.

En ce qui concerne l'opérateur de périodicité, l'aspect temporel n'est pas pris en compte dans le vecteur tel que nous l'avons défini. Pour cela, il nous faut considérer la période spécifiée par l'opérateur. Dans ce cas, nous créons un vecteur particulier, dit *périodique* où seront comptés pour chaque type d'événements, les événements produits durant la période spécifiée par l'opérateur.

Opération de comptage : créer les vecteurs à partir des {suites 2ème filtrage} en comptant le nombre d'événements par type d'événements et par suite.

4.3.3.4 Reconnaissance de schéma d'attaque

Cette étape finale porte sur l'ensemble des suites d'événements sélectionnées. Elle permet de voir si les suites d'événements filtrées par rapport aux instances de schémas correspondent à un schéma d'attaque.

Dans cette couche de traitement, nous utilisons les opérateurs temporels et les opérateurs atemporels sur suite d'événements (voir 4.3.1.2). Ce sont les opérateurs symbolisés dans les lignes dev_x (voir 4.3.2.2.3) du tableau représentant un schéma d'attaque.

La fonction de reconnaissance de schéma d'attaque est un *ET* logique entre les opérateurs définis dans une ligne dev_x . Chaque opérateur est alors évalué et si la suite ne remplit pas les conditions d'un des opérateurs, la fonction retourne *faux*, signifiant que la suite courante n'est pas une attaque.

Pour traiter les opérateurs d'itération et de périodicité, nous utilisons les vecteurs créés lors de l'étape précédente et nous comparons les valeurs du vecteur avec la (les) fréquence(s) spécifiée(s) par les opérateurs.

Pour les autres opérateurs, la suite est évaluée par chaque opérateur défini dans la ligne dev_x . Prenons par exemple l'opérateur de séquence $\langle suit \rangle$ appliqué sur deux types d'événements e_1 et e_2 de la manière suivante : $e_1 \langle suit \rangle e_2$. La fonction d'évaluation de cet opérateur aura en entrée la suite d'événements et vérifiera si dans cette suite, il existe une ou plusieurs séquences $e_1 e_2$.

Opération de reconnaissance de schéma d'attaque = \exists une séquence dans {séquences de 3ème filtrage, dans fenêtre}, caractérisée par les lignes dev_x .

4.4 Conclusion

Dans ce chapitre nous avons présenté notre modèle de gestion des événements de sécurité. Nous avons décrit les différents éléments du modèle :

- les messages et les événements de sécurité,
- les filtres d'événements,
- les suites d'événements,
- les schémas d'attaque.

Puis nous avons présenté le langage descriptif qui définit un ensemble d'opérateurs pour exprimer :

- les critères de sélection des événements ;
- les critères de corrélation des événements pour l'extraction des suites d'événements ;
- les conditions de reconnaissance des schémas d'attaque.

Nous avons ensuite proposé une façon d'implémenter les différents éléments du modèle et les opérateurs, en utilisant une représentation par tableau. Nous avons ainsi défini un langage symbolique qui facilite l'implémentation du langage descriptif.

Finalement, nous avons décrit l'algorithme de traitement des événements pour la reconnaissance de schémas d'attaque.

Nous verrons, dans le chapitre suivant, comment sont instanciés les schémas d'attaque, qui représentent l'élément clé de notre modèle de gestion des événements.

Chapitre 5 Gestion des politiques de sécurité

5.1 Introduction

Le but de ce chapitre est de décrire le *plan utilisateur* de notre modèle de gestion de sécurité. Ce plan implémente les politiques de sécurité qui traduisent les règles et procédures définissant la sécurité d'un réseau d'entreprise. Les politiques de sécurité ont un rôle majeur dans notre modèle car elles représentent l'entité directrice qui va influencer le comportement de notre système multi-agents. En effet, elles permettent de spécifier :

- les profils d'attaques à détecter,
- la conduite à tenir en cas d'attaque.

Les politiques de sécurité interviennent à trois niveaux (voir Figure 19). Elles permettent :

1. de sélectionner et d'instancier les schémas d'attaque à détecter ;
2. de créer les buts à envoyer aux différents agents concernés pour détecter ces schémas d'attaque ;
3. de sélectionner les événements pertinents à filtrer pour reconnaître les schémas d'attaque instanciés.

Après avoir identifié le rôle des politiques de sécurité, nous allons les représenter et les gérer. La gestion des politiques de sécurité n'est pas une tâche facile. En effet, de nombreux travaux ont été effectués sur les politiques de gestion en générale [Lupu et al. 1995] [Lupu et al. 1999] [Lupu and Sloman 1997a] [Lupu and Sloman 1997b] [Lupu and Sloman 1997c] [Lupu and Sloman 1997d] [Mariott 1997] et de sécurité en particulier [Lupu et al. 1997] [Yialelis and Sloman 1995b] [Yialelis 1996]. De ces travaux sont nés différents points de vue quant à la façon de les représenter et de les intégrer dans un système informatique [Mariott 1997]. Généralement, les politiques de sécurité sont spécifiées en langage naturel, ce qui rend difficile la vérification de la

conformité de l'implémentation des procédures aux politiques spécifiées. Pour cela, nous nous sommes intéressés aux travaux existants pour pouvoir représenter les politiques de sécurité dans notre système.

Dans ce chapitre, nous allons donc commencer par définir les politiques de gestion de sécurité. Puis nous présenterons les modèles proposés pour représenter et formaliser les politiques de gestion en général. Enfin nous décrirons notre propre modèle.

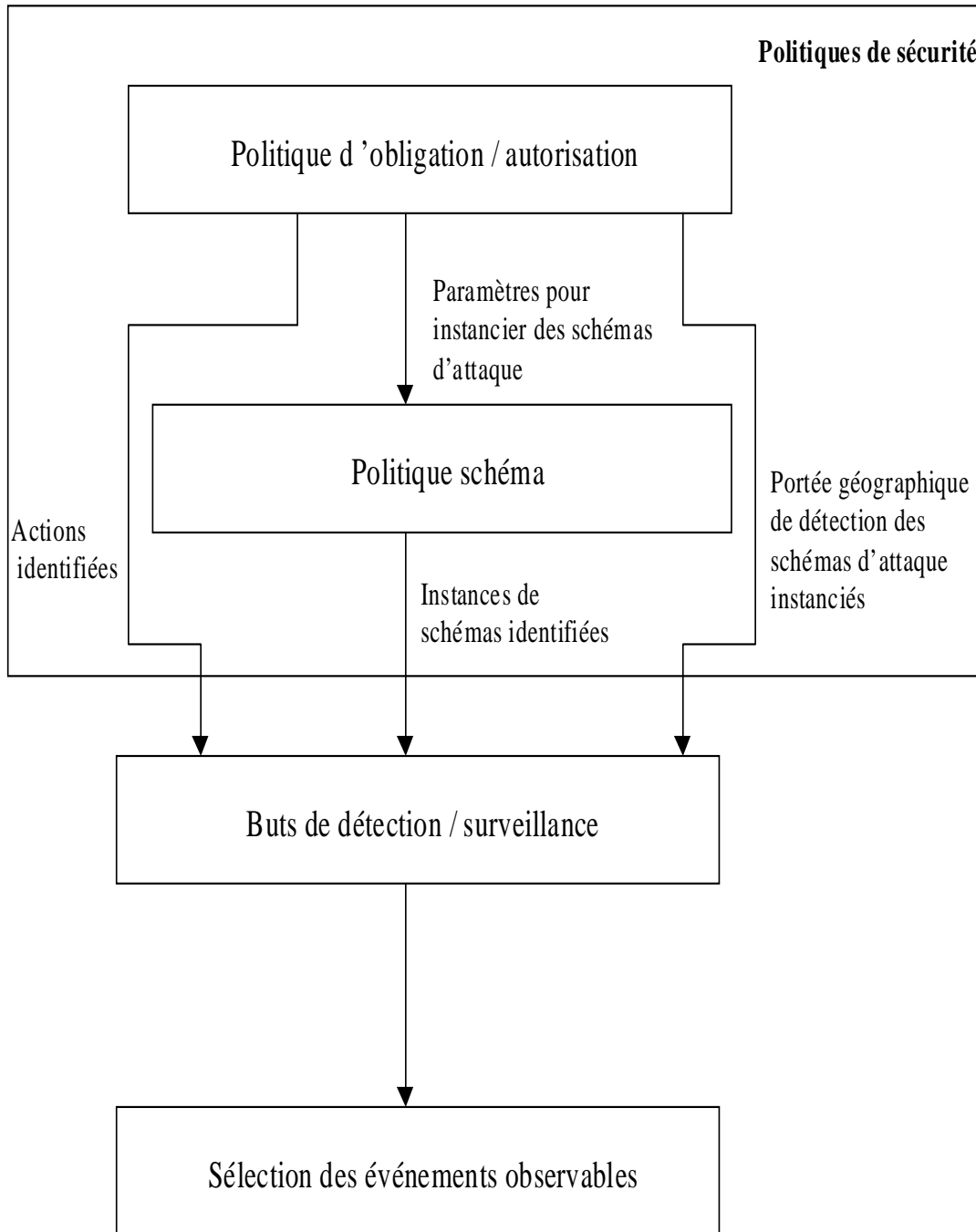


Figure 19 : Rôle d'une politique de sécurité

5.2 Définition d'une politique de gestion de sécurité

Les politiques de gestion en général et de sécurité en particulier, sont un concept très large et dans la littérature plusieurs définitions sont données à ce terme :

Définition de Sloman : Les politiques définissent la stratégie de gestion globale d'un système qui influence son comportement [Lupu et al. 1995] [Lupu et al. 1997] [Lupu et al. 1999] [Lupu and Sloman 1997a] [Lupu and Sloman 1997b] [Lupu and Sloman 1997c] [Lupu and Sloman 1997d].

Définition de Wies : les politiques sont dérivées des objectifs de gestion et définissent le comportement désiré des réseaux et systèmes hétérogènes distribués. Elles sont utilisées pour définir et contrôler le comportement des ressources d'un système [Wies 1995a][Wies 1995b] [Wies et al. 1997].

En règle générale, une politique de sécurité traduit l'ensemble des procédures et règles de sécurité d'une organisation [Walker and Cavanaugh 1998]. Elle doit définir comment les utilisateurs internes et externes doivent et peuvent interagir avec le réseau d'entreprise [Walker and Cavanaugh 1998].

A la définition d'une politique de gestion est souvent associée la notion de *domaine* et de *rôle*.

5.2.1 Notion de Domaine

Un *domaine* représente une collection d'objets (utilisateurs, fichiers, serveurs, stations de travail, etc.) qui ont été explicitement regroupés pour des buts de gestion [Yialelis and Sloman 1995a] [Yialelis and Sloman 1995b] [Yialelis 1996]. Les domaines permettent de regrouper les objets sur lesquels doivent s'appliquer les politiques. Ils peuvent être utilisés pour répartir les objets dans un large système en fonction de leur situation géographique, leur type ou à la convenance de l'administrateur [Lupu et al. 1997] [Damianou et al. 2000]. Les domaines ajoutent une flexibilité aux politiques en permettant de rajouter ou supprimer des objets dans un domaine sans changer la spécification de la politique [Damianou et al. 2000].

5.2.2 Notion de Rôle

Un *rôle* définit les devoirs et droits d'une position dans une organisation (directeur financier, gestionnaire de personnel, administrateur,...). Spécifier une politique en terme de rôles au lieu de personnes évite de redéfinir une politique lorsqu'un nouveau rôle est assigné à une personne [Lupu et al. 1995] [Lupu et al. 1997] [Lupu et al. 1999] [Lupu and Sloman 1997a] [Lupu and Sloman 1997b] [Lupu and Sloman 1997c] [Lupu and Sloman 1997d] [Yialelis et al. 1996]

5.2.3 Types de politiques de sécurité

Dans la littérature [Sloman 1993] [Mariott and Sloman 1996] [Steenekamp and Roos 1996], il existe plusieurs types de politiques, néanmoins, dans le cadre de ce travail, nous avons retenu deux types de politiques : les politiques d'*obligation* et les politiques d'*autorisation*.

5.2.3.1 Les politiques d'obligation

Les politiques d'obligation spécifient les actions qu'un *sujet* doit faire ou ne doit pas faire sur un ensemble d'objets *cibles* [Ponder 99]. Ces politiques sont déclenchées par des événements. On parle d'*obligation positive* (O₊) et d'*obligation négative* (O₋).

5.2.3.2 Les politiques d'autorisation

Les politiques d'autorisation sont essentiellement des politiques de sécurité de contrôle d'accès. Elles définissent ce que peut ou ne peut pas faire une source sur plusieurs cibles. On parle d'*autorisation positive* (A₊) et d'*autorisation négative* (A₋).

5.3 Hiérarchie et raffinement d'une politique de sécurité

Les politiques de gestion peuvent exister à différents niveaux d'abstraction [Marriott and Samani 1996] [Marriott 1997]. Par exemple, les politiques de gestion qui sont implémentées dans un système distribué sont une conséquence de politiques organisationnelles de haut niveau. Le concept de *hiérarchie de politique* définit différents niveaux d'abstraction pour les politiques. Le *raffinement d'une politique* signifie transformer les objectifs et les buts de haut niveau, qui sont souvent exprimés en langage naturel, en politiques qui sont moins abstraites et à un niveau qui pourrait être implémenté directement dans le système. Moffet et Sloman [Moffett and Sloman 1993] identifient le besoin de décrire formellement les relations entre les politiques de haut niveau et leurs raffinements. Plusieurs modèles de hiérarchie et raffinement ont été proposés. Nous retiendrons essentiellement le modèle de Wies [Wies 1995a] [Wies 1995b] [Wies et al. 1997] et de Heilbronner [Heilbronner 1997] [Heilbronner 1998] [Heilbronner 1999]. Puis nous proposerons notre propre modèle.

5.3.1 Le modèle de Wies

Wies [Wies 1995a] [Wies 1995b] [Wies et al. 1997] définit quatre niveaux d'abstraction (voir Figure 20) :

1. les *politiques de haut niveau / d'entreprise* qui dérivent directement des objectifs de l'entreprise et définissent les caractéristiques des services fournis ;
2. les *politiques orientées tâches* qui définissent comment les outils de gestion sont appliqués et utilisés ;
3. les *politiques fonctionnelles* qui définissent l'utilisation des services de gestion ;
4. les *politiques de bas niveau* qui opèrent au niveau des objets gérés.

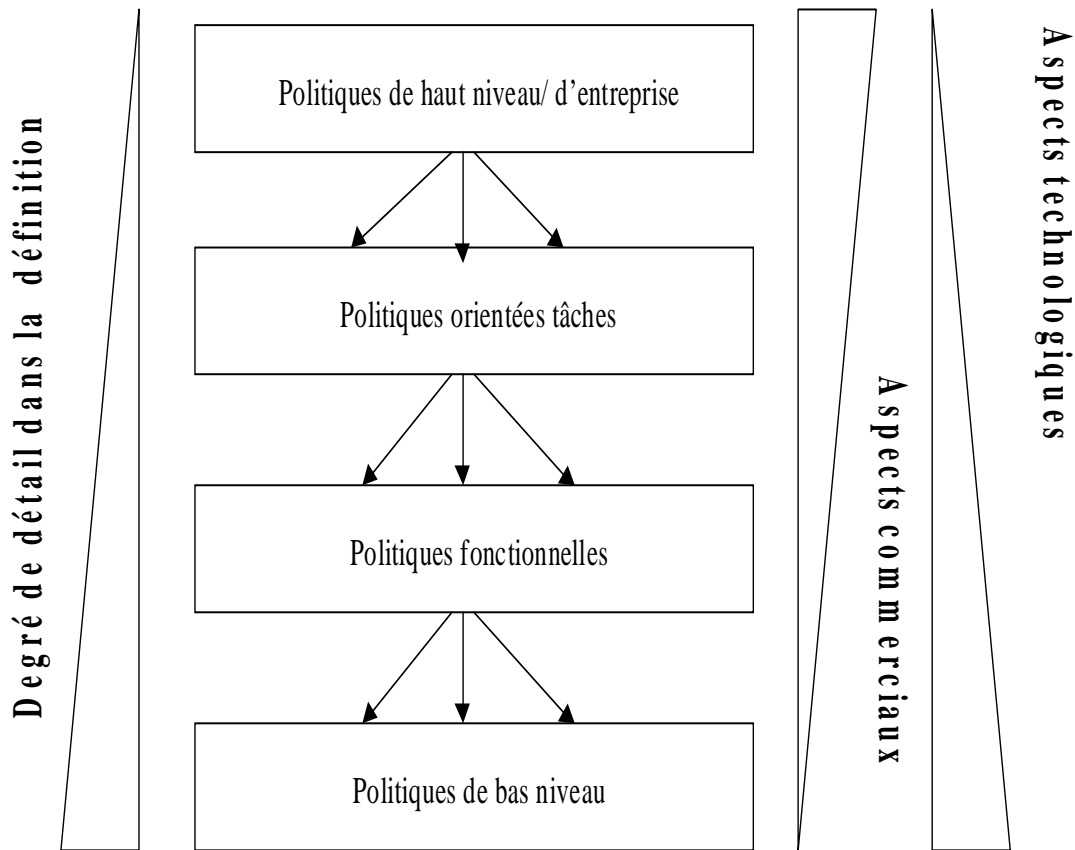


Figure 20 : Modèle de Wies

5.3.2 Le modèle de Heilbronner

Heilbronner [Heilbronner 1997] [Heilbronner 1998] [Heilbronner 1999] définit quatre niveaux d'abstractions :

1. les *politiques stratégiques* qui définissent les objectifs que l'administrateur veut atteindre (les produits et services fournis) ;
2. les *politiques orientées buts* qui définissent les politiques d'autorisation et d'obligation qui peuvent être négatives ou positives. A ce niveau d'abstraction sont spécifiés, de manière plus formelle, les sujets, cibles et les actions à réaliser ;
3. les *politiques opérationnelles* qui définissent les aspects techniques des actions de gestion à invoquer sur les ressources gérées conformément à la description formelle de l'interface de gestion (exemple IDL CORBA). Elles sont généralement spécifiées en terme de séquences d'actions à exécuter sur la cible ;
4. l'*infrastructure* où sont définis les objets gérés comme par exemple, les variables MIB SNMP, MOCs, etc.

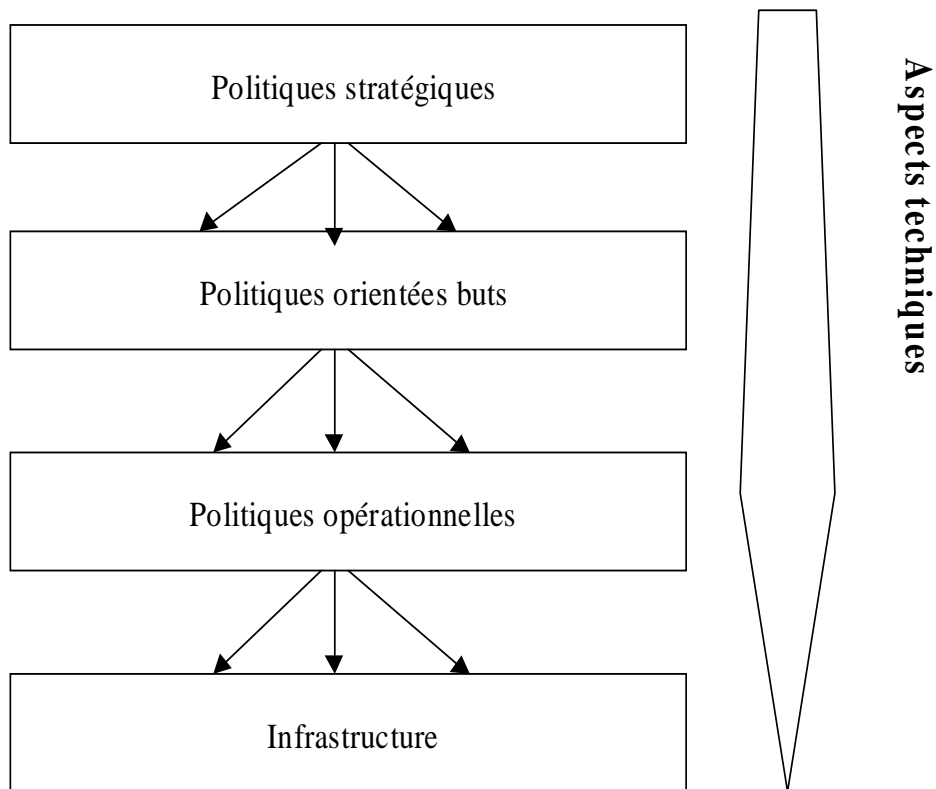


Figure 21 : Le modèle de Heilbronner

5.3.3 Autres modèles

D'autres approches ont été proposées [Mariott 1997]. Koch [Koch et al. 1996] définit une hiérarchie à trois niveaux : besoins (*requirements*), objectifs (*goal-oriented*) et le niveau opérationnel (*operational*). Dans son approche il utilise un langage différent pour chaque niveau ce qui permet de distinguer clairement les trois niveaux. Meyer [Meyer et al. 1996] propose également trois niveaux :

- les politiques de *haut niveau* pour décrire les objectifs ;
- les politiques de *niveau intermédiaire* pour décrire les stratégies indépendamment de la plate-forme utilisée ;
- les politiques de *bas niveau* pour la représentation concrète des stratégies.

Nous allons maintenant présenter le modèle que nous proposons.

5.3.4 Notre modèle

L'objectif de notre modèle de gestion de politiques de sécurité est :

- de permettre à l'administrateur de spécifier les règles à appliquer dans l'entreprise pour définir ce qui est autorisé et interdit ;

- d'implémenter ces règles et de veiller à ce qu'elles soient respectées.

Si nous reprenons le second point, toute violation d'une des règles définies par l'administrateur se traduirait par une attaque. Le rôle de notre SMA sera alors d'identifier ces violations et de reconnaître ainsi les attaques survenues dans le réseau. Pour que notre SMA puisse détecter ces attaques, il faudrait que les politiques de sécurité puisse être implémentées dans un langage qui puisse être interprété par les agents du système. Or, nous avons proposé (voir Chapitre 4), un langage pour la définition des schémas d'attaque afin qu'il puisse être interprété par nos agents. Par conséquent, le rôle des politiques de sécurité sera de pouvoir spécifier, au niveau le plus bas, les schémas d'attaque exprimés dans le langage proposé. A partir de là, nous avons identifié trois niveaux d'abstraction pour les politiques de sécurité (voir Figure 22) :

1. les *politique générales* ou *de haut niveau* P_{gen} qui spécifient de manière générale les règles de l'entreprise. Les politiques sont définies de manière informelle, dans un langage naturel ;
2. les *politique d'obligation et d'autorisation* $P_{ob/au}$ qui spécifient les règles d'obligation et d'autorisation. A ce niveau d'abstraction, seront définies :
 - les différentes entités sur lesquelles s'appliqueront ces règles, telles que : les stations de travail et les domaines à protéger ainsi que les utilisateurs internes ou externes. Ces utilisateurs ne sont pas forcément définis de manière précise. Ils peuvent être identifiés par le domaine auquel ils appartiennent ;
 - les actions autorisées et interdites ainsi que les contraintes temporelles et/ou géographiques.
3. les *politique schémas* P_{sch} et les *politiques de surveillance* P_{sur} :
 - les *politiques schémas* spécifient les schémas d'attaques à détecter. Nous nous focaliserons sur ce niveau de politique car ce sont les *politiques schéma* qui seront en définitif implémentées dans notre système ;
 - les *politiques de surveillance* spécifient les tâches de surveillance. Par le biais des politiques, l'administrateur peut vouloir surveiller des activités particulières concernant une partie du réseau ou un utilisateur particulier.

Nous nous limiterons, dans ce chapitre, à la spécification des politiques schémas.

L'intérêt de hiérarchiser une politique est de pouvoir automatiser le plus possible le processus de raffinement entre les différents niveaux de politiques de sécurité. Nous allons voir dans les paragraphes suivants comment vont interagir ces différents niveaux.

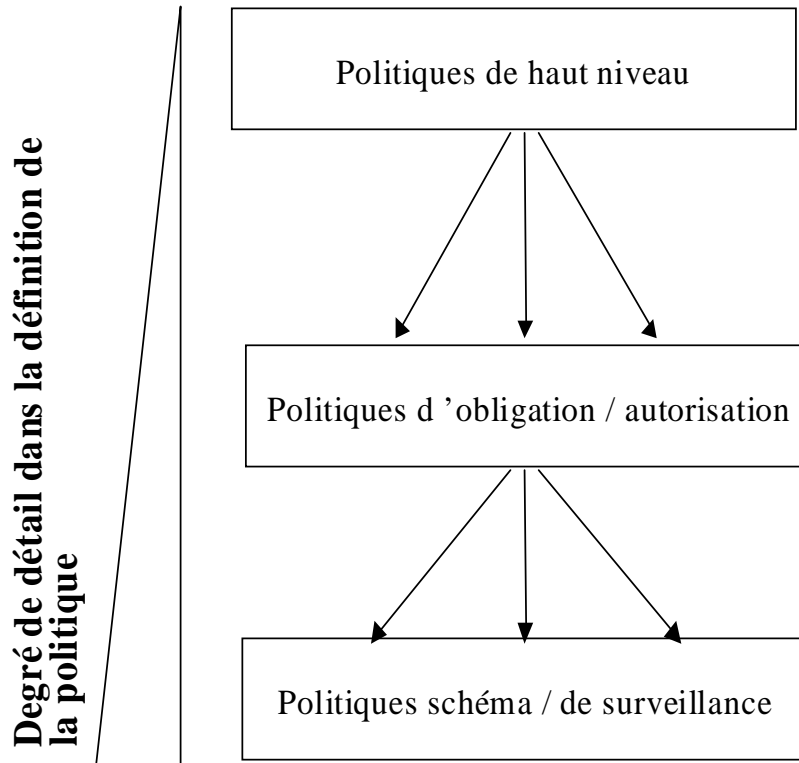


Figure 22 : Raffinement et hiérarchie d'une politique

5.3.4.1 Politique générale et politique d'obligation/autorisation

Le processus de raffinement entre les politiques générales et les politiques d'obligation/autorisation se fait manuellement et par conséquent est effectué par l'administrateur. Cela est généralement le cas dans tous les modèles existants [Heilbronner 1999] [Wies et al. 1997] [Mariott 1997] [Koch et al. 1996]. L'administrateur aura alors à traduire les politiques générales dans le formalisme utilisé pour écrire les politiques d'obligation/autorisation.

5.3.4.2 Politique d'obligation/autorisation et politique schéma

Pour que les politiques d'obligation/autorisation puissent être dérivées de manière automatique en politiques schémas, il faudrait qu'elles soient exprimées de manière précise. Autrement dit, à partir du formalisme utilisé pour exprimer les politiques d'obligation/autorisation, il faudrait pouvoir en déduire les différents opérateurs utilisés pour définir un schéma d'attaque (voir 5.4.3). Comme nous l'avons dit, les politiques de sécurité doivent nous permettre d'identifier :

- ce que nous voulons protéger et contre quoi nous voulons le faire ;
- ce que nous voulons faire dans le cas où une attaque serait détectée.

Pour cela, nous définissons deux types de *politiques schémas* :

1. les *politiques d'instanciation de schémas d'attaque* qui comme leur nom l'indique permettent d'instancier des schémas d'attaque,
2. les *politiques de réponse aux schémas d'attaque* qui permettent de préciser la liste d'actions à faire ou ne pas faire en cas de reconnaissance d'un schéma d'attaque.

Politiques d'instanciation de schémas

Ces politiques permettent la mise en œuvre des schémas d'attaque. Suivant que la politique de niveau supérieur est d'*obligation* ou d'*autorisation*, l'instanciation des schémas d'attaque ne se fera pas de la même façon.

Politiques d'autorisation

Les *politiques d'autorisation* définissent ce qui est autorisé ou interdit dans le réseau (ou partie du réseau) par rapport à des personnes, des entités du réseau et/ou des lieux géographiques. Elles vont nous permettre ainsi de définir et de créer de nouveaux schémas d'attaque. Suivant que la politique d'autorisation est *positive* ou *négative*, le traitement ne sera pas le même :

- à partir d'une politique d'autorisation positive (A₊), il faudra déduire les schémas d'attaque qui contredisent les actions autorisées ;
- à partir d'une politique d'autorisation négative (A₋), il faudra déduire les schémas d'attaque définis directement dans la politique d'autorisation.

Politiques d'obligation

Les *politiques d'obligation* vont nous permettre de parer des attaques spécifiques et d'instancier ainsi des schémas d'attaque existants afin de préciser ce que nous voulons réellement interdire. L'instanciation d'un schéma se fait :

- soit de façon nominative, i.e. par rapport à un utilisateur, réseau source ou réseau destination particulier ;
- soit de façon anonyme sans préciser la source et/ou destination.

De plus, la zone de détection d'une attaque peut être délimitée par l'administrateur :

- soit à tout le réseau d'entreprise,
- soit à une partie du réseau, dans le cas où il voudrait protéger fortement cette zone.

La politique d'obligation portera alors :

- soit que sur l'instance même du schéma d'attaque,
- soit sur l'instance du schéma et sur la zone géographique de détection.

Le mode *positive* ou *négative* d'une politique d'obligation n'aura aucune influence sur la manière d'instancier un schéma d'attaque. Le mode agira plus précisément sur les *politiques de réponse aux schémas d'attaque* afin de spécifier au système la manière de réagir face à une attaque.

Politiques de réponse aux schémas d'attaque

Ces politiques permettent la mise en œuvre des mesures de réponse. Elles définissent les actions que l'administrateur et les agents doivent faire ou ne pas faire en cas d'attaque. Cela concerne des actions de type :

- fermeture/ non fermeture de connexion,
- filtrage d'adresses source ou destination, en modifiant les paramètres d'un filtre/firewall.

Ces politiques devront être spécifiées par des *politiques d'obligation* (positive ou négative) au niveau supérieur.

Nous nous limiterons, dans cette thèse, à l'identification de ce type de politique sans aller plus loin dans le formalisme. Les mesures correctives par rapport aux attaques détectées sont hors du champ de ce travail. Néanmoins dans notre système nous définirons des actions simples de type : informer administrateur/ agents, envoyer alarme, etc.

5.4 Spécification d'une politique de sécurité

Dans la littérature, plusieurs formalismes ont été définis, néanmoins nous retiendrons les formalismes de Sloman [Mariott and Sloman 1996] [Lupu and Sloman. 1997d] [Lupu et al. 1997] et de Heilbronner [Heilbronner 1997] [Heilbronner 1998] [Heilbronner 1999]. Nous allons présenter ces deux formalismes puis proposer un formalisme qui est une extension de ces deux formalismes.

5.4.1 Formalisme de Sloman

Le formalisme proposé par Sloman [Mariott and Sloman 1996] [Lupu and Sloman. 1997d] utilise la syntaxe suivante :

Policy-identifier mode [trigger] subject {action} target [constraint]

Dans ce formalisme, à chaque règle (*identifier*) est associé un certain nombre d'arguments, facultatifs ou obligatoires. Cette notation a pour but principal l'établissement d'une relation d'un certain type (*mode*) entre une source (*subject*) et une cible (*target*) définissant une action (*condition, action*) à exécuter sous certaines contraintes (*constraint*).

- L'attribut *mode* permet de définir si la règle en question est une *autorisation* (A) ou une *obligation* (O) et si elle est *positive* (+) ou *négative* (-).

- L'attribut *subject* définit les objets pour lesquels l'obligation ou l'autorisation s'appliquent. Il s'agit par exemple des noms d'utilisateurs. Dans bien des cas, la source est une instance d'un domaine.
- L'attribut *target* comme la source définit les objets sur lesquels l'obligation ou l'autorisation s'appliquent.
- Les attributs *trigger* et *action* définissent ce que la source doit ou peut faire sur la cible (*action*) et dans quelles circonstances (*trigger*) cette action devra avoir lieu.
- L'attribut *constraint* permet de spécifier des contraintes quant à l'application de l'action mentionnée. Ces contraintes sont très souvent de type temporelle.

5.4.2 Formalisme de Heilbronner

Heilbronner définit un formalisme pour le deuxième et le troisième niveau de politique (but et opérationnel). Nous nous sommes intéressés au deuxième niveau (les *politiques orientées but*). La syntaxe utilisée pour définir ce type de politique est la suivante :

name modality subject sdomain tdomain action constraint

- Les attributs *subject*, *target*, *action* et *constraint* ont la même signification que dans le formalisme de Sloman. L'attribut *modality* correspond à l'attribut *mode* du formalisme de Sloman.
- Quant aux attributs *sdomain* et *tdomain* ils représentent les domaines de la source (*subject*) et de la cible (*target*).

5.4.3 Formalisme proposé

Notre objectif n'est pas de proposer un nouveau formalisme pour exprimer les politiques d'obligation/ autorisation mais d'utiliser les formalismes existants et de les étendre pour les appliquer à notre modèle. Pour cela, nous utilisons le formalisme défini par Sloman et Heilbronner que nous affinons pour pouvoir ensuite dériver les politiques schémas. Le formalisme proposé doit permettre de créer et d'instancier les schémas d'attaque. Pour créer un schéma d'attaque, nous avons besoin de :

- identifier le schéma (le nommer) ;
- spécifier la liste des types d'événements représentant le schéma d'attaque ;
- spécifier les contraintes sur les événements. La spécification des contraintes se fait en utilisant les opérateurs définis dans le langage d'expression des événements et des schémas d'attaque (voir chapitre précédent).

Pour instancier un schéma d'attaque, nous avons besoin de :

- sélectionner un schéma ;

- paramétrer le schéma en forçant les paramètres libres (attributs non spécifiés), comme par exemple la source ou destination des événements définis dans le schéma ;
- spécifier de nouvelles contraintes en cas de modification de certains attributs tels que le nombre de répétition d'un événement ou suite d'événements ;
- définir la portée de détection du schéma d'attaque (voir 5.3.4.2). Nous verrons dans le chapitre suivant que cette information permettra de sélectionner les groupes d'agents qui seront chargés de détecter ce schéma d'attaque.

Par conséquent, les politiques d'obligation/ autorisation sont définies par un ensemble d'attributs que nous regroupons dans la classe "*SecurityPolicy*" :

- un identifiant unique ;
- une description en langage naturelle ;
- l'auteur de la politique qui est généralement l'administrateur ;
- le *type de la politique* qui peut être : une politique schéma, une politique de surveillance ou une politique générale (n'est pas traité dans ce travail mais nous le rajoutons pour permettre une extension de la classe) ;
- le mode, qui peut être : obligation positive, obligation négative, autorisation positive et autorisation négative ;
- la *durée de vie* de la politique ;
- l'*état* de la politique qui peut prendre deux valeurs possibles : activée, désactivée ;
- le *nom du schéma d'attaque* ;
- la *liste des événements* représentant le schéma d'attaque ;
- la source de l'attaque ;
- la destination de l'attaque ;
- le *domaine à surveiller*, qui définit la portée de détection du schéma d'attaque ;
- des *contraintes sur le schéma d'attaque* ou sur la *liste d'événements* ;
- les actions à exécuter en cas de reconnaissance du schéma d'attaque ;
- la date de création de la politique.

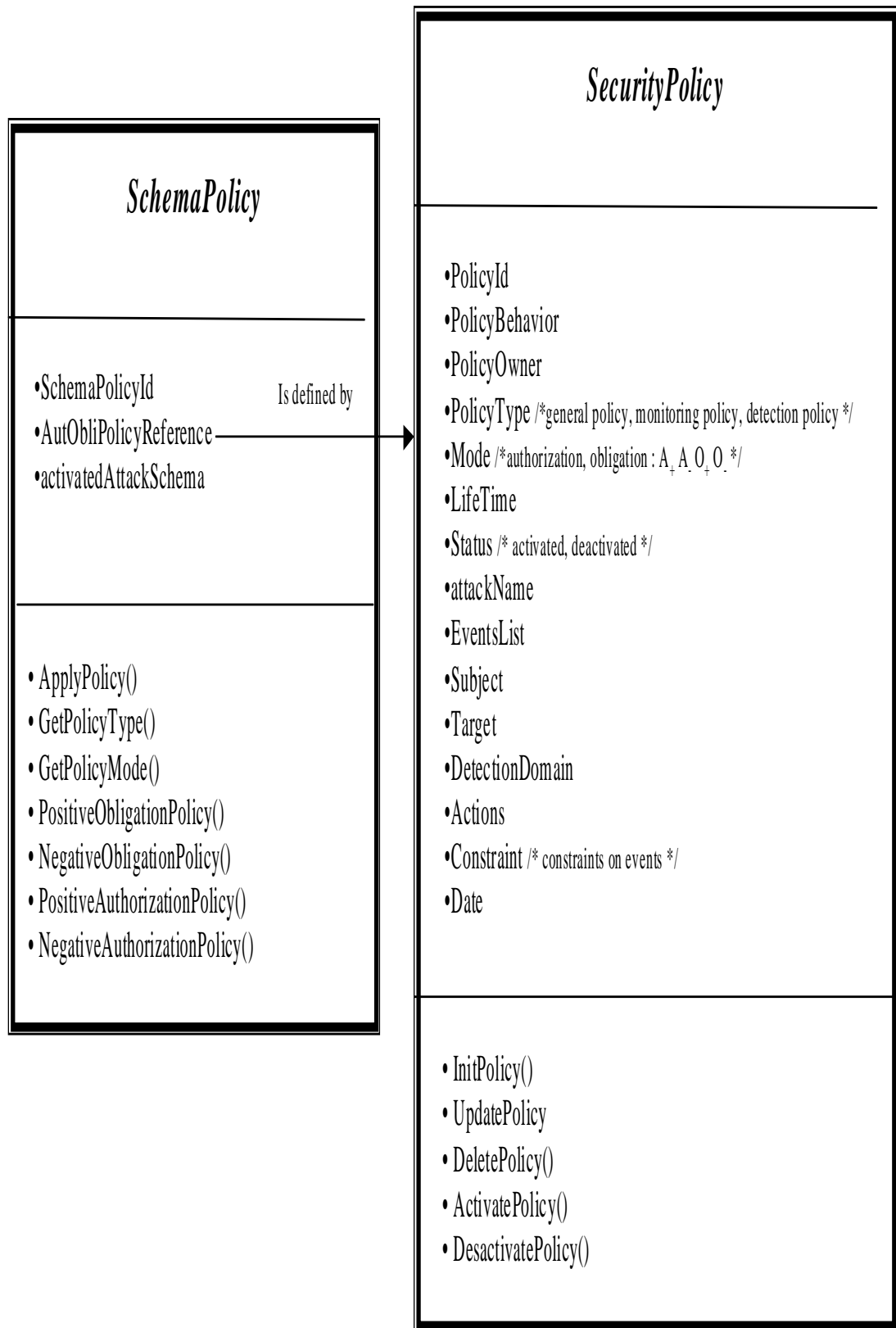


Figure 23 : Classes des politiques de sécurité

Pour définir une politique schéma, que nous représentons par la classe "*SchemaPolicy*" (voir Figure 23), nous avons besoin de :

- un identifiant unique ;
- la référence à la politique d'obligation/ autorisation dont la politique schéma dérive ;
- le schéma d'attaque à créer ou à instancier.

Pour dériver les politiques schémas à partir des politiques d'obligation/ autorisation, nous définissons cinq méthodes principales :

- `applyPolicy ()`, qui permet d'instancier une politique schéma à partir de la politique référencée ("*autObliPolicyReference*", voir Figure 23).
- `positiveObligationPolicy ()`, qui permet d'instancier une politique schéma et de définir les actions à faire en cas d'attaque ;
- `negativeObligationPolicy ()`, qui permet d'instancier une politique schéma et de définir les actions à ne pas faire en cas d'attaque ;
- `positiveAuthorizationPolicy()`, qui permet de déduire la politique d'autorisation positive en une politique d'autorisation négative afin de créer ensuite une instance de schéma d'attaque ;
- `negativeAuthorizationPolicy()` qui permet de créer une instance de schéma d'attaque.

Nous illustrons la dérivation des politiques schémas par deux exemples.

Exemple 1 : Création d'un schéma d'attaque par dérivation d'une politique d'autorisation négative

Dans cet exemple, nous voulons créer un schéma d'attaque à partir d'une politique d'autorisation négative.

Soit une politique d'autorisation P_1 définie avec les paramètres suivants :

PolicyId	P01
PolicyBehavior	détection d'échecs répétés de login provenant du Site A et à destination du serveur Web du réseau local B.
Policy Owner	Karima Boudaoud
Policy Type	politique de détection
Mode	autorisation négative
Life Time	du 10-10-2000 au 31-12-2000
Status	activée
Attack Name	<i>Doorknob Rattling</i>
Events List	e
Subject	<i>même</i>

Target *n'importe quel*
 Detection Domain *indifférent*
 Constraints
 window observation = 120 secondes
e.ObservationPoint = trafic réseau
e.EventType = connexion
e.EventName = telnet
e.Event Source = *même*
e.EventDestination = *n'importe quel*
e.EventResult = échec
e.Activity = *extranet*
 operators.frequency (**e**, 5)
 Date 10-10-2000

A partir de cette politique d'autorisation, nous obtenons le schéma de l'attaque "Doorknob Rattling" représenté de manière simplifiée par les paramètres suivants :

Schema Name **Doorknob Rattling**
 Window Observation 120 secondes
 Event Description List **e** (Observation Point trafic réseau
 Event Type connexion
 Event Name telnet
 Event Source *même*
 Event Destination *n'importe quel*
 Event Result échec
 Activity *extranet*)
 Series Attribute Operator List (Operator Type *Iteration*
 Event Description **e**
 Frequency 5)

Exemple 2 : Instanciation d'un schéma d'attaque par dérivation d'une politique d'obligation

Dans cet exemple, nous voulons instancier le schéma créé précédemment à partir d'une politique d'obligation (voir Figure 24).

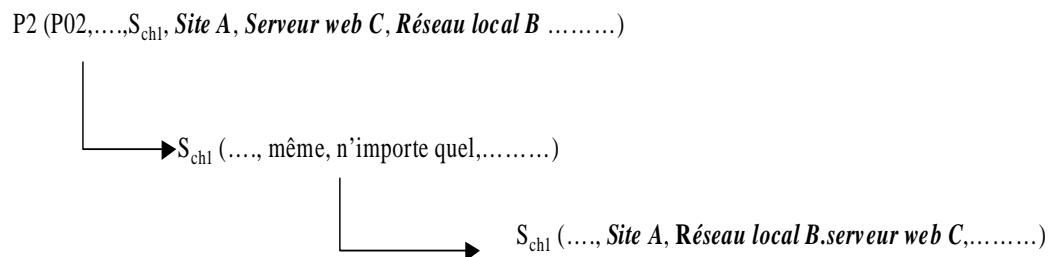


Figure 24 : Instanciation d'un schéma d'attaque

Soit une politique d'obligation P₂ définie avec les paramètres suivants :

PolicyId	P02
PolicyBehavior	détection d'échecs répétés de login provenant du Site A et à destination du serveur Web du réseau local B.
Policy Owner	Karima Boudaoud
Policy Type	politique de détection
Mode	obligation positive
Life Time	du 10-10-2000 au 31-12-2000
Status	activée
Attack Name	<i>Doorknob Rattling</i>
Subject	<i>SITE A</i>
Target	<i>Serveur Web</i>
Detection Domain	<i>Reseau local B</i>
Actions	<i>informer administrateur</i>
Date	10-10-2000

A partir de cette politique est instancié le schéma de l'attaque "*Doorknob Rattling*" avec les paramètres spécifiés dans la politique d'obligation.

Schema Name	<i>Doorknob Rattling</i>
Window Observation	120 secondes
Event Description List	e (Observation Point trafic réseau Event Type connexion Event Name telnet Event Source <i>SITE A</i> Event Destination <i>Reseau local B.Serveur Web</i> Event Result échec Activity extranet)
Series Attribute Operator List (Operator Type	<i>Iteration</i>
Event Description	e
Frequency	5)

5.5 Conclusion

Dans ce chapitre, nous avons présenté l'entité maîtresse de notre modèle de gestion de sécurité, à savoir les politiques de sécurité. Lorsque nous avons abordé les politiques de sécurité, nous nous sommes posé deux questions fondamentales :

1. comment interviennent-elles dans notre modèle ?
2. comment les représenter ?

Dans notre modèle, les politiques de sécurité permettent essentiellement d'instancier les schémas d'attaque à détecter. Pour pouvoir les représenter, il nous a fallu :

- d'une part, étudier les modèles de représentation existants,

- d'autre part, prendre en considération le langage d'expression de schémas d'attaque défini dans le Chapitre 4.

A l'issue de cela, nous avons proposé notre modèle où nous avons défini deux niveaux de représentation de politiques :

1. un niveau pour représenter les politiques d'obligation/autorisation,
2. un niveau pour représenter les politiques schémas.

Les politiques schémas permettent d'instancier des schémas d'attaque qui seront ensuite envoyés, sous forme de buts, à notre système multi-agents afin de les détecter. Nous décrivons le système multi-agents dans le chapitre suivant.

Chapitre 6 Le système multi-agents

6.1 Introduction

Dans ce chapitre nous décrivons le plan *Intelligence* de notre système de gestion de sécurité. Comme il a été dit dans le Chapitre 1, ce plan représente le système multi-agents défini pour la gestion de la sécurité d'un réseau. La conception d'un SMA n'est pas une tâche facile, en particulier dans le cadre de la gestion de la sécurité qui est un domaine très complexe. Il est maintenant largement approuvé que pour gérer et réduire la complexité des systèmes, il est nécessaire d'utiliser des techniques de modélisation et des méthodologies de conception qui supportent l'abstraction, l'organisation, la modularité et autres mécanismes [Kinny and Georgeff 1997]. Le but de ce chapitre n'étant pas de proposer une méthodologie pour la conception de notre SMA, nous nous sommes intéressés aux méthodologies existantes et nous avons choisi trois d'entre elles (voir Chapitre 3) : méthodologie de J. Ferber [Gutknecht et Ferber 1998] [Gutknecht et Ferber 1999], méthodologie de D. Kinny [Kinny et al. 1996] [Kinny and Georgeff 1997] et celle de M. Wooldridge [Wooldridge et al. 1999].

Chacune de ces trois méthodologies permet de décrire le modèle organisationnel d'un SMA, en définissant les rôles et les interactions entre les rôles. L'approche de D. Kinny [Kinny et al. 1996] permet en plus de décrire l'architecture interne de l'agent. Pour la conception de notre SMA, nous utilisons l'approche de J. Ferber et M. Wooldridge. Par conséquent, nous définissons :

- d'une part, la structure organisationnelle de notre système où nous identifions :
 - les rôles,
 - les interactions entre les rôles ;
- d'autre part, l'architecture interne de l'agent de sécurité, basée sur le modèle BDI, où nous identifions :
 - les fonctions principales nécessaires à un agent pour assurer la sécurité du réseau, indépendamment des types de rôles identifiés dans le modèle organisationnel ;

- les attitudes mentales de l'agent (ses croyances et ses buts).

Ce chapitre est donc organisé en deux parties principales. Dans la première partie nous présenterons le niveau *macro* de conception du SMA où nous définissons son organisation. Puis dans la seconde partie, nous décrivons le modèle interne d'un agent de sécurité.

6.2 Modèle organisationnel du système multi-agents

Dans cette section, nous décrivons les rôles, les interactions entre les rôles et la structure organisationnelle.

6.2.1 Identification des rôles

Pour décrire les rôles d'agents, nous utilisons l'approche de M. Wooldridge. Afin d'identifier les différents rôles, nous devons définir les fonctions que doit remplir notre système de gestion de sécurité, indépendamment de la technologie agent. Nous avons vu dans les chapitres précédents qu'il fallait :

1. spécifier les politiques de sécurité pour instancier les schémas d'attaque à détecter ;
2. distribuer la détection des schémas d'attaque à plusieurs entités distribuées dans le réseau. Chaque entité ayant à surveiller un domaine spécifique (éléments réseaux regroupés suivant l'organigramme de l'entreprise et/ou par zone géographique). La distribution des schémas d'attaque se fait suivant :
 - la portée de détection du schéma d'attaque qui définit le domaine où il doit être détecté,
 - les types d'activités (extranets, intranets, internes et locales) à surveiller pour détecter le schéma d'attaque,
3. détecter les schémas d'attaque en :
 - filtrant les événements pertinents pour la reconnaissance des schémas d'attaque instanciés,
 - analysant les événements filtrés,
4. agir en exécutant les actions spécifiées par les politiques de sécurité lors de l'instanciation des schémas d'attaque. Nous rappelons, que dans le cadre de cette thèse, cela se limite à informer l'entité concernée.

Chaque étape de traitement, mis à part la dernière, se traduit fonctionnellement par une ou plusieurs entités qui remplissent un rôle.

- Pour instancier les schémas d'attaque, nous définissons le rôle de *gestionnaire de politiques de sécurité*.

- Pour réaliser la distribution des schémas d'attaque, nous définissons un rôle de *gestionnaire extranet*. Suivant la taille et la distribution géographique (plusieurs régions dans un pays ou plusieurs pays) du réseau d'entreprise, nous proposons de définir un rôle supplémentaire : le *gestionnaire de sécurité intranet*.
- Pour assurer la détection des schémas d'attaque distribués, nous définissons trois rôles : *surveillant local extranet*, *surveillant local intranet* et *surveillant local interne*, qui sont remplis par des entités distribuées à différents points du réseau.

Nous explicitons chacun de ces rôles dans les paragraphes suivants.

Gestionnaire de politiques de sécurité

Le rôle de *gestionnaire de politique de sécurité* est donc d'assurer les fonctions suivantes :

- dialogue avec l'administrateur ;
- gestion des politiques de sécurité (création, mise à jour, activation,...) ;
- instanciation des schémas d'attaque à partir des politiques schémas ;
- traduction en termes de buts, la détection des schémas d'attaque.

Gestionnaire extranet

L'entité ayant le rôle de *gestionnaire extranet* a une vue globale du réseau d'entreprise. Les fonctions remplies par cette entité sont :

- gestion de la sécurité du réseau d'entreprise par rapport aux réseaux externes et entre les différents réseaux locaux constituant le réseau d'entreprise ;
- réception de buts traduisant les schémas d'attaque à détecter dans le réseau d'entreprise ;
- détection des attaques globales et coordonnées ;
- distribution des schémas d'attaque aux différentes entités distribuées dans le réseau et remplissant le rôle de *gestionnaire intranet*. Cela se traduit par une dérivation des buts reçus en sous-but, qui sont envoyés aux entités distribuées ;
- gestion et contrôle des entités distribuées ;
- réception des analyses pertinentes effectuées par les entités distribuées ;
- corrélation de ces analyses. D'autres analyses sont alors effectuées sur les événements suspects afin de confirmer ou non la détection d'une attaque. En fonction des résultats, d'autres analyses peuvent être demandées et de nouveaux schémas d'attaque peuvent être envoyés sous forme de nouveaux buts à atteindre.

Gestionnaire intranet

L'entité ayant le rôle de *gestionnaire intranet* a une vue locale du réseau d'entreprise. Elle remplit les mêmes fonctions que l'entité précédente mais au niveau d'un réseau local du réseau d'entreprise ; ce qui se résume en :

- gestion de la sécurité d'un réseau local constitué d'un ou plusieurs domaines ;
- détection des attaques coordonnées au sein du réseau local et se produisant entre ses différents domaines.
- gestion et contrôle des entités distribuées dans le réseau local remplissant le rôle de *surveillant local extranet*, de *surveillant local intranet* ou de *surveillant local interne*.

Surveillant local extranet

L'entité ayant le rôle de *surveillant local extranet* a pour fonction de détecter les schémas d'attaque qui se caractérisent par des activités de type *extranet*. Autrement dit, ce rôle est associé à la fonction de détection des attaques provenant ou en direction d'un réseau externe.

Surveillant local intranet

L'entité ayant le rôle de *surveillant local intranet* a pour fonction de détecter les schémas d'attaque qui se caractérisent par des activités de type *intranet* et *interne*. En d'autres termes, ce rôle a pour mission de détecter :

- des attaques provenant ou en direction d'autres réseaux locaux du même réseau d'entreprise ;
- des attaques internes au réseau local (i.e. : provenant ou en direction d'un ou plusieurs domaines du même réseau local).

Surveillant local interne

L'entité ayant le rôle de *surveillant local interne* a pour fonction de détecter les attaques locales à un domaine.

Pour pouvoir implémenter ces différents rôles, nous distinguons deux types de compétences : *gestion* et *surveillance*.

En fonction des différents types d'activités (voir Chapitre 4, section 4.2.1.1), nous identifions quatre types de surveillance et donc quatre types de compétences de surveillance :

- *surveillance extranet* pour la surveillance des activités extranets,
- *surveillance intranet* pour la surveillance des activités intranets,
- *surveillance interne* pour la surveillance des activités internes,
- *surveillance locale* pour la surveillance d'activités locales.

Ces types de surveillance sont paramétrés par :

$$S_{\text{surveillance}} = (T_{\text{type}}, A_{\text{ct}}, S_{\text{ens}}, S_{\text{sensibilite}}, T_{\text{type}A_{\text{ct}}}, T_{\text{type}S_{\text{source}}}, T_{\text{type}D_{\text{destination}}})$$

$T_{\text{type}} :=$ "même types" | "types différents";

$A_{\text{ct}} :=$ "activités extranets" | "activités intranets" | "activités internes" | "activités locales";

$S_{\text{ens}} :=$ "entrant" | "sortant";

$S_{\text{sensibilite}} :=$ "sensible" | "non sensible"

$T_{\text{type}A_{\text{ct}}} :=$ "Connexion" | "Transfert" | "Systèmes" | "Réseau";

$T_{\text{type}S_{\text{source}}} :=$ "même sources" | "sources différentes";

$T_{\text{type}D_{\text{destination}}} :=$ "même destinations" | "destinations différentes"

Pour ce qui est des compétences de *gestion* on distingue deux types :

- *gestion de politiques* pour la gestion des politiques de sécurité du réseau,
- *gestion de sécurité* d'un réseau distribué ou local.

Les types de compétences de gestion sont paramétrés par :

$$G_{\text{gestion}} = (T_{\text{type}}, R_{\text{réseau}})$$

$T_{\text{type}} :=$ "politique" | "sécurité" ;

$R_{\text{réseau}} :=$ "extranet" | "intranet" ;

Après avoir identifié les différents rôles, il nous faut maintenant déterminer comment ces rôles interagissent entre eux.

6.2.2 Interactions entre les rôles

Dans une organisation multi-agents, il existe inévitablement des dépendances et relations entre les différents rôles [Wooldridge et al. 1999]. Il nous faut donc maintenant analyser la nature et le but des interactions entre les rôles définis dans la section précédente. Ces interactions nous permettront par la suite de préciser les types de messages échangés entre les différents agents de notre système.

Reprenons les rôles deux à deux et analysant leurs dépendances. Fonctionnellement, les rôles sont indépendants les uns des autres. En effet, les dépendances entre les rôles définis sont soit *hiérarchiques* ou de type *report* d'informations (voir Figure 25).

1. Le rôle de *gestionnaire de politiques de sécurité* dépend de l'administrateur dans le sens où c'est ce dernier qui lui spécifie les politiques de sécurité à appliquer dans le réseau. C'est une dépendance hiérarchique.
2. Le *gestionnaire de politiques de sécurité* active les buts à atteindre qu'il envoie au *gestionnaire extranet*. Ce dernier est donc dépendant hiérarchiquement du *gestionnaire de politiques de sécurité*.
De plus le rôle de *gestionnaire extranet* reporte soit des alarmes, soit des rapports de sécurité pour alerter ou informer au sujet d'attaques produites dans le réseau. Il

peut également remonter d'autres types d'information à la demande de l'administrateur. Ce dernier pourrait demander par exemple l'état de sécurité du réseau, la liste des attaques les plus récentes, etc.

3. La dépendance entre le rôle de *gestionnaire intranet* et de *gestionnaire extranet* est hiérarchique car ce dernier distribue et spécifie aux *gestionnaires intranet* les schémas d'attaque à détecter localement. Le *gestionnaire intranet* reporte au *gestionnaire extranet* les attaques détectées ou les suspicions d'attaque.
4. Le *gestionnaire intranet* contrôle hiérarchiquement les *surveillants locaux*, en leur spécifiant les événements de sécurité pertinents à observer. Ces derniers lui reportent les résultats des analyses effectuées sur les événements.

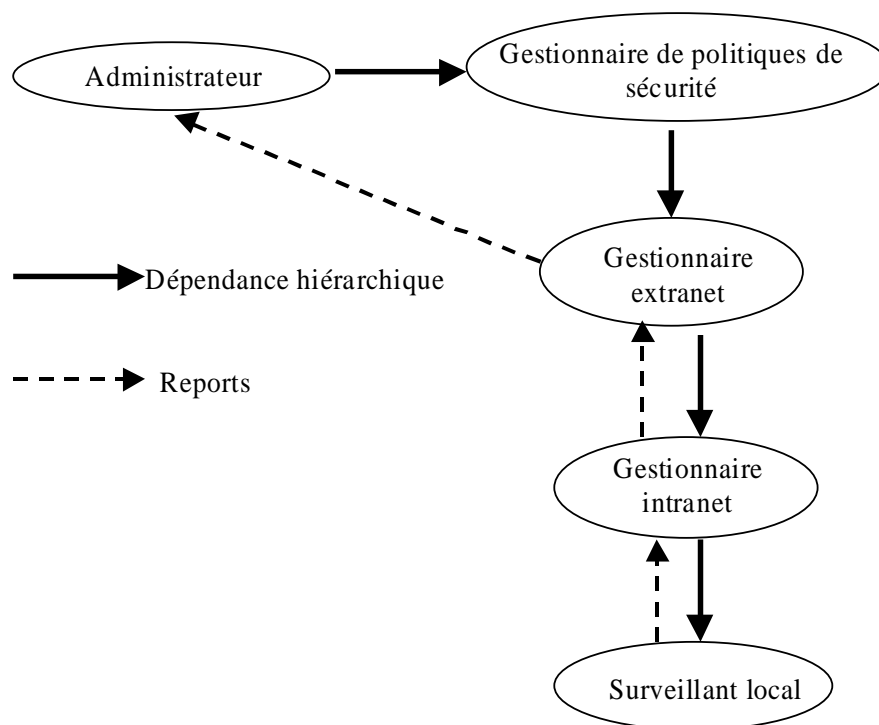


Figure 25 : Interactions entre les rôles

6.3 Structure organisationnelle du système multi-agents

J. Ferber définit une structure organisationnelle comme un ensemble de structures de groupe. Un groupe permet de regrouper un ensemble de rôles.

Dans notre SMA, nous regroupons les rôles, définis précédemment, en deux structures de groupe : une *structure de groupe gestionnaire* et une *structure de groupe surveillant local* (voir Figure 26) que nous décrivons dans les paragraphes suivants.

Groupe gestionnaire

Le *groupe gestionnaire* gère la sécurité globale d'un réseau, qui peut être local ou distribué. Il est défini par les rôles de *gestionnaire de politiques de sécurité*, de *gestionnaire extranet* et de *gestionnaire intranet*. Il a aussi bien une fonction de contrôle que de traitement de données. En effet, d'une part, il contrôle le *groupe*

surveillant local et d'autre part, il assure une corrélation de haut niveau des résultats d'analyse de ce groupe. Ainsi, il supporte la reconnaissance des attaques globales, incluant les attaques coordonnées internes à l'entreprise i.e.: se produisant entre les réseaux locaux du même réseau distribué. Nous pouvons dire que le *groupe gestionnaire* opère au niveau entreprise en corrélant les résultats produits par les différents sous-réseaux (intra-domaines).

Le *groupe gestionnaire* communique avec l'administrateur qui lui spécifie les politiques de sécurité à respecter et à qui il renvoie des rapports de sécurité et/ou alarmes. Ce groupe a une structure hiérarchique d'agents où nous distinguons trois niveaux d'agents qui ont un rôle de gestionnaire :

- l'*agent gestionnaire de politiques de sécurité* (AGPS) qui remplit le rôle de gestionnaire des politiques ;
- l'*agent gestionnaire extranet* (AGE) qui assure le rôle de gestionnaire de plus haut niveau, qui est celui de gestionnaire extranet. En réalité le nombre d'agents auxquels est attribué ce rôle dépend de la taille du réseau d'entreprise et de sa distribution géographique. En effet, pour faciliter la gestion de la sécurité d'un large réseau, nous avons choisi d'avoir un *agent gestionnaire extranet* par pays ;
- les *agents gestionnaires intranets* (AGI) qui ont le rôle de gestionnaire intranet.

Les agents gestionnaires intranet coopèrent et dialoguent entre eux pour assurer la sécurité du réseau distribué. En effet, un AGI peut recevoir des conclusions d'analyse ou des messages de suspicion d'autres AGI. Il corrèle alors ces suspicions avec les analyses des agents de niveau inférieur ou alors, en fonction de ces suspicions, demande à ses sous-agents des informations spécifiques (niveau de suspicion d'un utilisateur particulier, services utilisés par un utilisateur, etc...) afin d'approfondir les analyses de son/ses homologues et de confirmer leurs suspicions. Cette communication entre les AGI permet une détection coopérative des attaques coordonnées se produisant à différents points du réseau distribué.

Groupe surveillant local

Le *groupe surveillant local* est chargé de gérer la sécurité d'un domaine. Il n'a alors qu'une vue locale restreinte au domaine qu'il est chargé de surveiller. En l'occurrence, il est chargé de détecter les attaques locales au domaine. Ce groupe est composé d'un groupe d'*agents locaux* qui remplissent le rôle de *surveillant local extranet, intranet et interne* et qui sont distribués dans le réseau local. Ces *agents locaux* détectent des schémas d'attaque locaux à leur domaine. Les domaines sont distribués aux *agents locaux* par l'*agent gestionnaire intranet*. Ce groupe opère au niveau domaine en corrélant les résultats de ses différents *agents locaux*. Lorsqu'un agent suspecte une ou plusieurs activités, il le notifie aux autres agents afin de confirmer ou non l'existence d'une attaque.

Les *agents locaux* interagissent avec le réseau, soit passivement en regardant les fichiers d'audit et/ou observant le trafic réseau, soit activement en utilisant des sondes réseaux spécifiques. Ils récupèrent ainsi les événements de sécurité se produisant dans le réseau et les analysent pour détecter des comportements intrusifs.

Conformément aux trois rôles de *surveillants locaux* identifiés dans la section précédente, ce groupe contient trois types d'*agents locaux* :

- les **agents locaux extranets** (ALE) qui remplissent le rôle de *surveillant local extranet* ;
- les **agents locaux intranets** (ALI) qui assurent le rôle de *surveillant local intranet* ;
- les **agents locaux internes** (ALIn) qui ont le rôle de *surveillant local interne*.

Le *groupe surveillant local* assure un premier niveau de détection alors que le *groupe gestionnaire* assure un second niveau de détection plus approfondi.

Selon que le réseau soit local ou distribué, certains agents sont nécessaires et d'autres pas. Dans le cas d'un réseau distribué tous les agents seront présents, alors que pour un réseau local nous n'aurons besoin que de l'*agent gestionnaire de politiques de sécurité*, l'*agent gestionnaire extranet*, les *agents locaux extranets* et les *agents locaux internes*.

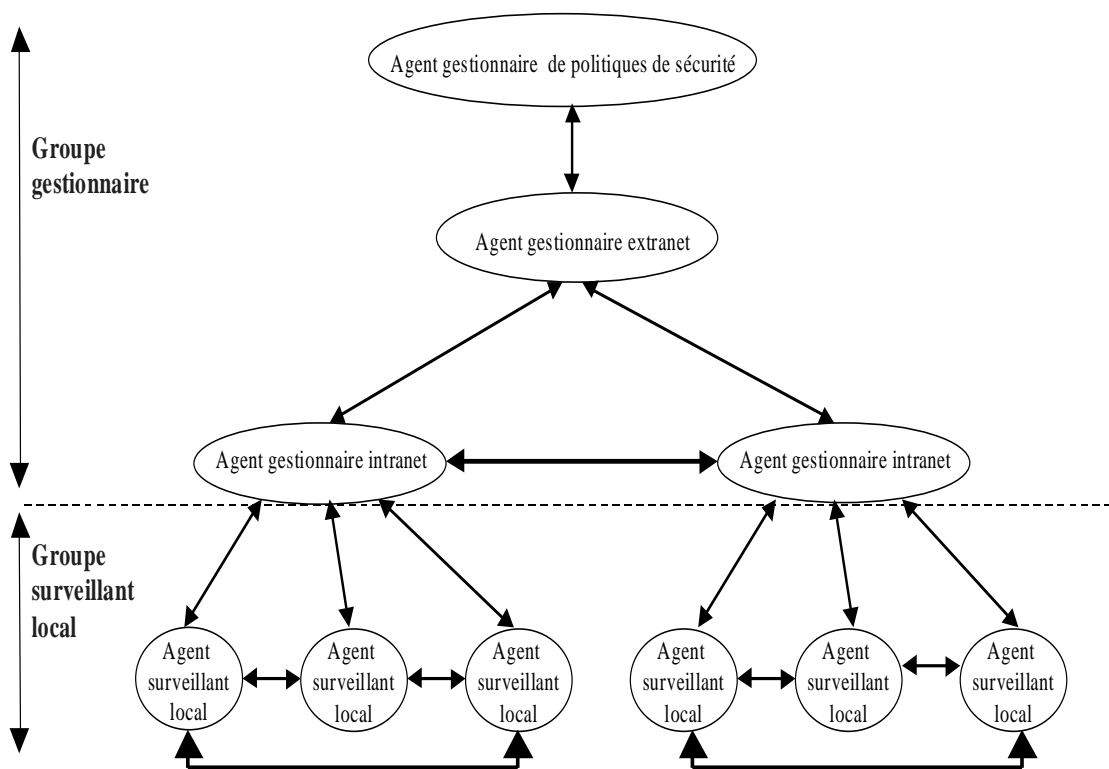


Figure 26: Architecture fonctionnelle du système multi-agents

L'organisation hiérarchique des agents permet d'assurer une analyse et détection aussi bien locale que globale. L'intérêt est d'essayer de détecter les attaques au plus tôt. En effet, chaque agent, à son niveau, a sa propre vision du réseau qui est limitée par le domaine qu'il doit surveiller et dont il est chargé de gérer sa sécurité.

Les domaines sont définis par des agents du *groupe gestionnaire*. Dans le *groupe surveillant local*, un domaine est un sous-réseau local d'entreprise qui représente un groupe de ressources réseaux, qui sont regroupées soit suivant l'organigramme de

l'entreprise en termes de départements, soit suivant des niveaux de sécurité qui sont spécifiés par les politiques de sécurité de l'entreprise. Dans le *groupe gestionnaire*, un domaine représente soit le réseau de l'entreprise soit un réseau local de ce même réseau d'entreprise.

Interactions entre le groupe gestionnaire et le groupe surveillant local

A partir des interactions entre les différents agents, nous pouvons déduire comment les deux groupes interagissent entre eux. Le *groupe gestionnaire* interagit avec le *groupe surveillant local* en :

- envoyant des messages sous formes de buts à atteindre dérivés des politiques de sécurité spécifiées par l'administrateur. Ces buts concernent la détection d'attaques spécifiques paramétrées par les politiques de sécurité;
- déléguant des fonctions de surveillance/détection spécifiques;
- demandant des informations particulières: l'état de sécurité d'une machine, le niveau de suspicion d'un utilisateur, la suite d'événements générée par un utilisateur, etc.;
- spécifiant les différents domaines à surveiller qui doivent être distribués aux différents *agents locaux*;
- recevant les rapports pertinents ou résultats d'analyse et alarmes.

L'interaction entre ces deux groupes permet la détection des attaques globales en corrélant les différentes analyses effectuées par les agents du groupe surveillant local.

Dans ce modèle hiérarchique multi-agents, chaque *agent gestionnaire* a la capacité de contrôler des agents spécifiques et d'analyser des données, alors que les *agents locaux* surveillent des activités spécifiques en vue de réaliser une analyse temps réel localisée.

Que ce soit au niveau du *groupe gestionnaire* ou du *groupe surveillant local*, les agents d'un même niveau communiquent et s'échangent leurs connaissances et analyses afin de détecter les activités intrusives de manière coopérative. Cette communication entre les agents permet de tracer les utilisateurs dans le réseau.

6.4 Modèle fonctionnel d'un agent de sécurité

Le système de gestion de sécurité doit pouvoir s'adapter à un environnement complexe, de part son évolution et variation continuelle, en termes de comportements utilisateurs (surtout dans le cas par exemple de la mobilité) et de problèmes de sécurité (nouvelles politiques de sécurité, nouvelles failles de sécurité, nouvelles attaques de sécurité,...). La connaissance manipulée par le système de gestion de sécurité varie donc constamment et cette dynamique complexifie la gestion de la sécurité des réseaux. De plus, le système doit non seulement considérer des informations reflétant l'état courant du système (événements de sécurité, attaques de sécurité à détecter, politiques de sécurité,...) mais aussi garder une trace des expériences passées. D'autre part, le système de gestion de sécurité doit respecter des contraintes temporelles assez strictes et par conséquent réagir au plus tôt lorsque des

événements indiquent un état anormal du réseau (comme par exemple une congestion du réseau due à une attaque de déni de service). Par conséquent, la complexité des fonctions de gestion de sécurité et en particulier de détection d'intrusions, nécessite d'avoir des agents capables de raisonner sur des problèmes et informations complexes mais aussi de réagir au plutôt pour respecter les contraintes temporelles. Ces agents doivent ainsi allier des capacités cognitives à des capacités réactives. Par conséquent, le choix d'un modèle d'agent hybride semble inévitable. Une fois le modèle choisi, il nous faut maintenant identifier les différentes fonctions et composantes d'un agent et la façon dont elles sont organisées pour fournir un comportement global. Pour gérer la sécurité du réseau, un agent devra d'une part, interagir avec le réseau et les autres agents; et d'autre part délibérer sur les événements collectés dans le réseau. Nous identifions donc deux types de fonctions (voir Figure 27) :

1. des fonctions de gestion des interactions de l'agent avec son environnement et les autres agents ;
2. des fonctions de délibération. Ces fonctions sont décrites dans les paragraphes suivants.

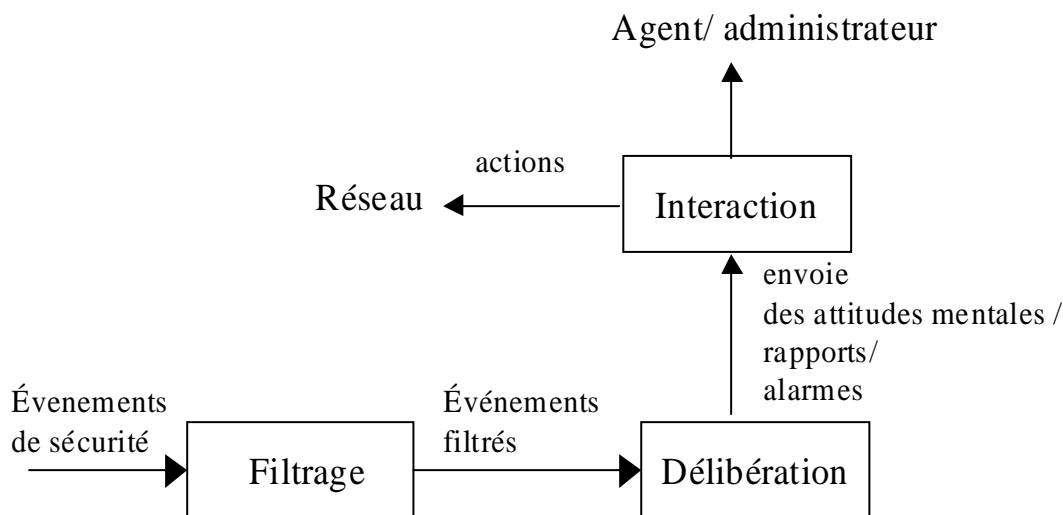


Figure 27: Interactions entre les différentes fonctions de l'agent

6.4.1 Fonction d'interaction

Nous avons vu plus haut que pour gérer la sécurité d'un réseau, les agents doivent non seulement communiquer avec les autres agents (l'agent humain et le SMA) mais aussi collecter les événements et agir sur le réseau en cas d'attaque. Par conséquent, un agent a deux domaines d'interaction : son environnement et les autres agents. Nous allons maintenant décrire ces interactions.

6.4.1.1 Interaction agents - réseau

Les agents interagissent avec leur environnement en :

1. filtrant et collectant les événements de sécurité produits dans le réseau;

2. agissant sur le réseau lorsqu'une attaque est détectée. Ils exécutent alors les actions appropriées comme par exemple reconfigurer les paramètres du firewall lorsqu'il existe, fermer une connexion établie par l'attaquant ou bloquer le compte utilisé par ce dernier.

Nous allons nous attarder sur la première fonction.

La fonction de *filtrage d'événements* permet de récupérer les événements de sécurité que l'agent est chargé d'observer. En effet, les événements se produisant dans l'environnement de l'agent (réseau), ne sont pas tous collectés :

- ils sont filtrés suivant les classes d'événements à observer. Ces classes sont spécifiées à l'agent lorsqu'il reçoit un *but de détection*, notion que nous verrons dans la section 6.4.2.2 ;
- ainsi, au moment où un événement se produit dans le réseau, l'agent teste son appartenance à l'ensemble des classes. S'il appartient à cet ensemble, il est collecté ;
- les événements filtrés sont ensuite classifiés puis rangés dans une file d'attente avant d'être traités par le module de délibération.

6.4.1.2 Interaction agent-agent/ admin-agent

La fonction d'*interaction* gère les interactions entre agents ou entre l'agent et l'administrateur.

- La fonction d'interaction entre agents gère les communications entre l'agent et ses voisins, son manager et ses subordonnés. Elle réalise la réception et l'envoi des messages aux autres agents. Le *module d'interaction* permet ainsi aux agents de se communiquer leurs analyses, leurs décisions et leurs connaissances. L'agent peut alors faire part de ses croyances et de ses suspicions aux autres agents gestionnaires et/ou locaux. Il peut aussi leur communiquer les buts à atteindre.
- La fonction d'interaction avec l'administrateur joue le rôle d'interface entre l'agent et l'administrateur. Cette fonction n'existe qu'au niveau de l'*agent gestionnaire des politiques de sécurité* et de l'*agent gestionnaire extranet*. Cette interface assure :
 - ◆ la réception des spécifications de l'administrateur ; en l'occurrence la spécification des politiques de sécurité qui sont reçues et traitées par l'*agent gestionnaire des politiques de sécurité* ;
 - ◆ la délivrance des rapports de sécurité et l'envoi des alarmes lorsqu'une attaque est détectée ;
 - ◆ l'envoi des requêtes de l'administrateur à l'*agent gestionnaire extranet* pour demander des informations telles que : l'état courant de sécurité du réseau, la liste des récentes attaques détectées ou la liste des utilisateurs suspects.

Pour supporter les interactions entre agents, nous utilisons un langage de communication que nous présentons brièvement dans la section 6.5.

6.4.2 Fonction de délibération

La résolution du problème de gestion de sécurité, en particulier de détection d'intrusions, doit prendre en compte des caractéristiques importantes du réseau telles que :

1. sa variation continue, notamment en termes de profils utilisateurs et de services offerts;
2. la variation de ses problèmes de sécurité tels que de nouvelles vulnérabilités et types d'attaques de plus en plus complexes.

Le caractère dynamique et non prédictible du comportement de l'environnement de l'agent (réseau) complexifie la gestion de sa sécurité. Par conséquent, pour assurer la sécurité du réseau, le SMA doit être capable de :

- détecter les attaques complexes en raisonnant sur des informations qui varient constamment,
- détecter rapidement les attaques simples en utilisant des règles simples de type situation → action.

Cela confirme notre choix d'une architecture hybride pour les agents de notre système. Pour la partie délibérative de nos agents, nous avons opté pour une approche basée sur le modèle BDI [Rao and Georgeff 1991a] [Rao and Georgeff 1991b] [Rao and Georgeff 1992] car c'est l'une des architectures délibératives les plus importantes et probablement la seule qui soit largement acceptée [Oliveira 1998].

La fonction de délibération est une fonction fondamentale de l'agent. C'est sur elle que repose l'intelligence et l'autonomie de l'agent. Elle manipule les attitudes mentales de l'agent qui seront développées dans les sections 6.4.2.1 et 6.4.2.2. Grâce à cette fonction, l'agent est capable de raisonner et d'extrapoler en se basant sur ses attitudes mentales, ses connaissances et son expérience de manière rationnelle. Par conséquent, les décisions de l'agent dépendent de l'état de sécurité du réseau, de ses attitudes mentales, des croyances des autres agents et de l'évolution du système voisin (les autres agents).

Dans le Chapitre 4, nous avons décrit le modèle de gestion des événements qui permet une représentation des concepts concrets (événements, suites d'événements, filtre d'événements et schémas d'attaque) du réseau à gérer, indépendamment du SMA. L'utilisation d'une technologie agent dont le comportement est guidé par son état mental, nécessite une mise en correspondance entre ce modèle événementiel, qui représente le modèle opérationnel, et le modèle BDI, qui est le modèle abstrait de notre système de gestion de sécurité. Le modèle BDI est guidé par les politiques de sécurité spécifiées par l'administrateur alors que le modèle opérationnel est piloté à un niveau plus élevé et plus abstrait par le modèle BDI.

A ce niveau de conception, nous proposons de formaliser le modèle de gestion de sécurité, en précisant les différentes entités impliquées. Soit $S = (P, R, A_g, D, A_{tt}, E, A, T)$, le modèle de gestion de sécurité où :

- ◆ P désigne l'ensemble des politiques de sécurité ;
- ◆ R exprime l'ensemble des rôles associés aux différents agents ;
- ◆ A_g représente l'ensemble des agents constituant le SMA ;
- ◆ D désigne l'ensemble des connaissances concernant le réseau à sécuriser et qui seront utilisées par le SMA. $D = \{\text{utilisateurs, machines, liste des utilisateurs suspicieux/attaquants, ...}\}$;
- ◆ A_{tt} représente les attitudes mentales de l'agent (croyances, buts, ...) ;
- ◆ E symbolise l'ensemble des événements de sécurité pertinents se produisant dans le réseau ;
- ◆ A représente l'ensemble des actions et mesures à prendre dans le cas de détection d'une attaque ou tout simplement d'atteinte d'un but ;
- ◆ T est l'ensemble des points temporels.

D et A_{tt} représentent le modèle d'information du SMA, construit sur le modèle BDI.

Ce modèle permet de représenter les attitudes mentales d'un agent qui sont regroupées essentiellement en [Shoham and Cousins 1994] [Brazier et al. 1996] :

1. *attitudes informationnelles* qui représentent les connaissances et croyances de l'agent,
2. *attitudes motivationnelles* qui représentent les buts et intentions de l'agent.

En plus de ces concepts, nous avons, dans le cadre de la sécurité, introduit une nouvelle notion informationnelle: la *suspicion* qui est fondamentale pour la détection des attaques de sécurité. Nous allons, dans cette section, décrire chacune de ces attitudes mentales.

Nous définirons chacun des concepts par rapport au contexte de gestion de sécurité car la définition d'un concept n'est pas unique. En effet, elle dépend de l'utilisation que l'on veut en faire et du contexte dans lequel il est utilisé [Ferber 1995].

Dans la section suivante, nous allons commencer par expliciter les attitudes informationnelles puis motivationnelles. Dans cette description, nous allons montrer comment les différentes attitudes mentales interagissent avec les politiques et les entités du modèle opérationnel pour gérer la sécurité du réseau.

6.4.2.1 Attitudes informationnelles

Bien que *connaissance* et *croyance* soient deux notions très proches et très souvent confondues, il existe une différence au niveau du degré de vérité de l'information qu'elles représentent où :

- la *connaissance* est une information qui est vraie;
- la *croyance* est une information qui peut être vraie ou fausse [Rao and Georgeff 1995].

6.4.2.1.1 Les connaissances

Nous définissons les connaissances comme étant des informations concernant le réseau à gérer et qui seront utilisées par les attitudes mentales. Nous distinguons deux types de connaissances :

- les *connaissances à caractère immédiat*;
- les *connaissances à caractère permanent*.

Connaissances à caractère immédiat

Les connaissances à caractère immédiat, comme leur nom l'indique, sont de courte durée et représentent les observations faites par l'agent sur son environnement. Leur validité a une durée de vie limitée dans le temps. Ces connaissances sont les événements de sécurité produits dans le réseau et observés par l'agent.

Connaissances à caractère permanent

Les connaissances à caractère permanent représentent les informations concernant le réseau et nécessaires pour gérer sa sécurité :

- les informations de configuration du réseau (équipements, utilisateurs, applications);
- la liste des utilisateurs connus/ groupes d'utilisateurs;
- la liste des machines connues;
- la liste des adresses connues (internes/externes, local/intranet/extranet), adresses interdites, réservées, impossibles;
- la liste des utilisateurs ayant la fonction d'administrateur et liste des administrateurs par machine;
- la liste des utilisateurs suspects/ attaquants : Ce sont les utilisateurs suspicieux. Lorsqu'un utilisateur est identifié comme suspect ou attaquant, un "casier judiciaire" lui est établi contenant les informations suivantes:

NomUtilisateur	Niveau	NomMachine	Date de l'attaque	Nombre
	S/A	Origine Destination		

- la liste des adresses suspectieuses/ attaquantes : Ce sont les adresses à surveiller ou à interdire l'accès;
- la liste des utilisateurs autorisés à se connecter de l'extérieur du réseau distribué. Par exemple, des utilisateurs particuliers en déplacement, les clients et/ou fournisseurs de l'entreprise;

NomUtilisateur	Sources autorisées	Destinations autorisées

- la liste des utilisateurs absents : ce sont les utilisateurs en congé, en mission ou en arrêt maladie.

6.4.2.1.2 Les croyances

Les croyances représentent la perception qu'a l'agent du comportement du réseau et de son état de sécurité [Rao and Georgeff 1991c]. Elles désignent aussi les connaissances qu'il a sur les autres agents et sur lui-même. Nous distinguons ainsi trois types de croyances :

- les ***croyances personnelles*** : elles expriment les connaissances de l'agent sur son propre état, i.e.: les informations le concernant, notamment par exemple le domaine qu'il est chargé de surveiller (éléments réseaux) ;
- les ***croyances relationnelles*** : elles sont très utiles et même nécessaires pour permettre la coopération entre agents. Elles représentent ce que sait l'agent des autres agents avec qui il communique. Ce sont donc toutes les informations dont il a besoin pour communiquer avec eux. Cela désigne aussi bien leur identité, leurs rôles que leurs compétences ;
- les ***croyances environnementales*** : ce sont les croyances les plus importantes pour gérer la sécurité du réseau et en l'occurrence, détecter les attaques de sécurité. Elles regroupent les *croyances environnementales locales* et les *croyances environnementales des autres*. Comme leur nom l'indique, les *croyances locales* désignent ce que croit l'agent sur le comportement et l'état de sécurité du réseau alors que les *croyances des autres* représentent les *croyances locales* des autres agents voisins. Dans le cadre de ce travail, nous nous focalisons sur ce type de croyances et nous distinguons quatre types de croyances environnementales :
 - ◆ les ***croyances schémas*** : ce sont des croyances abstraites et à caractère permanent. Elles décrivent les schémas d'attaque à détecter, qui ne sont instanciés que lors de l'envoi d'un but ;
 - ◆ les ***croyances scénarios*** : ce sont des croyances concrètes et à caractère immédiat. Elles représentent les suites d'événements de sécurité. Une *croyance scénario* est associée à une ou plusieurs *croyances schémas*. Etant donné

qu'elles ont une durée de vie limitée, les *croyances scénarios* ont une validité temporelle qui dépend de la validité temporelle des événements constituant la séquence d'événements de sécurité. Cette validité temporelle permet de traiter ces croyances et d'archiver celles qui ne sont plus valides temporellement.

- ◆ les *croyances locales* : ce sont des croyances définies par les *croyances scénarios* , une suspicion (que nous verrons plus tard) et une conclusion. Cette conclusion est le résultat du traitement d'une *croyance scénario* et d'une suspicion en fonction du but à atteindre (i.e. attaque à détecter). Les *croyances locales* peuvent également être fonction des *croyances locales* des autres agents voisins.

Si *croyance scénario* correspond à une attaque **X**

alors conclusion = attaque **X** détectée

sinon conclusion = suspicion forte | suspicion moyenne | suspicion faible

- ◆ les *croyances gestionnaires* : ce sont des croyances qui n'existent qu'au niveau d'un agent gestionnaire et qui sont fonction des *croyances locales* des différents agents locaux qu'il est chargé de gérer. Les *croyances gestionnaires* peuvent également être fonction des *croyances gestionnaires* des autres agents gestionnaires.

6.4.2.1.3 Les suspicions

La suspicion, introduite dans le cadre de la sécurité et qui sera justifiée dans le chapitre suivant, exprime la suspicion qu'a un agent sur une *croyance scénario* . C'est une notion très importante mais nous verrons l'importance dans la distribution des tâches de détection. Elle est différente de la croyance dans le sens où lorsqu'un agent observe une suite d'événements qui ne correspond ni à une séquence normale, ni à un schéma d'attaque alors il la suspectera comme une attaque mais pour le confirmer, il faudra un complément d'informations ou de confirmations de la part d'autres agents. L'agent pourra dans ce cas dire aux autres agents : "je *suspecte* que cette séquence d'événements soit une attaque".

Dans le chapitre suivant, nous verrons qu'en réalité la *suspicion* signifie l'atteinte d'un but local et donc la reconnaissance d'un sous-schéma partiel. Une *suspicion* est associée à une *croyance schéma* et est le résultat de l'analyse d'une *croyance scénario* par rapport à la *croyance schéma* . Elle est exprimée par un **niveau de suspicion**, le ou les attributs atemporels de la *croyance scénario* sur lesquels se porte la suspicion, la fréquence, s'il y a lieu, et le résultat des événements constituant la *croyance scénario* .

6.4.2.2 Attitudes motivationnelles

Les attitudes motivationnelles permettent de guider le comportement de l'agent. Dans la littérature, il existe plusieurs attitudes de ce type (buts, préférences, engagements, intentions) mais dans le cadre de cette thèse nous retiendrons uniquement les **buts**.

Les **buts** représentent l'état que doit atteindre l'agent ; en l'occurrence ses objectifs et les tâches qu'il doit accomplir [Rao and Georgeff 1991c]. Ils représentent l'attitude mentale la plus importante, vu que c'est elle qui guide le comportement de l'agent et ses décisions. A un haut niveau, les buts traduisent les politiques de sécurité de l'entreprise que l'on veut mettre en place et faire respecter. Puis ces buts sont

décomposés en sous-buts qui sont dits *buts délégués* car ils sont ensuite délégués aux différents agents concernés. Nous verrons dans le chapitre 7, à quel point les buts et leur délégation sont importants dans la délégation des tâches de détection et de surveillance. Ces buts sont définis par trois entités :

- une connaissance ou croyance;
- un objectif qui définit la contenance des valeurs limites de la croyance ou connaissance. Si la croyance/connaissance est associée à une valeur numérique, i.e.: par exemple à une période de temps (valeur temporelle), l'objectif précisera l'intervalle ou l'ensemble de valeurs auquel doit appartenir la croyance/connaissance. En d'autres termes, la période de temps dans laquelle l'agent doit atteindre son but. Cela peut être aussi une valeur logique, i.e.: que la croyance doit être vraie, par exemple si le but est de détecter un scénario d'attaque alors l'objectif sera que la croyance soit vraie c'est à dire la suite d'événements observée doit correspondre au schéma qui décrit la croyance;
- une ou plusieurs actions qui peuvent être soit *informationnelles*, soit *réactives*. Une action de type réactive, s'appliquerait plus à un but de détection d'attaque, qui serait traduit par: si je crois que j'ai tel schéma d'attaque (telle croyance) alors je dois exécuter un certain nombre d'actions. Une action de type informationnelle, serait par exemple d'informer l'administrateur ou l'agent gestionnaire de l'existence d'une certaine croyance ou juste de répondre à une requête formulée par un autre agent ou par l'administrateur.

Nous distinguons trois types de buts :

- les ***buts de surveillance*** qui permettent de demander à l'agent de surveiller des activités spécifiques, par exemple les activités d'un certain utilisateur, ou les *activités extranet entrantes*, etc. Le *but de surveillance* est défini par trois paramètres: la ou les types d'activités à surveiller, la durée de surveillance et l'action à exécuter si ces activités sont observées (informer l'administrateur ou l'agent gestionnaire ou des agents particuliers) ;
- les ***buts de détection*** qui permettent de spécifier à l'agent les attaques à détecter et les mesures à prendre si une attaque est détectée. Ce sont les buts les plus importants dans le cadre de la gestion de sécurité et de la détection d'intrusions. Ces buts sont associés à :
 - ◆ une *croyance schéma*,
 - ◆ un objectif qui spécifie les valeurs limites de la *croyance schéma*, plus précisément, celles des attributs de la croyance. Par exemple, que le résultat des événements de type connexion, constituant la croyance, doit être un échec, i.e.: observer les échecs de connexion,
 - ◆ une action de type *réactive* ;
- les ***buts informationnels*** qui permettent de demander à l'agent de récupérer des informations spécifiques sur l'état de sécurité du réseau, par exemple la liste des

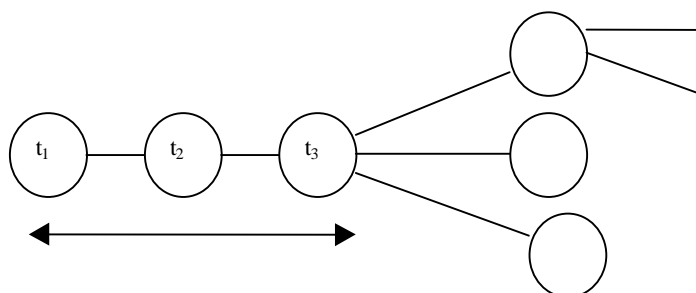
attaques détectées durant une certaine période de temps, la liste des utilisateurs suspects, les connexions courantes externes,...

Un *but de détection* permet d'instancier une *croyance schéma* et donc de définir une instance de schéma d'attaque en utilisant les opérateurs définis pour la description d'une parade d'attaque. Il est exprimé en fonction d'un type d'attaque à détecter, i.e. : *croyance schéma*, de valeurs attribuées aux attributs du schéma, de contraintes temporelles sur l'attribut temporel de la *croyance schéma* et des actions à exécuter si le schéma instancié est identifié (i.e : si le but est atteint).

6.4.2.3 Représentation des attitudes mentales

Rao et Georgeff [Rao and Georgeff 1991a] [Rao and Georgeff 1991b] [Rao and Georgeff 1992] modélisent les croyances, buts et intentions par une sémantique de *mondes possibles* où ils associent à chaque situation un ensemble de *mondes accessibles de croyances* (belief-accessible world), de *mondes accessibles de buts* (goal-accessible world) et de *mondes accessibles d'intentions* (intention-accessible world). Ces mondes caractérisent, respectivement, les mondes que l'agent croit possibles (les états possibles de l'environnement), les buts à atteindre et les buts qu'il s'engage à atteindre.

Ce modèle de représentation est trop abstrait pour être appliqué en tant que tel au contexte de gestion de sécurité. Dans notre modèle, les états possibles du réseau (mondes possibles) représenteraient toutes les suites d'événements qui pourraient se produire pendant toute la durée de vie d'un agent (de t_1 à t_n) et seraient symbolisés par un arbre temporel. Dans cet arbre, les chemins représenteraient les différentes suites d'événements possibles commençant par un événement e_x . Les arbres temporels ne pourraient pas être représentés explicitement; seuls les arbres construits au cours de la vie d'un agent seraient explicites.



La partie du monde dans lequel l'agent se trouve

Les attitudes mentales de l'agent peuvent être représentées de deux façons :

- ◆ soit de manière *intensionnelle*, en ne spécifiant que les propriétés qui permettent de les construire;
- ◆ soit de manière *extensionnelle*, en listant tous les éléments qui les constituent.

6.4.2.3.1 Représentation des croyances

Nous représentons les quatre croyances environnementales par quatre classes de croyances, qui dérivent d'une classe principale "*Belief*" (voir Figure 28). Cette classe

est principalement composée de 7 attributs, qui caractérisent une croyance environnementale :

- ◆ un identifiant unique,
- ◆ l'agent auteur de la croyance,
- ◆ le type de croyance (schéma, scénario, locale, gestionnaire),
- ◆ l'ensemble des buts pour lesquelles a été créée la croyance,
- ◆ la validité temporelle (vrai ou faux),
- ◆ la durée de vie,
- ◆ la date de création de la croyance.

Pour les autres classes de croyances, nous avons des attributs supplémentaires.

Une *croyance schéma* est représentée par une instance de schéma d'attaque.

Une *croyance scénario* est représentée par :

- ◆ une suite d'événements de sécurité pertinents qui se sont produites dans le réseau,
- ◆ ainsi que la référence à la *croyance schéma* qui a permis de générer cette *croyance scénario*.

Une *croyance locale* est représentée par :

- ◆ une *croyance scénario*,
- ◆ les *croyances scénarios* des autres s'il en existe,
- ◆ une *suspicion*,
- ◆ une conclusion.

Une *croyance gestionnaire* est représentée par :

- ◆ les *croyances scénario* des sous-agents,
- ◆ les *croyances scénarios* des autres gestionnaires s'il en existe,
- ◆ une conclusion.

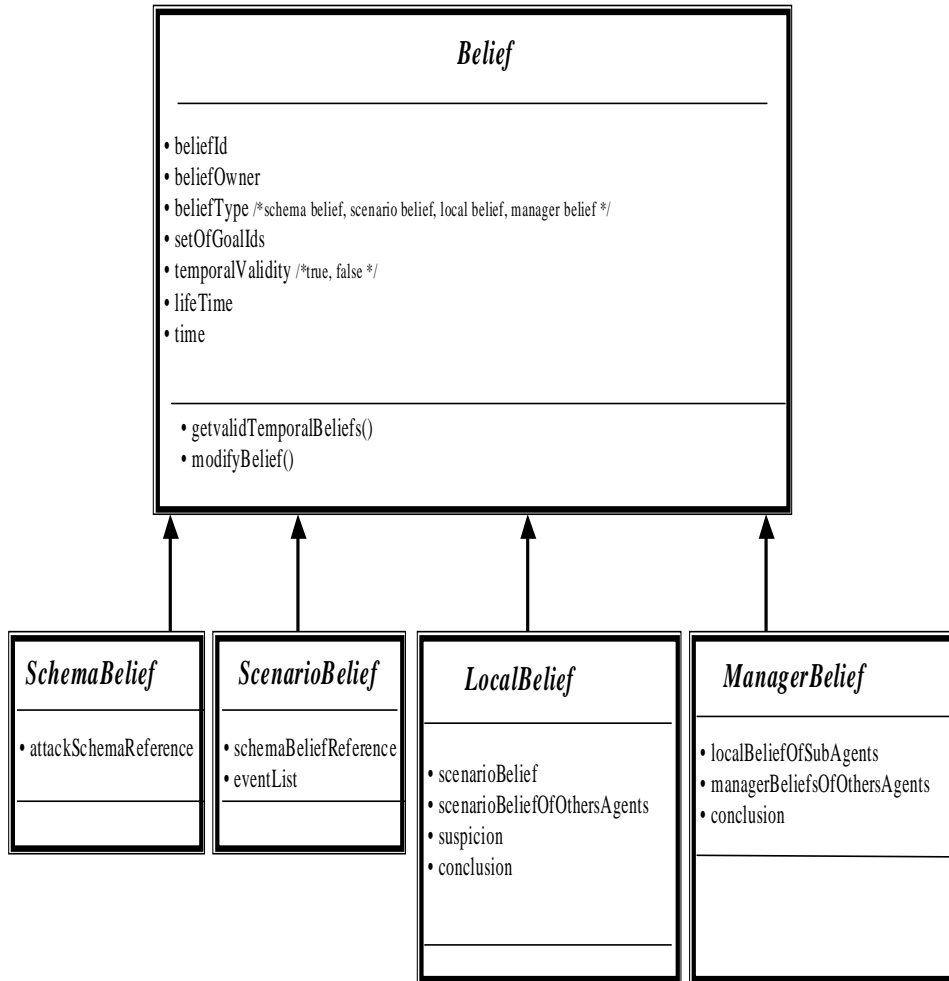


Figure 28 : Les classes de croyances

6.4.2.3.2 Représentation des suspicions

Une *suspicion* est représentée par (voir Figure 29) :

- ◆ la référence à la *croyance scénario* qui a généré la suspicion,
- ◆ le niveau de suspicion,
- ◆ les attributs sur lesquels portent la suspicion.

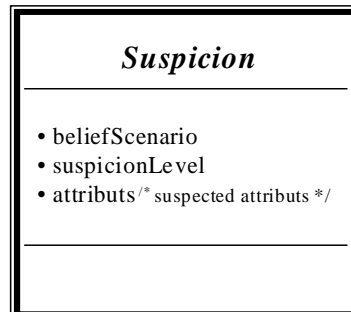
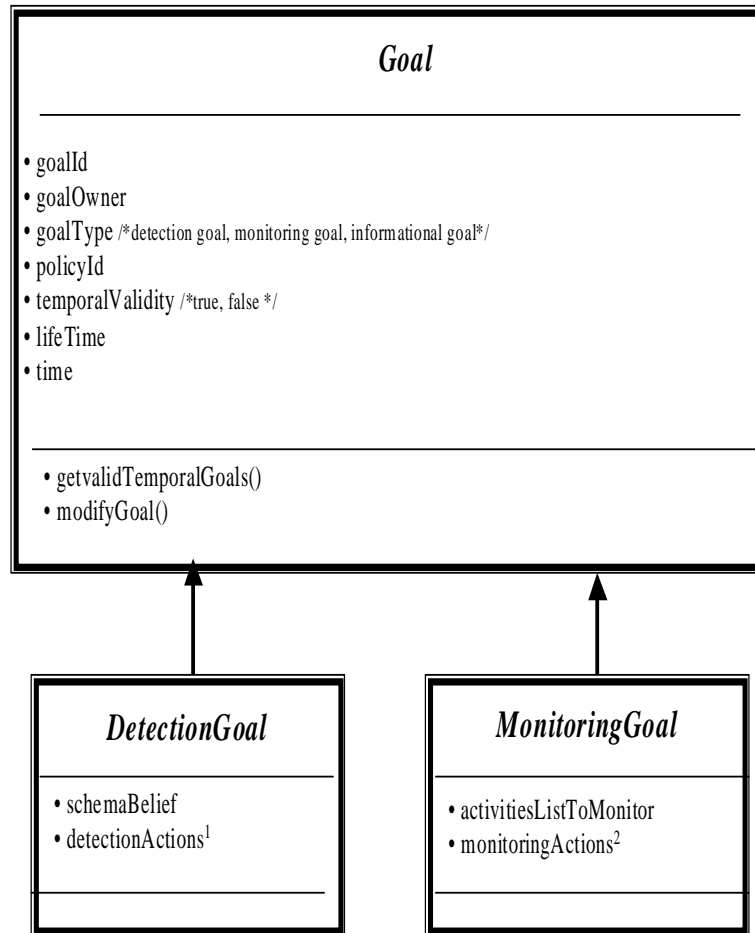


Figure 29 : La classe suspicion

6.4.2.3.3 Représentation des buts

Les buts sont représentés par trois classes : une classe principale "goal" et deux classes dérivées "detectionGoal" et "monitoringGoal" (voir Figure 30).



¹ close connection, inform manager, inform neighboring agents, register User Id in list of suspicious users, modify firewall parameters.

² send monitored activities to manager/administrator

Figure 30 : Les classes de buts

Pour la classe principale "goal", nous avons les attributs suivants :

- ◆ un identifiant unique,
- ◆ l'agent auteur de la création du but,
- ◆ le type de but (détection, surveillance, informationnel),
- ◆ l'identifiant de la politique d'où est dérivé le but,
- ◆ la validité temporelle (vrai ou faux),

- ◆ la durée de vie,
- ◆ la date de création du but.

Pour les autres classes de buts, nous avons des attributs supplémentaires.

Un *but de détection* est représenté par :

- ◆ une *croyance schéma*,
- ◆ les actions à exécuter lorsque le schéma d'attaque est reconnue.

Un *but de surveillance* est représenté par :

- ◆ une liste d'activités à surveiller,
- ◆ les actions à exécuter lorsque la surveillance est achevée.

6.4.2.4 Durée de vie des attitudes mentales

Les croyances et les buts ont une durée de vie limitée dans le temps. Cette durée de vie permet de limiter :

- ◆ la durée de mémorisation pour les croyances,
- ◆ la durée d'action, dans le cas des buts.

6.4.2.4.1 *Durée de vie des croyances*

Nous avons vu que les *croyances scénarios* étaient à caractère immédiat. En réalité, toute croyance a une durée de vie limitée dans le temps d'où l'importance de l'attribut temporel *date de création*. Cet attribut permet de calculer la validité temporelle des croyances en fonction de leur durée de vie. Cette validité temporelle permet de traiter les croyances et d'archiver celles qui ne sont plus valides temporellement.

Pour les *croyances scénario*, l'attribut "*time*" est redéfini à chaque mise à jour de cette croyance (à chaque modification de la suite d'événements). Puis sa validité temporelle est testée en fonction de son attribut temporel et de sa durée de vie. Ce test est effectué périodiquement et dès que la croyance est obsolète temporellement, elle est oubliée et donc archivée. Une *croyance scénario* peut être oubliée également lorsqu'elle a été reconnue comme un *croyance schéma*. En effet, lors du traitement et de l'analyse d'une *croyance scénario*, si elle correspond à une attaque, cela signifie qu'un but a été atteint et elle n'a donc plus lieu d'exister à cet instant précis. Cependant, elle pourra être utilisée ultérieurement pour la reconnaissance de nouvelles attaques non répertoriées.

6.4.2.4.2 *Durée de vie des buts*

Les buts ont également une durée de vie qui va permettre à l'agent de savoir jusqu'à quel moment il devra assurer la détection d'une attaque ou la surveillance de certaines activités.

6.4.2.5 Processus de délibération

Pour gérer la sécurité du réseau, le SMA doit d'une part gérer les politiques de sécurité et d'autre part analyser les événements de sécurité qui se produisent dans le réseau. Les politiques de sécurité permettent de sélectionner les événements pertinents à observer. Une fois les événements collectés, ils sont analysés en vue de détecter des attaques potentielles. Par conséquent, nous distinguons deux phases de traitement des attitudes mentales (voir Figure 31) :

1. une phase pour la sélection des événements,
2. une phase pour le traitement des événements.

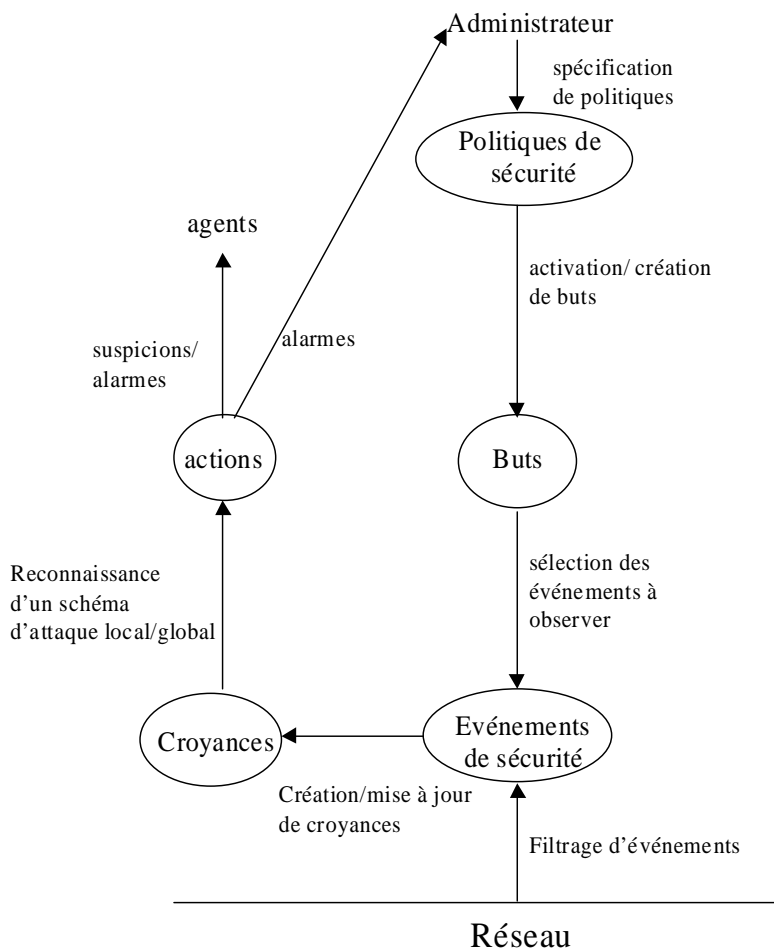


Figure 31 : Interactions entre les différents éléments d'information

6.4.2.5.1 Sélection des événements à observer

Cette phase passe par deux étapes :

1. la première permet d'activer les *buts de détection* à atteindre, qui dérivent des politiques de sécurité spécifiées par l'administrateur. Ces buts sont créés avec les instances de schémas d'attaque ;

2. la deuxième étape consiste à sélectionner les événements à collecter à partir des buts à atteindre et donc en fonction des instances de *filtres d'événements* des différents schémas d'attaque sélectionnés dans la phase précédente.

6.4.2.5.2 Traitement des événements de sécurité

Cette phase de traitement s'appuie sur les opérations de : *sélection*, *regroupement*, *comptage* et *reconnaissance de schémas d'attaque* (voir Chapitre 4). Cette phase se décompose en deux étapes principales : la *phase de filtrage* et la *phase de délibération*.

1. Lors de la *phase de filtrage* est exécutée l'opération de *sélection* où les événements pertinents sont collectés. Rappelons que cette opération permet de sélectionner, dans une fenêtre d'observation, les événements de sécurité dont les attributs ont des valeurs égales à celles spécifiées dans les instances de *filtres d'événements*.
2. La *phase de délibération* consiste à traiter les croyances en vue d'atteindre les buts activés.
 - à partir des événements collectés lors de la phase de filtrage, une mise à jour des *croyances scénarios* de l'agent est effectuée. Lors de cette phase sont exécutées les opérations de *regroupement* et de *comptage* ;
 - les *croyances locales de l'agent* peuvent également être mises à jour lorsque l'agent reçoit les *croyances locales* d'autres agents.
 - les *croyances scénarios* sont ensuite analysées afin de voir si elles ne correspondent pas à des *croyances schémas*. Dans cette phase est exécutée l'opération de *reconnaissance de schémas d'attaque*. Si c'est le cas, cela signifie que l'agent a atteint son *but de détection*.

6.5 Langage de communication

Pour gérer les interactions entre agents, nous avons besoin d'un langage de communication. L'un des standards de communication de haut niveau, qui a été développé est le langage KQML (Knowledge Query and Manipulation Language). Ce langage est fondé sur les actes de langage pour permettre à des agents cognitifs de coopérer [Ferber 1995]. Nous adoptons KQML comme le langage de communication agent de notre SMA car il fournit un grand choix d'actes de communication (performatives) (voir Annexe A), qui sont adaptés pour supporter la délégation, la coopération et la coordination. Le but du langage de communication agent est de permettre l'envoi et la réception des attitudes mentales entre agents, notamment la délégation des buts et l'échange de croyances et suspicions. Ainsi ces attitudes mentales ont été choisies comme contenu des performatives KQML. Nous avons sélectionné les performatives dont nous avons besoin pour supporter les interactions de nos agents. Nous présentons brièvement les performatives choisies.

Délégation des buts

Pour la délégation des buts, nous avons choisi les performatives suivantes :

- ◆ *achieve* : permet à un agent gestionnaire de déléguer un but à un autre agent ;
- ◆ *unachieve* : permet à un agent gestionnaire d'annuler la délégation d'un but ;
- ◆ *ask-if* : permet à un agent gestionnaire de demander à l'agent, à qui il doit déléguer un but, s'il accepte de recevoir un but ;
- ◆ *tell* : permet à un agent de confirmer à un agent gestionnaire d'accepter de recevoir un but ;
- ◆ *untell* : permet à un agent de notifier à un agent gestionnaire de ne pas accepter de recevoir un but.

Coopération et coordination

Pour l'échange des croyances et des suspicions, nous avons choisi les performatives suivantes :

- ◆ *ask-one* : permet à un agent de demander à un autre agent de lui envoyer ses croyances et ses suspicions ;
- ◆ *ask-all* : permet à un agent de demander à tous les agents de lui envoyer leurs croyances et leurs suspicions ;
- ◆ *tell* : permet à un agent de renvoyer à l'agent initiateur de la demande, les croyances et les suspicions demandées ;
- ◆ *untell* : permet à un agent d'annuler l'envoi des croyances et des suspicions.

Pour gérer la mise à jour de la base de connaissances des agents, il existe cinq performatives :

- ◆ *insert* : permet à un agent d'insérer dans la base de connaissances d'un autre agent des croyances ou des suspicions ;
- ◆ *unsinsert* : permet à un agent d'annuler l'insertion de croyances ou de suspicions dans la base de connaissances d'un autre agent ;
- ◆ *delete-one* : permet à un agent de demander à un autre agent d'effacer une croyance ou une suspicion de sa base de connaissances ;
- ◆ *delete-all* : permet à un agent de demander à un autre agent d'effacer de sa base de connaissances toutes les croyances ou toutes les suspicions qui satisfont une certaine condition ;
- ◆ *undelele* : permet à un agent d'annuler la demande effectuée par un *delete-one* ou *delete-all*.

Diffusion

Pour la diffusion des attitudes mentales, nous avons choisi les performatives suivantes :

- ◆ *broadcast* : permet à un agent de diffuser la délégation d'un but ou l'envoi de croyances et de suspicions à d'autres agents
- ◆ *forward* : permet à un agent de faire suivre la délégation d'un but ou l'envoi de croyances et de suspicions à un autre agent

6.6 Conclusion

Dans ce chapitre, nous avons décrit notre SMA conçu pour la détection des attaques. Pour le concevoir, nous nous sommes basés sur des méthodologies de conception existantes, notamment celle de J. Ferber, de D. Kinny et de M. Wooldridge. Conformément à ces méthodologies, nous avons défini :

- la structure organisationnelle de notre SMA où nous avons identifié : les rôles, les interactions entre les rôles et les groupes ;
- l'architecture interne de nos agents où nous avons identifié trois fonctions :
 - la fonction de filtrage pour la collecte des événements ;
 - la fonction de délibération qui manipule les attitudes mentales de l'agent (ses buts, ses croyances et ses suspicions) pour la reconnaissance de schémas d'attaque ;
 - la fonction d'interaction entre agents qui gère les communications entre agents. Pour supporter les interactions entre agents, nous avons dû choisir un langage de communication agent. Nous avons opté pour le langage KQML car il offre un grand choix de performatives pour supporter l'envoi, la réception et la mise à jour des attitudes mentales.

Dans le chapitre suivant, nous décrivons la distribution des schémas d'attaque et la délégation des buts de détection.

Chapitre 7 Distribution de la détection des schémas d'attaque

7.1 Introduction

Nous avons vu que la distribution et la délégation étaient deux caractéristiques importantes pour une détection d'intrusions efficace (voir Chapitre 1 et Chapitre 2). En effet, nous avons vu qu'une approche centralisée n'a pas de sens car elle complexifie la détection des attaques réseaux. De plus, par souci de simplification et de limitation de la charge du réseau, il est très important de répartir les observations et analyses sur plusieurs agents. D'où l'architecture proposée (voir chapitre précédent). Nous avons également vu que la variabilité des réseaux à sécuriser, en termes de nouvelles ressources, de services et surtout de types d'attaque, qui ne cessent d'augmenter, nécessite un système de gestion de sécurité dynamique. Cette dynamique est nécessaire pour que les tâches de détection puissent être modifiées dynamiquement afin de s'adapter aux changements du réseau. Cela est rendu possible grâce au mécanisme de délégation, qui a été l'objet de nombreux travaux, en particulier dans la gestion de réseaux, connus sous le nom de gestion par délégation [Goldszmidt 1993] [Goldszmidt and Yemini 1995]. La délégation étant également une propriété des SMAs, nous l'avons exploité afin d'avoir un système qui soit le plus flexible possible.

Dans ce chapitre, nous voulons montrer comment se traduit la distribution et la délégation dans notre système. Nous analysons et décrivons comment la détection d'un schéma d'attaque est distribuée dans le SMA et comment sont délégués les buts de détection. Puis nous discutons des problèmes de couverture, pour la détection des attaques, par rapport aux deux approches centralisée et distribuée.

7.2 Distribution de la détection des attaques

Nous avons vu les différents éléments du modèle événementiel dont nous avons besoin pour détecter les attaques ainsi que leurs interactions (voir Chapitre 4). Lors de l'analyse nous n'avons pas pris en compte l'aspect distribué du réseau et du SMA. Or, cela est irréaliste puisque la détection d'une attaque ne se fait pas par une seule entité centralisée. Si nous reprenons le schéma d'attaque, qui est l'élément principal, nous l'avons décrit en tant que *schéma d'attaque global*. Pour pouvoir détecter ce *schéma*

global, il nous faut le décomposer en *sous-schémas locaux* afin qu'il puisse être détecté localement par les différents agents répartis dans le réseau. En effet, les agents n'ayant qu'une vision locale du réseau, ils ne peuvent détecter que les attaques locales. Nous distinguons donc deux notions différentes : *schéma global* et *schéma local*.

Schéma global

Le *schéma global* est un schéma d'attaque que le SMA doit détecter. La détection de ce schéma ne sera notifiée qu'aux agents gestionnaires.

Schéma local

Le *schéma local* est un schéma dérivé du *schéma global* mais qui doit être détecté par les agents locaux.

Interprétation de la reconnaissance d'un schéma global et d'un schéma local

La détection d'une attaque locale ne signifie pas que l'attaque globale aie été détectée mais que simplement qu'il y a une forte suspicion de l'occurrence de l'attaque. En effet, la reconnaissance d'un *schéma global* signifie qu'il y a une forte suspicion d'attaque et que la probabilité que cela soit une attaque est égale à 1. Alors que la reconnaissance d'un *schéma local* signifie qu'il y a une *forte suspicion* d'attaque mais que la *probabilité* que cela soit une attaque est inférieure à 1.

Pour que les différents agents puissent différencier la détection d'un schéma d'attaque global, d'un schéma d'attaque local, nous proposons d'introduire la notion de *suspicion*. La *suspicion* est le résultat de la reconnaissance d'un *schéma local*. Sans cette notion supplémentaire, les agents ne pourraient pas se coordonner.

Après avoir défini ces trois notions (schéma global, schéma local et suspicion), nous décrivons dans les sous-sections suivantes les interactions entre ces notions, notamment :

- comment un schéma d'attaque global est dérivé en schémas locaux,
- et comment la reconnaissance des schémas d'attaque locaux permet d'identifier un schéma d'attaque global.

Dérivation d'un schéma global

Pour dériver un schéma global en schémas locaux, nous devons :

- dériver une classe schéma (schéma global) en classes sous-schéma (schémas locaux) ;
- dériver les opérateurs définis dans la classe schéma (schéma global).

La dérivation d'un schéma global en schémas locaux ne se traduit pas de la même façon suivant le type des opérateurs utilisés pour décrire le schéma global. En fonction du type d'opérateurs, les classes sous-schémas ont :

- soit la même structure (attributs + méthodes) que le schéma global,

- soit une structure modifiée (restreinte ou étendue).

Reconnaissance d'un schéma global

Après avoir dérivé le schéma global en schémas locaux :

- les schémas locaux sont envoyés aux différents agents concernés ;
- chaque agent aura à reconnaître son schéma local ;
- les différents agents devront coopérer pour reconnaître le schéma global.

Pour la reconnaissance d'un schéma local, le principe est le même que celui vu précédemment (voir Chapitre 4) mais c'est l'interprétation qui est différente :

- la reconnaissance d'un schéma global est une attaque,
- alors que la reconnaissance d'un schéma local est une suspicion d'attaque. Lorsqu'un agent reconnaît un schéma local, il génère une suspicion qu'il envoie aux autres agents.

Pour la coopération entre les différents agents, nous proposons deux modes de fonctionnement :

Premier mode

Dans le premier mode de fonctionnement, le processus de reconnaissance d'un schéma global est sous le contrôle d'un agent gestionnaire :

- à chaque fois qu'un agent local génère une suspicion, due à l'observation d'une séquence d'événements qui correspond à un schéma local, il l'envoie à son agent gestionnaire ;
- l'agent gestionnaire analyse ensuite cette suspicion et compare le schéma global avec la suite d'événements observée par l'agent local. En fonction du résultat :
 - soit il génère de nouveaux schémas locaux qu'il envoie aux différents agents ;
 - soit il génère une alarme qu'il envoie à l'entité au-dessus pour l'informer de la détection d'une attaque.

Dans ce mode de traitement, il n'y a pas de coopération explicite entre les agents de même niveau pour détecter une attaque car cette coopération est gérée par l'agent gestionnaire.

Deuxième mode

Dans le deuxième mode, il y a davantage de coopération entre les agents sous la responsabilité de l'agent gestionnaire :

- les agents s'échangent leurs suspicions et modifient leurs schémas locaux en fonction des suspicions reçues ;
- l'agent gestionnaire n'intervient en aucun cas dans ces échanges. En fait, il ne communique avec ses agents que pour :
 - envoyer le premier schéma local ;
 - recevoir une alarme lorsque le schéma global est reconnu.

La première solution est plus simple à mettre en œuvre, cependant elle alourdit la tâche du gestionnaire et nécessite plus d'échanges d'information que la deuxième solution. En effet, à chaque fois qu'un agent local envoie une suspicion à l'agent gestionnaire, celui-ci envoie de nouveaux schémas locaux à détecter. Alors que dans la seconde solution, le gestionnaire n'intervient qu'à deux reprises : envoie d'un schéma local et réception d'une alerte.

Rôle de la suspicion dans la modification des schémas

Lorsqu'un agent gestionnaire ou local reçoit une suspicion, il modifie son schéma global ou local :

- dans le premier mode de fonctionnement, la réception d'une suspicion entraîne une modification du schéma global ;
- dans le deuxième mode, se sont les schémas locaux (rattachés au même schéma global) qui sont modifiés. Cela suppose une dynamique des schémas pour pouvoir prendre en compte ces modifications.

Nous avons vu que, lors du processus de filtrage et plus exactement lors de la sélection des suites d'événements (voir Chapitre 4), des lignes dev_z du tableau symbolique étaient créées ou modifiées en créant ainsi de nouvelles instances du schéma d'attaque. Rappelons que les lignes dev_z permettent de spécifier les valeurs de certains attributs sur lesquels ont été appliqués les opérateurs portant sur les attributs d'un événement (par exemple c'est à ce moment là qu'est spécifiée l'adresse source si l'opérateur appliqué à cet attribut est "*même source*").

Lorsqu'un agent reçoit une suspicion il crée de nouvelles instances (en créant de nouvelles lignes dev_z) en spécifiant les valeurs des attributs sur lesquels ont été appliqués les opérateurs (exemple : la source ou la destination) avec celles figées dans la suite d'événements envoyée par l'agent, auteur de la suspicion.

Dans les sections suivantes, nous décrivons les processus de dérivation et de reconnaissance d'un schéma global qui sont différents suivant le type d'opérateurs (itération, séquence, etc.) définis dans un schéma global.

7.2.1 L'opérateur d'itération

Pour le traitement de l'opérateur d'itération, nous tenons compte de la fréquence spécifiée par l'opérateur.

Dérivation du schéma global

La dérivation de l'opérateur d'itération entraîne les changements suivants :

- syntaxiquement, l'attribut fréquence est le même dans la classe schéma et la classe sous-schéma. Cependant sémantiquement, nous l'appelons fréquence_{globale} dans le schéma global et fréquence_{locale} dans le schéma local ;
- nous rajoutons dans la classe sous-schéma :
 - un attribut supplémentaire, que nous appellons *niveau critique*. Cet attribut est initialisé à la valeur de la fréquence_{globale} et nous servira à recalculer la valeur de la fréquence_{locale}. A chaque modification de la fréquence_{locale}, ce *niveau critique* est recalculé mais nous verrons cela plus en détail dans la section suivante ;
 - la référence de la classe schéma d'où a été dérivée la classe sous-schéma. Cette référence a été rajoutée afin de permettre aux agents de se coordonner et corréler leurs résultats ;
 - deux méthodes : *calculerFrequence()* et *calculerNiveauCritique()* qui nous serviront à calculer la fréquence_{locale} et le *niveau critique*.

Le processus de dérivation se traduit de la manière suivante :

- lorsqu'un agent gestionnaire dérive un schéma global en schémas locaux, il modifie la valeur de la fréquence. La fréquence_{locale} est alors égale à la valeur entière supérieure du résultat de la division de la fréquence_{globale} par le nombre d'agents à qui doivent être envoyés les schémas locaux (dont l'agent gestionnaire est responsable).

$$Fréquence_{locale} = \left\lceil \frac{Fréquence_{globale}}{Nombre_{agents}} \right\rceil$$

Nous verrons dans la section suivante que cette fréquence_{locale} est modifiée à chaque fois qu'un agent local génère une *suspicion* ;

- l'agent gestionnaire envoie ensuite les schémas locaux aux différents agents.

Reconnaissance du schéma global

Nous décrivons le traitement de l'opérateur de fréquence pour les deux modes de fonctionnement, puis nous proposons une troisième solution :

Première solution

Dans le premier mode de fonctionnement (agent gestionnaire):

- à chaque fois qu'un agent détecte que la fréquence_{locale} est atteinte, il envoie une suspicion à l'agent gestionnaire;

- celui-ci calcule une fréquence_{globale} intermédiaire, que nous notons fréquence_{globale-intermédiaire}, (initialement égale à la fréquence globale) et une nouvelle fréquence_{locale} :

$$Fréquence_{globale\text{intermédiaire}} = Fréquence_{globale\text{intermédiaire}} - Fréquence_{locale}$$

$$Fréquence_{locale} = \left\lceil \frac{Fréquence_{globale\text{intermédiaire}}}{Nombre_{agents}} \right\rceil$$

- puis, il envoie un nouveau schéma local avec la nouvelle fréquence_{locale};
- les agents auront alors à détecter une nouvelle fréquence_{locale} pour le même schéma global.

Ce calcul se poursuit jusqu'à ce que la fréquence_{globale} soit atteinte. En d'autres termes jusqu'à ce que la fréquence_{globale-intermédiaire} soit nulle. Le schéma global est reconnu, lorsque la fréquence_{globale} est atteinte.

Deuxième solution

Pour ce qui est du deuxième mode de fonctionnement, nous avons introduit, lors du processus de dérivation (paragraphe 7.2.1), un nouvel attribut dans le schéma local qui est le *niveau critique*. En réalité, cet attribut joue le rôle de "fréquence_{globale-intermédiaire}", qui a été vu précédemment. C'est vrai que nous pourrions nous poser la question pourquoi ne pas avoir utilisé, comme dans le premier mode de traitement, la fréquence_{globale-intermédiaire} ? En fait c'est par souci de validité sémantique que nous avons préféré différencier les deux terminologies puisque, a priori, il n'y a que l'agent gestionnaire qui a connaissance du schéma global et que les autres n'ont qu'une vue locale de ce schéma. Voyons maintenant comment se déroule la reconnaissance d'un schéma global :

Lorsqu'un agent reconnaît un schéma local (répétition d'un même événement) :

- il modifie la valeur de la fréquence_{locale} et celle du *niveau critique* de son schéma local. Les modifications se traduisent par un recalcul des valeurs, de la manière suivante :

$$Niveau\ critique = Niveau\ critique - Fréquence_{locale}$$

$$Fréquence_{locale} = \left\lceil \frac{Niveau\ critique}{Nombre_{agentsvoisins} + 1} \right\rceil$$

- puis, il envoie aux autres agents, une suspicion avec la suite d'événements qu'il a reconnue et les nouvelles valeurs de la fréquence_{locale} et du *niveau critique* ;
- les agents mettent alors à jour la fréquence_{locale} et le *niveau critique*. La modification de ces deux valeurs se fait à chaque réception d'une nouvelle suspicion par rapport à la même instance de schéma local. Ce processus se poursuit en effectuant les mêmes opérations de calcul sur la fréquence_{locale} et le *niveau critique* jusqu'à ce que celui-ci soit nul ;
- lorsque le *niveau critique* est nul, le dernier agent qui aura fait ce calcul, envoie un message avec une forte suspicion d'attaque à l'agent gestionnaire afin que ce dernier puisse confirmer que c'est une attaque ;
- l'agent gestionnaire vérifie alors qu'il y a bien une reconnaissance du schéma global. En réalité, il compare la valeur de la fréquence_{globale} et celle du nombre d'occurrences des événements remontés par l'agent auteur de la suspicion. Dans certains cas, le gestionnaire peut, avant de confirmer l'attaque, demander à un autre agent un complément d'information sur un attribut particulier de la suite d'événements.

Nous illustrons cela avec un exemple. Prenons l'exemple du schéma d'attaque : "tentatives répétées de login" avec trois agents locaux.

Etape 0 :

Schéma global :

$$Fréquence_{globale} = 11$$

Schéma local :

$$Fréquence_{locale} = \left\lceil \frac{Fréquence_{globale}}{Nombre_{agentsLocaux}} \right\rceil = 4$$

$$Niveau_{critique} = Fréquence_{locale} = 11$$

Etape 1 :

Dès qu'une suite d'événements contient quatre tentatives de logins, l'agent qui reconnaît cette suite envoie le message de suspicion avec les valeurs :

$$\text{Niveau critique} = \text{Niveau critique} - \text{Fréquence}_{\text{locale}} = 11 - 4 = 7$$

$$\text{Fréquence}_{\text{locale}} = \left\lfloor \frac{\text{Niveau critique}}{\text{Nombre}_{\text{agentsvoisins} + 1}} \right\rfloor = \left\lfloor \frac{7}{3} \right\rfloor = 3$$

Etape 2 :

Le prochain agent qui atteint cette fréquence, envoie une suspicion avec les valeurs :

$$\text{Niveau critique} = 7 - 3 = 4$$

$$\text{Fréquence}_{\text{locale}} = \left\lfloor \frac{4}{3} \right\rfloor = 2$$

Etape 3 :

La prochaine reconnaissance de schéma local donne les résultats suivants :

$$\text{Niveau critique} = 4 - 2 = 2$$

$$\text{Fréquence}_{\text{locale}} = \left\lfloor \frac{2}{3} \right\rfloor = 1$$

Etape 4 :

Lors de cette étape, nous avons les valeurs :

$$\text{Niveau critique} = 2 - 1 = 1$$

$$\text{Fréquence}_{\text{locale}} = \left\lfloor \frac{1}{3} \right\rfloor = 1$$

Etape 5 :

Le prochain agent qui observe cette fréquence calcule le nouveau *niveau critique* :

$$\text{Niveau critique} = 1 - 1 = 0$$

La valeur du *niveau critique* étant nulle, le processus s'arrête et l'agent envoie une dernière suspicion, concernant cette instance de schéma local, à l'agent gestionnaire.

Celui-ci confirme que le schéma global est reconnu et que donc une attaque s'est produite.

Dans cette solution, nous utilisons un processus de calcul récurrent, qui suppose qu'il n'y a pas de délais de transmission de messages et que les informations sont reçues de manière instantanée. Or cela est, dans la pratique, irréalisable car nous savons bien qu'il y a des problèmes de délai de coordination. Nous proposons alors une troisième façon de faire.

Troisième solution

Dans cette troisième solution, nous limitons le processus de calcul à deux étapes. Nous calculons deux valeurs identiques de fréquence_{locale} :

- lors de la première étape, la fréquence_{locale} est calculée de la même façon que précédemment ;
- dès que cette fréquence_{locale} est atteinte, une suspicion est envoyée à tous les agents voisins.
- lorsque la deuxième fréquence_{locale} est atteinte, une suspicion est envoyée à l'agent gestionnaire.

Si nous reprenons l'exemple précédent, cela donne :

Etape 0 :

Schéma global :

$$Fréquence_{globale} = 11$$

Schéma local :

$$Fréquence_{locale} = \left\lceil \frac{Fréquence_{globale}}{Nombre_{agentslocaux}} \right\rceil = 4$$

$$Niveau\ critique = Fréquence_{locale} = 11$$

Etape 1 :

Dès qu'une suite d'événements contient quatre tentatives de login, l'agent qui reconnaît cette suite envoie un message de suspicion. A la réception de ce message de suspicion, les agents se mettent en attente de l'observation de cette même fréquence (égale à 4). Dès qu'un agent observe cette valeur, il envoie une suspicion à l'agent gestionnaire.

Les solutions 2 et 3 ne sont pas équivalentes car le schéma global n'est pas atteint avec le même nombre d'occurrences des événements. Dans la solution 2, le nombre d'occurrences est égal à la fréquence_{globale} spécifiée dans le schéma global. Tandis que dans la solution 3, le nombre d'occurrences varie dans un intervalle de valeurs que nous discuterons dans la section 7.3.

7.2.2 L'opérateur de séquence

Pour l'opérateur de séquence, le traitement est beaucoup plus simple que pour l'opérateur d'itération car il n'y a pas de fréquence à considérer et donc aucun calcul à faire.

Dérivation du schéma global

Pour la dérivation d'un schéma global représenté par une séquence d'événements, le schéma local est identique au schéma global. Par exemple :

Si schéma global = e_1 suit e_2 suit e_3
alors schéma local = e_1 suit e_2 suit e_3

Reconnaissance du schéma global

Pour l'opérateur de séquence, nous considérons que le deuxième mode de traitement (i.e. coopération directe entre les agents sans l'intervention de l'agent gestionnaire) car le premier mode de traitement engendrerait plus d'échanges d'information.

Pour la reconnaissance d'un schéma global représenté par une séquence d'événements, l'ordre temporel des événements est trivial. Par conséquent, lors de la coopération entre agents pour la reconnaissance du schéma global, nous considérons deux cas de figure :

- transmission immédiate de messages entre agents,
- et délai de transmission de messages entre agents.

Lorsque les schémas locaux (identiques au schéma global) sont créés par l'agent gestionnaire, il les envoie aux différents agents. Puis, les agents coopèrent de la manière suivante :

- chaque agent aura alors à observer le premier événement de la séquence ;
- dès qu'un agent observe cet événement, il informe les agents voisins :
 - que cet événement a été observé,
 - et qu'il faut maintenant observer le second événement ;
- pour les autres événements, nous avons deux traitements possibles :
 - si nous considérons qu'il n'y a pas de délai de transmission de messages, alors à chaque fois qu'un événement est observé, il en informe directement les autres agents afin qu'ils observent l'événement suivant ;

- sinon, à chaque fois qu'un agent observe l'événement attendu, il vérifie d'abord que la date de l'événement est postérieure à la date de l'événement précédent. Si c'est le cas, il informe les autres agents, sinon il ne fait rien ;
- ce processus se poursuit jusqu'à l'observation de toute la séquence et cela dans la fenêtre d'observation définie dans le schéma global.

Cas particulier

Lorsqu'un agent observe une sous-séquence du schéma local, il la conserve jusqu'à l'observation de l'événement attendu. Puis, si l'événement suivant à observer correspond au premier événement de la séquence observée, alors l'agent envoie toute la séquence observée. Chaque agent aura alors à observer l'événement suivant la sous-séquence déjà observée. Prenons un exemple :

Soit le schéma global = e_1 *précède* e_2 *précède* e_3 *précède* e_4 *précède* e_5 *précède* e_6
alors schéma local = e_1 *précède* e_2 *précède* e_3 *précède* e_4 *précède* e_5 *précède* e_6

Lorsque les agents reçoivent le schéma local, ils auront à observer le premier événement e_1 , mais ils devront aussi conserver toutes les sous-séquences suivantes observées avant e_1 .

Etape 0 :

Le premier événement à observer est e_1 .

Etape 1 :

Dès qu'un agent observe e_1 , il en informe les autres agents. Le prochain événement à observer est e_2 . Supposons qu'entre temps un agent observe la sous-séquence e_4 e_5 . Dans ce cas, cet agent conserve cette sous-séquence en mémoire, jusqu'à l'observation de e_2 ou de e_3 ou de l'expiration de la fenêtre temporelle.

Etape 2 :

Lorsque l'événement e_2 et e_3 sont observés, l'agent en possession de la sous-séquence e_4 e_5 en informe les autres agents. Tous les agents se mettront alors en attente d'observer l'événement e_6 .

7.2.3 les opérateurs non séquentiels

Pour les opérateurs non séquentiels, l'ordre temporel des événements à observer n'a aucune importance. La seule contrainte temporelle est celle de la fenêtre d'observation.

Dérivation du schéma global

Pour la dérivation des autres opérateurs, le schéma local est identique au schéma global.

Reconnaissance du schéma global

Pour la même raison que pour l'opérateur de séquence, nous ne considérons ici que le deuxième mode de traitement.

Etant donné que pour les opérateurs non séquentiels, l'ordre temporel des événements n'a aucune importance, les agents coopèrent de la manière suivante :

- lorsque les agents reçoivent les schémas locaux, ils se mettent en attente d'observer l'un des événements représentant le schéma local ;
- dès qu'un agent observe un des événements, il en informe les autres ;
- chaque agent devra alors observer l'un des événements restants ;
- ce processus se poursuit jusqu'à l'observation de toute la suite ou de l'expiration de la fenêtre temporelle d'observation.

Prenons un exemple :

Soit le schéma global = e_1 *et* e_2 *et* e_3

alors schéma local = e_1 *et* e_2 *et* e_3

Etape 0 :

Le premier événement à observer est l'un des événements e_1 , e_2 , e_3 .

Etape 1 :

Supposons qu'un agent observe e_2 . Le prochain événement à observer sera e_1 ou e_3 .

Etape 2 :

Si un agent observe l'événement e_1 alors le prochain événement à observer sera e_3 .

Etape 3 :

Le schéma local est identifié, dès que l'événement e_3 est observé. L'agent ayant reconnu son schéma local, en informe l'agent gestionnaire.

7.2.4 Combinaison d'opérateurs

Dans les deux sections précédentes, nous avons vu comment traiter les différents opérateurs mais cela s'applique au cas où le schéma d'attaque global se compose d'une suite d'événements liés par des opérateurs de même type. Que se passe-t-il lorsque dans cette suite il y a différents types d'opérateurs ? Deux cas peuvent se présenter :

1. le schéma global contient une combinaison de deux ou plus opérateurs sauf l'opérateur d'itération ;
2. le schéma global contient une combinaison de deux ou plus opérateurs y compris celui d'itération.

Dérivation du schéma global

Pour le processus de dérivation :

1. si le schéma global ne contient pas d'opérateur d'itération, le schéma local est identique au schéma global ;
2. sinon, la dérivation du schéma global entraîne les modifications vues précédemment (i.e. modification de la valeur de la fréquence_{locale}).

Reconnaissance du schéma global

Pour la reconnaissance du schéma global :

1. si le schéma global ne contient pas d'opérateur d'itération, le traitement est identique à celui décrit pour les opérateurs de séquence et les opérateurs non séquentiels. Prenons un exemple :

Soit le schéma global = e_1 **précède** (e_2 **et** e_3)
alors schéma local = e_1 **précède** (e_2 **et** e_3)

Etape 0:

Le premier événement à observer est l'événement e_1 .

Etape 1:

Une fois que l'événement e_1 , est observé, le prochain événement à traiter sera e_2 ou e_3 .

Etape 2:

Si un agent observe l'événement e_2 alors le prochain événement à observer sera e_3 .

Etape 3:

Le schéma local est identifié, dès que l'événement e_3 est observé. L'agent ayant reconnu son schéma local, en informe l'agent gestionnaire.

2. si le schéma global contient un opérateur d'itération, le traitement est identique à celui décrit pour les opérateurs de séquence et les opérateurs non séquentiels. Prenons un exemple :

Soit le schéma global = **3 fois** e_1 **précède** e_2 **et** e_3
alors schéma local = **3 fois** e_1 **précède** e_2 **et** e_3

Etape 0:

A la première étape il faudra traiter l'opérateur d'itération tel qu'il a été vu précédemment.

Etape 1:

Une fois que la sous-suite **3 fois** e_1 a été identifiée, la reconnaissance du schéma global sera identique au cas précédent (cas où le schéma global ne contient pas d'opérateur d'itération).

7.2.5 *Phénomène de l'oubli*

Dans le deuxième mode de traitement, nous avons vu qu'à chaque fois qu'un agent reconnaît un schéma local, il envoie un message de suspicion avec :

- soit la nouvelle valeur du niveau critique et de la fréquence locale ;
- soit uniquement l'événement observé.

Un problème se pose lorsque l'un des événements qui a généré la suspicion n'est plus dans la fenêtre temporelle d'observation courante. En effet, lorsque ce cas se produit :

- cet événement doit être oublié,
- et la suspicion doit être annulée auprès de tous les agents.

Nous voilà donc face à un problème d'annulation de la suspicion auprès des agents ayant reçu cette suspicion, lorsqu'un événement doit être oublié. L'annulation d'une suspicion signifie un retour vers l'objectif précédent, i.e. la reconnaissance du schéma local précédent (ou de la sous-suite précédente). Pour résoudre ce problème, notamment, celui de retour arrière des objectifs, nous proposons deux mécanismes :

1. l'envoi d'un message d'annulation de suspicion par l'agent ayant généré la suspicion, qui va permettre :
 - soit de remonter le niveau critique et la fréquence locale en remettant les valeurs relatives à l'objectif précédent ;
 - soit de reconsidérer le schéma local représenté par la suite précédente ;
2. le second mécanisme concerne l'utilisation de la durée de vie des croyances. Une suspicion est générée par une croyance et lorsque cette croyance n'est plus valide temporellement (fin de sa durée de vie), la suspicion contenue dans cette croyance n'est plus valide. Par conséquent, la suspicion est annulée.

Dans les deux cas, lorsque la suspicion est annulée, il y a un retour en arrière des buts à atteindre par les agents.

Notre choix s'est porté sur le deuxième mécanisme car la validité d'une suspicion est étroitement liée à la durée de vie de la croyance qui l'a générée.

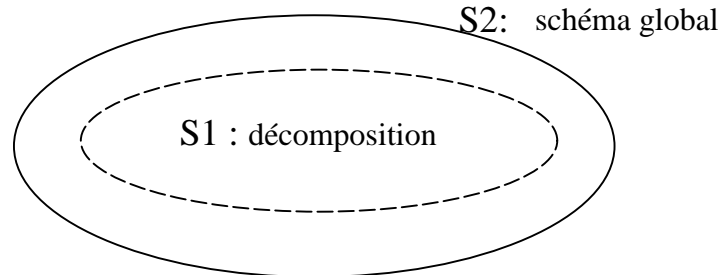
7.3 *Détection centralisée vs distribuée*

Dans cette section nous voulons voir si l'approche distribuée de détection est équivalente à l'approche centralisée. Nous discutons de l'espace de couverture de détection des attaques qui dépend de la politique de décomposition, qui peut être : *minimaliste*, *maximaliste* ou *intermédiaire*. Soit :

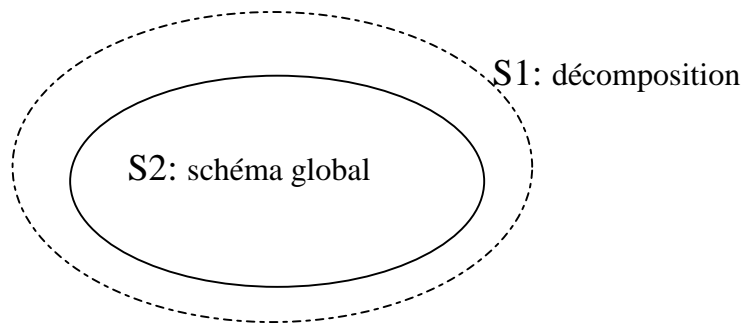
- S1 la couverture des attaques détectées par l'approche distribuée,

- S2 la couverture des attaques détectées en centralisé.

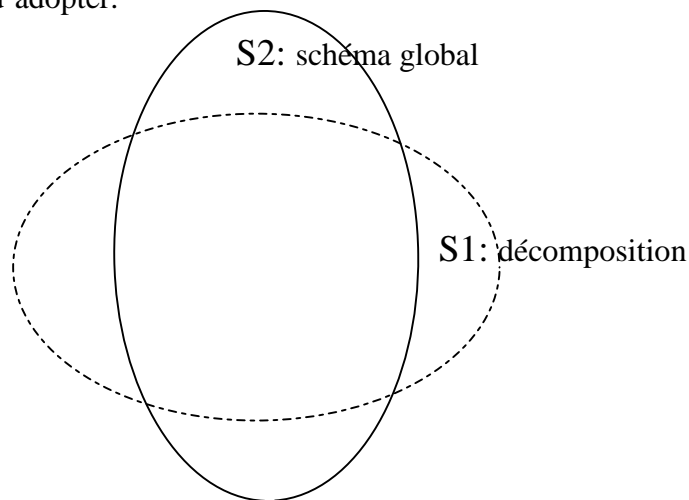
Une *politique minimaliste* a pour but de maximiser la taille qui se trouve à l'intérieur (S1). En terme de détection cela signifie que les attaques sont détectées mais que certaines attaques pourraient passer au travers (perte d'alarmes) ;



Une *politique maximaliste* a pour but de minimiser la taille de S1, de telle sorte que le domaine S2 soit recouvert. Dans ce cas nous avons une détection de toutes les attaques mais aussi des fausses attaques (fausses alarmes).



Enfin, une *politique intermédiaire* a pour but d'augmenter S1 pour essayer de se rapprocher le plus de S2. Cela signifie détecter tous les schémas d'attaques prédéfinis mais aussi limiter les fausses alarmes (fausses attaques). C'est la politique que nous avons choisi d'adopter.



La deuxième approche que nous avons proposée pour le traitement de l'opérateur d'itération (voir 7.2.1), est une solution, en théorie, parfaite car elle est

rigoureusement équivalente à une solution centralisée (l'attaque est détectée lors de l'atteinte de la fréquence globale). Malheureusement, cette solution est quelque peu irréaliste car, dans la pratique, il y a des incertitudes temporelles et des délais de transmission. Sur un plan opérationnel, nous avons donc opté pour la troisième solution. Cependant, bien que la décomposition du schéma global en sous-schémas locaux permette de simplifier la détection des attaques, elle n'offre pas une équivalence rigoureuse dans le cas de la solution choisie. En fait, suivant la politique de décomposition, en ce qui concerne l'affectation des valeurs de fréquences locales, la couverture de détection des attaques n'est pas la même. C'est :

- soit une couverture partielle (minimaliste),
- soit une couverture qui déborde (maximaliste).

Dans la solution choisie, nous avons opté pour deux niveaux de fréquence locale tels que :

$$Fréquence_{locale1} = \left\lceil \frac{Fréquence_{globale}}{Nombre_{agentsLocaux}} \right\rceil$$

$$Fréquence_{locale2} = Fréquence_{locale1}$$

En réalité, suivant les valeurs que nous pourrions donner à ces deux fréquences, la couverture de détection ne sera pas la même. Nous avons vu que le nombre d'occurrences des événements qui permettrait la reconnaissance d'un schéma local n'était pas forcément égal à la fréquence globale. Ce nombre est contenu dans un intervalle de valeurs, dont nous allons discuter les bornes. Nous proposons trois solutions possibles de calcul pour les deux niveaux de fréquence locale. Nous allons au travers d'un exemple montrer que l'approche centralisée et distribuée ne se recouvrent pas complètement. Reprenons l'exemple traité pour l'opérateur d'itération (voir 7.2.1) mais avec une fréquence_{globale} égale à 10 :

Schéma global

$$Fréquence_{globale} = 10$$

$$Nombre_{agents} = 3$$

Soit les agents ag₁, ag₂, ag₃.

Pour chaque solution, nous allons déterminer les bornes minimales et maximales de l'intervalle de valeurs, en considérant :

- le meilleur cas de figure pour le calcul de la borne inférieure,
- le pire des cas de figure pour le calcul de la borne supérieure.

Première proposition

C'est la troisième solution proposée pour le traitement de l'opérateur d'itération (voir 7.2.1).

Schéma local

$$Fréquence_{locale1} = \left\lceil \frac{10}{3} \right\rceil = 4$$

$$Fréquence_{locale2} = Fréquence_{locale1} = 4$$

Détermination de la borne inférieure (meilleur cas de figure)A l'étape 0

L'agent ag_1 observe \rightarrow 0 événement

L'agent ag_2 observe \rightarrow 0 événement

L'agent ag_3 observe \rightarrow 4 événements

\Rightarrow une suspicion est générée et envoyée aux autres agents

A l'étape 1

L'agent ag_1 observe \rightarrow 0 événement

L'agent ag_2 observe \rightarrow 0 événement

L'agent ag_3 observe \rightarrow 4 événements

\Rightarrow atteinte de la deuxième fréquence locale donc l'agent reconnaît le schéma local

Le nombre total d'occurrences des événements observés est égal à 8 (4 + 4).

Donc la borne inférieure est égale à 8.

Détermination de la borne supérieure (pire cas de figure)A l'étape 0

L'agent ag_1 observe \rightarrow 3 événements

L'agent ag_2 observe \rightarrow 3 événements

L'agent ag_3 observe \rightarrow 4 événements

\Rightarrow une suspicion est générée et envoyée aux autres agents

A l'étape 1

L'agent ag_1 observe \rightarrow 3 événements

L'agent ag_2 observe \rightarrow 3 événements

L'agent ag_3 observe \rightarrow 4 événements

\Rightarrow deuxième fréquence locale atteinte donc l'agent reconnaît le schéma local

Le nombre total d'occurrences des événements observés est égal à 14 (3 + 3 + 4 + 4).

Donc la borne supérieure est égale à 14.

L'intervalle de valeurs pour cette solution est égal à $[8, 14] = [\text{Fréquence}_{\text{globale}} - 2, \text{Fréquence}_{\text{globale}} + 4]$.

Deuxième proposition

Nous proposons d'augmenter la première fréquence locale, ce qui nous donne :

$$\text{Fréquence}_{\text{locale1}} = \left\lceil \frac{\text{Fréquence}_{\text{globale}}}{\text{Nombre}_{\text{agentsStocaux}}} \right\rceil + 1$$

$$\text{Fréquence}_{\text{locale2}} = \left\lceil \frac{\text{Fréquence}_{\text{globale}}}{\text{Nombre}_{\text{agentsStocaux}}} \right\rceil$$

Schéma local

$$\text{Fréquence}_{\text{locale1}} = \left\lceil \frac{10}{3} \right\rceil + 1 = 5$$

$$\text{Fréquence}_{\text{locale2}} = \left\lceil \frac{10}{3} \right\rceil = 4$$

Détermination de la borne inférieure (meilleur cas de figure)

A l'étape 0

L'agent ag₁ observe → 0 événement

L'agent ag₂ observe → 0 événement

L'agent ag₃ observe → 5 événements ⇒ une suspicion est générée et envoyée aux autres agents

A l'étape 1

L'agent ag₁ observe → 0 événement

L'agent ag₂ observe → 0 événement

L'agent ag₃ observe → 4 événements ⇒ atteinte de la deuxième fréquence locale donc l'agent reconnaît le schéma local

Le nombre total d'occurrences des événements observés est égal à 9 (5 + 4).

Donc la borne inférieure est égale à 9.

Détermination de la borne supérieure (pire cas de figure)A l'étape 0L'agent ag_1 observe \rightarrow 3 événementsL'agent ag_2 observe \rightarrow 3 événementsL'agent ag_3 observe \rightarrow 5 événements \Rightarrow une suspicion est générée et envoyée aux autres agentsA l'étape 1L'agent ag_1 observe \rightarrow 3 événementsL'agent ag_2 observe \rightarrow 3 événementsL'agent ag_3 observe \rightarrow 4 événements \Rightarrow deuxième fréquence locale atteinte donc l'agent reconnaît le schéma local

Le nombre total d'occurrences des événements observés est égal à 15 (3 + 3 + 5 + 4).

Donc la borne supérieure est égale à 15.

L'intervalle de valeurs pour cette solution est égal à $[9, 15] = [\text{Fréquence}_{\text{globale}} - 1, \text{Fréquence}_{\text{globale}} + 5]$.Troisième proposition

Nous proposons de diminuer la deuxième fréquence locale, ce qui nous donne :

$$\text{Fréquence}_{\text{locale1}} = \left\lfloor \frac{\text{Fréquence}_{\text{globale}}}{\text{Nombre}_{\text{agentsLocaux}}} \right\rfloor$$

$$\text{Fréquence}_{\text{locale2}} = \left\lfloor \frac{\text{Fréquence}_{\text{globale}}}{\text{Nombre}_{\text{agentsLocaux}}} \right\rfloor$$

Schéma local

$$\text{Fréquence}_{\text{locale1}} = \left\lfloor \frac{10}{3} \right\rfloor = 4$$

$$\text{Fréquence}_{\text{locale2}} = \left\lfloor \frac{10}{3} \right\rfloor = 3$$

Détermination de la borne inférieure (meilleur cas de figure)A l'étape 0L'agent ag_1 observe \rightarrow 0 événementL'agent ag_2 observe \rightarrow 0 événement

L'agent ag_3 observe \rightarrow 4 événements \Rightarrow une suspicion est générée et envoyée aux autres agents

A l'étape 1

L'agent ag_1 observe \rightarrow 0 événement

L'agent ag_2 observe \rightarrow 0 événement

L'agent ag_3 observe \rightarrow 3 événements \Rightarrow atteinte de la deuxième fréquence locale donc l'agent reconnaît le schéma local

Le nombre total d'occurrences des événements observés est égal à 7 (4 + 3).

Donc la borne inférieure est égale à 7.

Détermination de la borne supérieure (pire cas de figure)

A l'étape 0

L'agent ag_1 observe \rightarrow 2 événements

L'agent ag_2 observe \rightarrow 2 événements

L'agent ag_3 observe \rightarrow 4 événements \Rightarrow une suspicion est générée et envoyée aux autres agents

A l'étape 1

L'agent ag_1 observe \rightarrow 2 événements

L'agent ag_2 observe \rightarrow 2 événements

L'agent ag_3 observe \rightarrow 3 événements \Rightarrow deuxième fréquence locale atteinte donc l'agent reconnaît le schéma local

Le nombre total d'occurrences des événements observés est égal à 11 (2 + 2 + 4 + 3).

Donc la borne supérieure est égale à 11.

L'intervalle de valeurs pour cette solution est égal à $[7, 11] = [\text{Fréquence}_{\text{globale}} - 3, \text{Fréquence}_{\text{globale}} + 1]$.

Discutons maintenant des valeurs des trois intervalles que nous venons de calculer, pour voir quelle serait la solution la plus optimale. Soit I_1 , I_2 et I_3 ces trois intervalles :

$$I_1 = [8, 14] ; I_2 = [9, 15] \text{ et } I_3 = [7, 11]$$

La borne inférieure de ces intervalles représente à quelle fréquence sera reconnue une attaque qui pourrait ne pas en être une. Quant à la borne supérieure, elle représente le nombre d'occurrences maximal au bout duquel une attaque serait détectée.

$[b_{\min}, \text{Fréquence}_{\text{globale}}[$ représente alors l'intervalle de fausses attaques (fausses alarmes) et $]\text{Fréquence}_{\text{globale}}, b_{\max}[$, l'intervalle d'attaques non détectées.

La solution la plus optimale serait celle dont la longueur (L) des deux intervalles serait la plus petite. Si nous calculons la longueur pour chaque intervalle, nous aurions :

Pour I_1 : $[8, 10[$ $L_{\min} = 2$
 $]10, 14]$ $L_{\max} = 4$

Pour I_2 : $[9, 10[$ $L_{\min} = 1$
 $]10, 15]$ $L_{\max} = 5$

Pour I_3 : $[7, 10[$ $L_{\min} = 3$
 $]10, 11]$ $L_{\max} = 1$

Nous remarquons que I_2 et I_3 ont les petites longueurs L_{\min} et L_{\max} . Ce qui signifie qu'aucune des solutions ne peut offrir en même temps les longueurs L_{\min} et L_{\max} les plus petites. Il nous faut donc faire un compromis. Nous pensons qu'il vaut mieux choisir la solution qui a le plus petit intervalle d'attaques non détectées et donc la plus petite longueur L_{\max} car le problème de fausses alarmes pourrait être géré par le gestionnaire qui en comparant la fréquence globale et le nombre réelle d'occurrences d'événements pourra confirmer ou pas que c'est une attaque. Dans notre cas donc, nous opterons pour la troisième solution donc celui dont l'intervalle est I_3 .

Le but des solutions proposées a été de se rapprocher le plus possible de la couverture initiale (celle du schéma global). La discussion a porté sur le choix des valeurs des deux niveaux de fréquence locale. Suivant le choix des valeurs, nous n'avions pas les mêmes couvertures.

Nous avons vu que malgré la méthode de décomposition adoptée dans la section 7.2.1 (troisième solution), nous n'arrivons pas à une couverture rigoureuse. Le paramétrage étant très sensible à la couverture, il n'y a pas de réelle équivalence. En fait nous avons trois cas de figure :

- soit nous ne détectons pas tout,
- soit nous détectons plus que ce qui est détectable,
- enfin soit nous détectons pratiquement tout en limitant ce qui ne doit pas être détecté.

7.4 Délégation des buts de détection

Dans les sections précédentes, nous avons présenté le processus de dérivation et de traitement des schémas locaux. L'objectif de ce processus est de pouvoir spécifier aux différents agents les schémas d'attaques qu'ils auraient à détecter localement. Cela est réalisé grâce au mécanisme de délégation :

- lorsque l'agent gestionnaire des politiques reçoit une politique de sécurité, il identifie le (ou les) schéma(s) d'attaque paramétrés par cette politique. Il associe à chaque schéma d'attaque, un *but global* de détection et transforme ainsi la politique reçue en plusieurs *buts globaux* qui seront envoyés aux agents gestionnaires extranet ;

- chaque agent gestionnaire extranet dérive ensuite le schéma global en schémas locaux. Ces schémas locaux sont envoyés sous formes de *buts locaux* délégués aux différents agents intranet ;
- lorsque les agents intranet reçoivent les schémas locaux à détecter, ils réalisent la même opération que les agents gestionnaires extranet.

7.5 Conclusion

Dans ce chapitre, nous avons décrit comment était réalisée la détection des attaques dans un contexte multi-agents distribué. Nous avons présenté le processus de dérivation des schémas d'attaque et leur traitement en vue de les détecter et nous avons discuté différentes solutions. Nous avons vu que la solution choisie (la troisième solution dans le cas de l'opérateur d'itération) n'était pas rigoureusement équivalente à une solution centralisée. Cependant c'est une solution optimale dont l'objectif est de détecter à temps les attaques et de limiter les fausses alarmes. La dérivation d'un schéma global en schémas locaux permet de déléguer ensuite des buts de détection de ces schémas aux différents agents.

Chapitre 8 Implémentation

8.1 Introduction

Nous avons présenté, dans le cadre de ce travail, un modèle à base d'un SMA pour la gestion de la sécurité d'un réseau, plus précisément pour la détection d'intrusions. Dans ce chapitre, nous présentons brièvement l'implémentation de notre modèle, dont une vue générale est représentée dans la Figure 32.

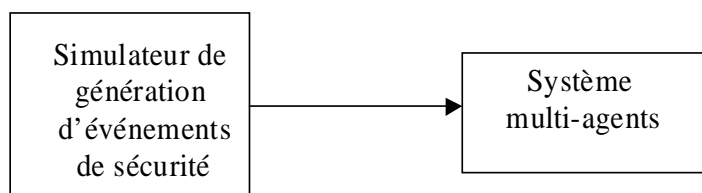


Figure 32: Vue générale de l'implémentation

Nous décrivons l'implémentation :

- du simulateur de messages dont l'objectif est de générer les messages qui seront traités par le SMA ;
- des différents types d'agents du SMA : agents surveillants locaux, agents gestionnaires extranet, agents gestionnaires intranet et agent gestionnaire des politiques de sécurité.

En terme d'expérimentation, nous avons réalisé le simulateur de messages et les agents locaux pour la détection locale des schémas d'attaque. Nous n'avons malheureusement, pour des raisons de temps, pas pu aller au bout de l'implémentation de l'agent gestionnaire des politiques de sécurité, des agents gestionnaires extranet et des agents gestionnaires intranet.

Ce chapitre est organisé de la manière suivante. Dans un premier temps, nous décrivons l'implémentation du simulateur de messages. Puis, nous présentons l'environnement de développement, pour l'implémentation du SMA. Nous décrivons ensuite l'implémentation des différents types d'agents du SMA. Enfin nous finissons par une petite discussion et un exemple de scénario que nous pouvons réaliser.

8.2 *Le simulateur de messages*

Pour la génération des messages, nous avons implémenté un simulateur de messages défini par la classe "*MessagesSimulator*" (voir Figure 33). Cette classe est caractérisée par les listes suivantes :

- liste des machines internes,
- liste des machines externes,
- liste des types d'événements
- liste des noms utilisateurs,
- liste des ports.

```

Public class MessagesSimulator {
.....
.....

public Object createMessage() {
    SecurityEvent createdMessage= new SecurityEvent();
    x = rand.nextInt(4);
    switch (x) {
    case 0: {
        y = rand.nextInt(4); String newType = new String();
        newType = (String)(net.eventTypesList.get(y));
        createdMessage = new SecurityEvent(event,x,newType);
        break; }
    case 1: {
        y = rand.nextInt(13); String newSourceHostId = new String();
        newSourceHostId = (String)(net.externalHostAddressesList.get(y));
        createdMessage = new SecurityEvent(event,x,newSourceHostId);
        break; }
    case 2: {
        y = rand.nextInt(10); String newDestHostId = new String();
        newDestHostId = (String)(net.internalHostAddressesList.get(y));
        createdMessage = new SecurityEvent(event,x,newDestHostId);
        break; }
    case 3: {
        y = rand.nextInt(11); String newDestPortId = new String();
        newDestPortId = (String)(net.portNumbersList.get(y));
        createdMessage = new SecurityEvent(event,x,newDestPortId);
        break; }
    case 4: {
        String newUserName = new String();
        newUserName = (String)(net.userNamesList.get(y));
        createdMessage = new SecurityEvent(event,x,newUserName);
        break; }
    }
    return (createdMessage);
}

```

Figure 33: La méthode de création de messages

La méthode principale de la classe est "*createMessage()*" (voir Figure 33). Cette méthode génère un message de manière aléatoire à partir des listes définies dans la classe.

Dans le noyau du simulateur, cette méthode est activée tous les Δ_t .

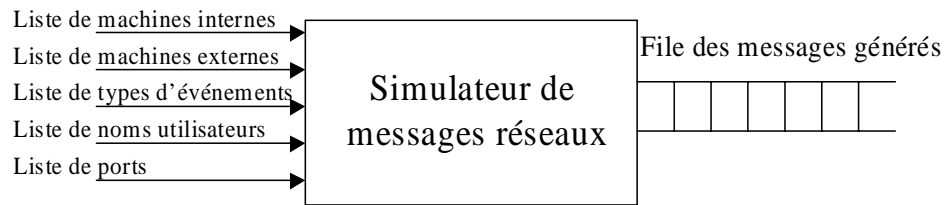


Figure 34 : Simulateur de messages

8.3 *Le système multi-agents*

8.3.1 *L'environnement de développement*

Afin d'implémenter notre modèle, nous avons dû choisir une plate-forme multi-agents opérationnelle, qui offre les caractéristiques des agents de notre modèle : agents hybrides et interactifs. Notre choix s'est porté sur la plate-forme DIMA [Guessoum 1996], développée par Z. Guessoum au LIP6-Paris car elle présente ces caractéristiques. DIMA est une plate-forme multi-agents modulaire et générique qui propose d'étendre le comportement simple d'un objet actif à un ensemble de comportements. Dans DIMA, un agent est une entité pro-active qui n'agit pas simplement en réponse à des messages reçus d'autres agents. En effet, l'agent, par exemple, interagit avec son environnement et délibère pour déterminer l'action la plus appropriée.

L'architecture d'un agent DIMA (voir Figure 35) est composée de deux strates:

- la première strate est constituée de modules interactifs, qui représentent les différents comportements concurrents d'un agent tels que la communication, le raisonnement et la perception. Grâce à ces modules, l'agent peut avoir des propriétés telles que l'autonomie et la coopération. Par exemple, le module de communication gère les interactions entre l'agent et les autres agents du système.
- la seconde strate est composée d'un module de supervision qui permet à l'agent de raisonner sur les états des modules de la première strate et de superviser leurs interactions.

Exemples de comportements

Parmi les comportements implémentés, sous forme de modules, dans DIMA, nous pouvons citer: le module de perception, le module de communication et le module de raisonnement [Guessoum 1996]. Nous décrivons brièvement ces trois modules, dans ce qui suit.

- Le **module de perception** gère les interactions entre l'agent et son environnement. Il permet de mettre à jour un ensemble de données qui regroupe les valeurs de variables externes accessibles à l'aide de capteurs. Ces données représentent un modèle de l'environnement perçue.
- Le **module de communication/action** gère deux fonctions :
 - la réception et l'envoi de messages pour communiquer avec les autres agents;
 - l'exécution d'actions pour agir sur l'environnement.
- Le **module de raisonnement** reflète l'état mental de l'agent (ses plans, ses croyances, ses buts, etc.). Il génère les réponses adéquates aux messages transmis par le *module de communication* et aux changements détectés par le *module de perception*. Il est composé :
 - d'une base de connaissances comprenant les objets qui décrivent l'environnement de l'agent et un ensemble de règles qui représentent les opérations appropriées applicables;
 - d'un moteur d'inférence avec un mécanisme de synchronisation des règles de production;
 - d'une base de méta-connaissances qui permet une représentation déclarative du contrôle du raisonnement.

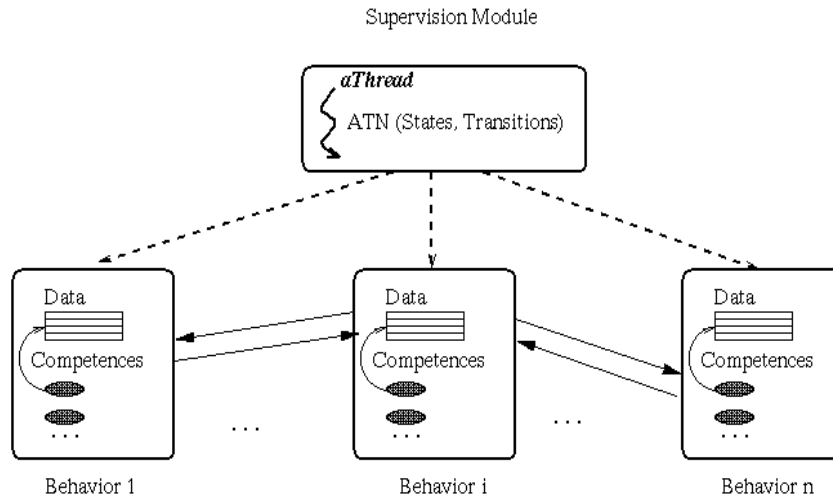


Figure 35 : L'architecture d'un agent hybride dans DIMA

8.3.2 Le système multi- agents

Le système multi-agents décrit dans le Chapitre 6, regroupe trois classes d'agents : les agents surveillants locaux, les agents gestionnaire extranet et intranet et l'agent gestionnaire des politiques de sécurité. Suivant la classe d'agent, nous n'implémentons pas les même modules.

8.3.2.1 Les agents surveillants locaux

Chaque agent surveillant local a quatre modules (voir Figure 36) :

- un module de sélection de messages, qui réalise la sélection des messages à filtrer ;
- un module de filtrage, qui réalise le filtrage des messages ;
- un module de délibération, qui traite les événements filtrés et la reconnaissance des schémas d'attaque ;
- un module de communication, qui permet les communications entre les agents surveillants locaux et entre un agent surveillant local et un agent gestionnaire intranet.

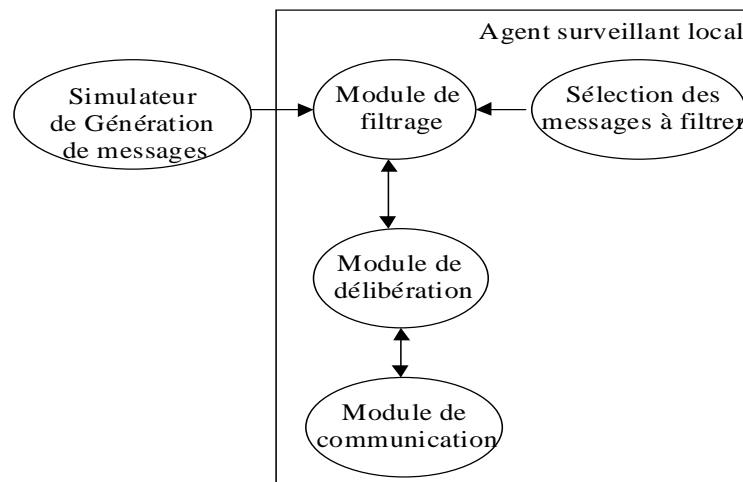


Figure 36: Modules d'un agent surveillant local

Module de sélection des messages

Pour implémenter le module de sélection des messages, nous avons défini quatre schémas d'attaque. Ces schémas ont été définis pour détecter les attaques suivantes :

- "*Doorknob Rattling*" : tentatives répétées de login,
- "*Ping Sweep*" : envoi de requêtes "*Ping*" répétées pour identifier les machines connectées au réseau d'entreprise,
- "*Port Scanning*" : balayage des différents ports ouverts sur une machine,
- "*ICMP flooding*" : envoi de requêtes "*ICMP*" répétées pour surcharger le réseau d'entreprise.

Les schémas d'attaques instanciés sont représentés par les tableaux symboliques suivants (en utilisant le système de représentation par tableau décrit dans 4.3.2.1) :

Schéma d'attaque "Doorknob Rattling"

	Doorknob Rattling	Point Observation	Type	Nom	Source	Destination	Résultat	Activité
5	d _e	traffic réseau	connexion	Telnet	même	n'importe quel	echec	extranet entrant

Schéma d'attaque "Ping Sweep"

	Ping Sweep	Point Observation	Type	Nom	Source	Destination	Résultat	Activité
5	d _e	traffic réseau	connexion	Telnet	même	différent _{att}		extranet entrant

Schéma d'attaque "Port Scanning"

	Port Scanning	Point Observation	Type	Nom	Source	Destination	Résultat	Activité
3	d _e	traffic réseau	connexion	Telnet	même	même		extranet entrant

Schéma d'attaque "ICMP Flooding"

	ICMP Flooding	Point Observation	Type	Nom	Source	Destination	Résultat	Activité
20	d _e	traffic réseau	réseau	ICMP	même	différent _{att}		extranet entrant

A partir de ces schémas instanciés, la méthode "*selectionOfMessagesToFilter*" (voir Figure 37), met à jour la liste des types de messages à filtrer avec les messages de type "TELNET" et "ICMP".


```

public void selectionOfMessagesToFilter(AttackSchema schema) {

    SecurityEvent message = new SecurityEvent();

    for (int i = 0; i < schema.eventDescriptifList.size(); i++) {
        message = (SecurityEvent)(schema.eventDescriptifList.get(i));

        listOfMessagesToFilter.add(message);
    }
}

```

Figure 37: La méthode de sélection de messages à filtrer

Module de filtrage des messages

Le module de filtrage de messages sélectionne tous les messages pertinents, à partir des messages générés. Il exécute la méthode "*extractEvent()*" (voir Figure 39) qui filtre tous les messages de type "TELNET" et "ICMP". A l'issue de ce filtrage, nous obtenons une liste d'événements de sécurité en attente d'être analysés (voir Figure 38).



Figure 38 : Filtrage des messages réseaux

```

public void extractEvent(int ind) {

    collectedEvent = eventGenerator.listOfGeneratedEvent.get(ind);
    securityEvent =(SecurityEvent)(collectedEvent);

    if (isInListOfEventTypesToFilter(securityEvent.eventType)) {

        listOfFilteredEvent.add(collectedEvent);
    }
}

public boolean isInListOfEventTypesToFilter(String eventType){
    return (listOfEventTypesToFilter.contains (eventType));
}

```

Figure 39: La méthode de filtrage d'événements

Module de délibération

Les événements filtrés, sont capturés par les agents locaux et traités par le module de délibération et plus exactement par le moteur de délibération, implémenté par la méthode "*BDIEngine()*" (voir Figure 40).

Le premier traitement que subit un événement de sécurité est effectué par la méthode "*eventDangerTypeFilterOperator()*", qui évalue la nature du danger de l'événement (normal, suspicieux ou attaque) et initialise le champ "*dangerType*".

```

BDIEngine() {

eventDangerTypeFilterOperator(listOfFilteredEvents);
beliefOperator(listOfAnalyzedEvents);
goalOperator(seriesList);
}

public eventDangerTypeFilterOperator(LinkedList listOfFilteredEvents) {

for (int i=0; i< listOfFilteredEvents.size(); i++) {
event = listOfFilteredEvents.get(i);
event.setDangerType();
listOfAnalyzedEvents().add(event);
}
}

public void beliefOperator(LinkedList listOfAnalyzedEvents) {

for (int i=0; i< listOfSchemaAttacks.size(); i++) {
schema = listOfSchemaAttacks.get(i);
schema.extractEvenSeries(listOfAnalyzedEvent);
}
}

public void goalOperator(LinkedList seriesList) {

for (int i=0; i< seriesList.size(); i++) {
serie = seriesList.get(i);
schema = serie.getSchemaReference();
conclusion = schema.evaluateEvenList(serie);
if (conclusion. == "ATTACK") {
message = "There is a ";
message = message.concat(schema.schemaName);
message = message.concat("attack");
sendMessage(message); }
else if (conclusion == "SUSPICION") {
message = "There is a suspicious scenario that looks at";
message = message.concat(schema.schemaName);
message = message.concat("attack");
sendMessage(message);}
}
}

```

Figure 40: Le moteur de délibération

Puis est exécuté l'opérateur Croyance ("*beliefOperator()*"), qui permet d'extraire les différentes suites d'événements en utilisant la méthode "*extractEventSeries()*". Nous obtenons quatre listes, une pour chaque schéma d'attaque que nous voulons détecter. L'opérateur But ("*goalOperator()*") est ensuite exécuté où pour chaque liste est exécutée la méthode "*evaluateEventList()*" pour la reconnaissance d'un schéma d'attaque. Si l'une des suites d'événements analysées ne correspond pas à un schéma d'attaque mais qu'il existe un événement dans cette liste qui est suspicieux alors la liste analysée est suspicieuse.

Module de communication

Le module de communication réalise l'envoi et la réception des messages entre agents. Ce module étant un module générique dans DIMA, indépendant de l'application, nous l'utilisons pour implémenter notre module.

8.3.2.2 Les agents gestionnaires extranet et intranet

Les agents gestionnaires extranet et intranet, n'ont pas de module de filtrage mais les modules suivants (voir Figure 41) :

- un module de délibération, qui permet de gérer les buts et les croyances de l'agent ;
- un module de génération de buts, qui permet de générer des sous-buts, à partir des buts envoyés par l'agent gestionnaire des politiques de sécurité, pour la détection des sous-schémas locaux ;
- un module de communication, qui gère les communications : entre l'agent gestionnaire extranet et l'agent gestionnaire des politiques de sécurité, entre l'agent gestionnaire extranet et les agents gestionnaires intranet, entre les agents gestionnaires intranet et entre l'agent gestionnaire intranet et les agents surveillants locaux.

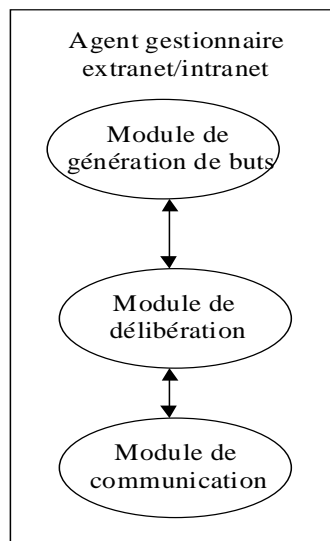


Figure 41: Les modules d'un agent gestionnaire extranet/intranet

8.3.2.3 L'agent gestionnaire des politiques de sécurité

L'agent gestionnaire des politiques de sécurité a cinq modules (voir Figure 42) :

- un module interface utilisateur, qui permet à l'administrateur de spécifier les politiques de sécurité ;
- un module de gestion de politiques de sécurité, qui permet de gérer les politiques de sécurité, notamment la dérivation des politiques d'obligation/autorisation en politiques schémas ;
- un module d'instanciation de schémas d'attaque, qui permet d'instancier les schémas d'attaque, à partir des politiques schémas ;
- un module de génération de buts, qui permet de générer les buts de détection pour la détection des schémas instanciés ;
- un module de communication, qui gère les communications entre l'agent gestionnaire des politiques de sécurité et l'agent gestionnaire extranet.

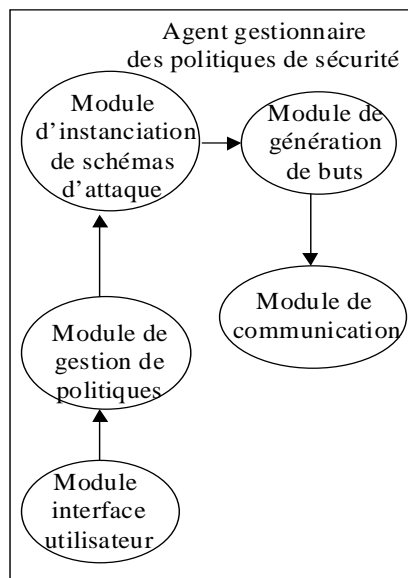


Figure 42 : Les modules d'un agent gestionnaire des politiques de sécurité

8.3.3 Discussion

Nous avons réalisé le simulateur de messages et les agents locaux qui détectent localement des sous-schémas d'attaques. Nous avons ainsi validé :

- les notions de schémas d'attaque, de filtres d'événements, d'événements et de suite d'événements ;
- la fonction de filtrage ;

- la fonction de délibération pour le traitement des événements et la reconnaissance de schémas d'attaque.

A l'issue de cette expérimentation ont été détectés trois schémas d'attaque sur quatre : "*Ping Sweep*", "*Port Scanning*" et "*Doorknob Rattling*".

Voici un exemple d'exécution :

System Output: List of generated events :	System Output: ELEMENT:6
System Output: ELEMENT:1	System Output: event number =:6
System Output: event number =:1	System Output: eventType=:telnet
System Output: eventType=:ICMP	System Output: userName=:deniro
System Output: userName=:deniro	System Output: SourceHostId=:194.143.195.56
System Output: SourceHostId=:212.4.227.4	System Output: DestHostId=:193.55.114.15
System Output: DestHostId=:193.55.114.15	System Output: DestPortId=:513
System Output: DestPortId=:21	System Output: time=:29410006
System Output: time=:29410001	
System Output: ELEMENT:2	System Output: ELEMENT:7
System Output: event number =:2	System Output: event number =:7
System Output: eventType=:ICMP	System Output: eventType=:telnet
System Output: userName=:deniro	System Output: userName=:deniro
System Output: SourceHostId=:212.4.227.4	System Output: SourceHostId=:194.143.195.56
System Output: DestHostId=:193.55.114.15	System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:21	System Output: DestPortId=:514
System Output: time=:29410002	System Output: time=:29410007
System Output: ELEMENT:3	System Output: ELEMENT:8
System Output: event number =:3	System Output: event number =:8
System Output: eventType=:ICMP	System Output: eventType=:telnet
System Output: userName=:deniro	System Output: userName=:deniro
System Output: SourceHostId=:212.4.227.4	System Output: SourceHostId=:194.143.195.56
System Output: DestPortId=:512	System Output: DestHostId=:193.55.114.15
System Output: time=:29410003	System Output: DestPortId=:513
	System Output: time=:29410008
System Output: ELEMENT:4	System Output: ELEMENT:9
System Output: event number =:4	System Output: event number =:9
System Output: eventType=:ICMP	System Output: eventType=:ftp
System Output: userName=:deniro	System Output: userName=:deniro
System Output: SourceHostId=:212.4.227.4	System Output: SourceHostId=:194.143.195.56
System Output: DestHostId=:193.55.114.15	System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:513	System Output: DestPortId=:513
System Output: time=:29410004	System Output: time=:29410009
System Output: ELEMENT:5	System Output: ELEMENT:10
System Output: event number =:5	System Output: event number =:10
System Output: eventType=:ICMP	System Output: eventType=:ftp
System Output: userName=:deniro	System Output: userName=:deniro
System Output: SourceHostId=:194.143.195.56	System Output: SourceHostId=:212.4.227.4
System Output: DestHostId=:193.55.114.15	System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:513	System Output: DestPortId=:513
System Output: time=:29410005	System Output: time=:29410010

<p>System Output: List of filtered events</p> <p>System Output: event number=:1</p> <p>System Output: eventType=:ICMP</p> <p>System Output: userName=:deniro</p> <p>System Output: SourceHostId=:212.4.227.4</p> <p>System Output: DestHostId=:193.55.114.15</p> <p>System Output: DestPortId=:21</p> <p>System Output: Location=:<i>External</i></p> <p>System Output: Direction=:<i>OnGoing</i></p> <p>System Output: time=:29410001</p> <p>System Output: event number =:2</p> <p>System Output: eventType=: ICMP</p> <p>System Output: userName=:deniro</p> <p>System Output: SourceHostId=:212.4.227.4</p> <p>System Output: DestHostId=:193.55.114.15</p> <p>System Output: DestPortId=:21</p> <p>System Output: Location=:<i>External</i></p> <p>System Output: Direction=:<i>OnGoing</i></p> <p>System Output: timeInMillis=:29410002</p> <p>System Output: event number =:3</p> <p>System Output: eventType=: ICMP</p> <p>System Output: userName=:deniro</p> <p>System Output: SourceHostId=:212.4.227.4</p> <p>System Output: DestPortId=:512</p> <p>System Output: Location=:<i>External</i></p> <p>System Output: Direction=:<i>OnGoing</i></p> <p>System Output: time=:29410003</p> <p>System Output: event number =:4</p> <p>System Output: eventType=: ICMP</p> <p>System Output: userName=:deniro</p> <p>System Output: SourceHostId=:212.4.227.4</p> <p>System Output: DestHostId=:193.55.114.15</p> <p>System Output: DestPortId=:513</p> <p>System Output: Location=:<i>External</i></p> <p>System Output: Direction=:<i>OnGoing</i></p> <p>System Output: time=:29410004</p>	<p>System Output: event number=:5</p> <p>System Output: eventType=: ICMP</p> <p>System Output: userName=:deniro</p> <p>System Output: SourceHostId=:194.143.195.56</p> <p>System Output: DestHostId=:193.55.114.15</p> <p>System Output: DestPortId=:513</p> <p>System Output: Location=:<i>External</i></p> <p>System Output: Direction=:<i>OnGoing</i></p> <p>System Output: time=:29410005</p> <p>System Output: event number =:6</p> <p>System Output: eventType=:telnet</p> <p>System Output: userName=:deniro</p> <p>System Output: SourceHostId=:194.143.195.56</p> <p>System Output: DestHostId=:193.55.114.15</p> <p>System Output: DestPortId=:513</p> <p>System Output: Location=:<i>External</i></p> <p>System Output: Direction=:<i>OnGoing</i></p> <p>System Output: time=:29410006</p> <p>System Output: event number =:7</p> <p>System Output: eventType=: telnet</p> <p>System Output: userName=:deniro</p> <p>System Output: SourceHostId=:194.143.195.56</p> <p>System Output: DestHostId=:193.55.114.15</p> <p>System Output: DestPortId=:514</p> <p>System Output: Location=:<i>External</i></p> <p>System Output: Direction=:<i>OnGoing</i></p> <p>System Output: time=:29410007</p> <p>System Output: event number =:8</p> <p>System Output: eventType=: telnet</p> <p>System Output: userName=:deniro</p> <p>System Output: SourceHostId=:194.143.195.56</p> <p>System Output: DestHostId=:193.55.114.15</p> <p>System Output: DestPortId=:513</p> <p>System Output: Location=:<i>External</i></p> <p>System Output: Direction=:<i>OnGoing</i></p> <p>System Output: time=:29410008</p>
--	---

```
System Output: Set danger type

System Output: event number =:1
System Output: eventType=:ICMP
System Output: userName=:deniro
System Output: SourceHostId=:212.4.227.4
System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:21
System Output: Location=:External
System Output: Direction=:OnGoing
System Output: DangerType.type=:SUSPICIOUS
System Output: This event has a suspicious userName ==deniro

System Output: event number =:2
System Output: eventType=:ICMP
System Output: userName=:deniro
System Output: SourceHostId=:212.4.227.4
System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:21
System Output: Location=:External
System Output: Direction=:OnGoing
System Output: DangerType.type=:SUSPICIOUS
System Output: This event has a suspicious userName ==deniro

System Output: event number =:3
System Output: eventType=:ICMP
System Output: userName=:deniro
System Output: SourceHostId=:212.4.227.4
System Output: DestPortId=:512
System Output: Location=:External
System Output: Direction=:OnGoing
System Output: DangerType.type=:SUSPICIOUS
System Output: This event has a suspicious userName ==deniro

System Output: event number =:4
System Output: eventType=:ICMP
System Output: userName=:deniro
System Output: SourceHostId=:212.4.227.4
System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:513
System Output: Location=:External
System Output: Direction=:OnGoing
System Output: DangerType.type=:SUSPICIOUS
System Output: This event has a suspicious userName ==deniro
```

```

System Output: event number=:5
System Output: eventType=:ICMP
System Output: userName=:deniro
System Output: SourceHostId=:194.143.195.56
System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:513
System Output: Location=:External
System Output: Direction=:OnGoing
System Output: DangerType.type=:SUSPICIOUS
System Output: This event has a suspicious userName ==deniro
System Output: This event has a suspicious source ==194.143.195.56

System Output: event number =:6
System Output: eventType=:telnet
System Output: userName=:deniro
System Output: SourceHostId=:194.143.195.56
System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:513
System Output: Location=:External
System Output: Direction=:OnGoing
System Output: DangerType.type=:SUSPICIOUS
System Output: This event has a suspicious userName ==deniro
System Output: This event has a suspicious source ==194.143.195.56

System Output: event number =:7
System Output: eventType=: telnet
System Output: userName=:deniro
System Output: SourceHostId=:194.143.195.56
System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:514
System Output: Location=:External
System Output: Direction=:OnGoing
System Output: DangerType.type=:SUSPICIOUS
System Output: This event has a suspicious userName ==deniro
System Output: This event has a suspicious source ==194.143.195.56

System Output: event number =:8
System Output: eventType=: telnet
System Output: userName=:deniro
System Output: SourceHostId=:194.143.195.56
System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:513
System Output: Location=:External
System Output: Direction=:OnGoing
System Output: DangerType.type=:SUSPICIOUS
System Output: This event has a suspicious userName ==deniro
System Output: This event has a suspicious source ==194.143.195.56

```



```
System Output: Created Series
System Output: Number of created Series = 2

System Output: Series number = 1

System Output: event number=:1
System Output: eventType =:ICMP
System Output: userName =:deniro
System Output: SourceHostId =:212.4.227.4
System Output: DestHostId =:193.55.114.15
System Output: DestPortId =:21
System Output: Location=:External
System Output: Direction=: OnGoing
System Output: DangerType .type=:SUSPICIOUS
System Output: time=:29410001

System Output: event number =:2
System Output: eventType =:ICMP
System Output: userName =:deniro
System Output: SourceHostId =:212.4.227.4
System Output: DestHostId =:193.55.114.15
System Output: DestPortId =:21
System Output: Location=:External
System Output: Direction=: OnGoing
System Output: DangerType .type=:SUSPICIOUS
System Output: timeInMillis =:29410002

System Output: event number =:3
System Output: eventType =:ICMP
System Output: userName =:deniro
System Output: SourceHostId =:212.4.227.4
System Output: DestPortId =:512
System Output: Location=:External
System Output: Direction=: OnGoing
System Output: DangerType .type=:SUSPICIOUS
System Output: time=:29410003

System Output: event number =:4
System Output: eventType =:ICMP
System Output: userName =:deniro
System Output: SourceHostId =:212.4.227.4
System Output: DestHostId =:193.55.114.15
System Output: DestPortId =:513
System Output: Location=:External
System Output: Direction=: OnGoing
System Output: DangerType .type=:SUSPICIOUS
System Output: time=:29410004

System Output: event number=:5
System Output: eventType =:ICMP
System Output: userName =:deniro
System Output: SourceHostId =:194.143.195.56
System Output: DestHostId =:193.55.114.15
System Output: DestPortId =:513
System Output: Location=:External
System Output: Direction=: OnGoing
System Output: DangerType .type=:SUSPICIOUS
System Output: time=:29410005

System Output: There is a Ping Sweep attack
```

```
System Output: Series number = 2

System Output: event number =:6
System Output: eventType=:telnet
System Output: userName=:deniro
System Output: SourceHostId=:194.143.195.56
System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:513
System Output: Location=:External
System Output: Direction=:OnGoing
System Output: DangerType.type=:SUSPICIOUS
System Output: time=:29410006

System Output: event number =:7
System Output: eventType=: telnet
System Output: userName=:deniro
System Output: SourceHostId=:194.143.195.56
System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:514
System Output: Location=:External
System Output: Direction=:OnGoing
System Output: DangerType.type=:SUSPICIOUS
System Output: time=:29410007

System Output: event number =:8
System Output: eventType=: telnet
System Output: userName=:deniro
System Output: SourceHostId=:194.143.195.56
System Output: DestHostId=:193.55.114.15
System Output: DestPortId=:513
System Output: Location=:External
System Output: Direction=:OnGoing
System Output: DangerType.type=:SUSPICIOUS
System Output: time=:29410008

System Output: There is a Port Scanning attack
```

Par manque de temps nous n'avons malheureusement pas réalisé l'expérimentation des agents gestionnaires extranet/intranet et de l'agent gestionnaire des politiques de sécurité.

8.3.4 Exemple de scénario

Notre expérimentation a été réalisée, en vue d'implémenter le SMA suivant :

- un agent gestionnaire des politiques de sécurité,
- un agent gestionnaire extranet,
- deux agents gestionnaires intranet
- cinq agents locaux.

L'objectif de notre scénario est de détecter les quatre attaques définies dans le paragraphe 8.3.2.1 et ce dans un réseau constitué de deux réseaux LANs (voir Figure 43) : LAN₁ et LAN₂.

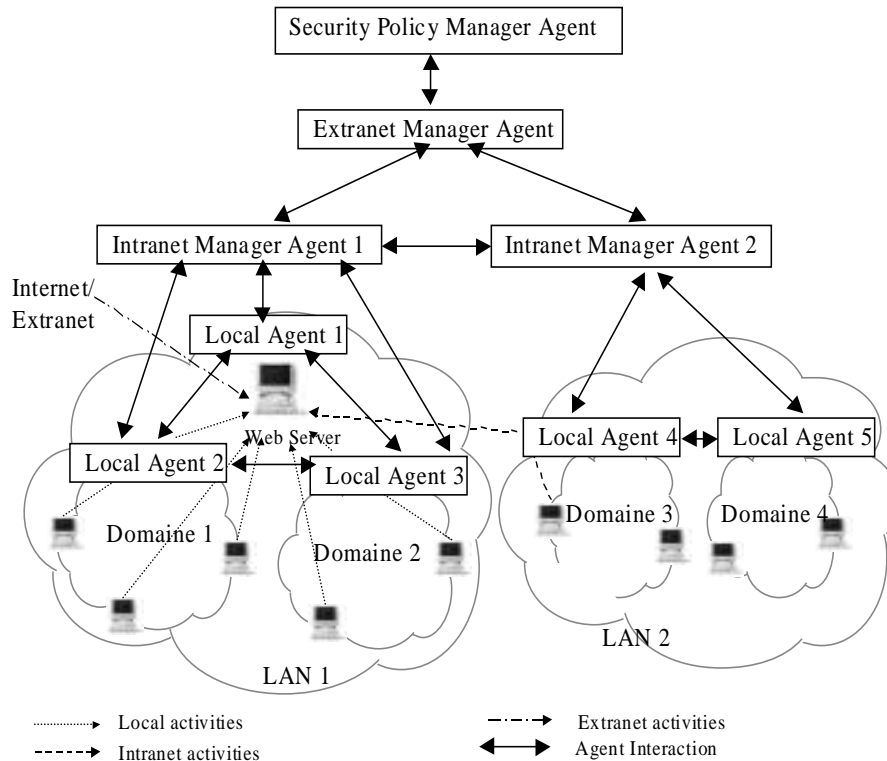


Figure 43 : Exemple de scénario

Dans le réseau LAN₁, nous avons trois agents locaux : AL₁, AL₂ et AL₃. L'agent AL₁ a pour but de surveiller le serveur Web du réseau. Les agents AL₂ et AL₃ ont pour but de surveiller respectivement les domaines 1 et 2. Ces trois agents sont gérés par l'agent gestionnaire intranet AGI₁, qui est responsable du réseau LAN₁.

Dans le réseau LAN₂, nous avons deux agents locaux : AL₄ et AL₅, qui ont pour but de surveiller respectivement les domaines 3 et 4. Ces deux agents sont gérés par l'agent gestionnaire intranet AGI₂, qui est responsable du réseau LAN₂.

Les agents AGI₁ et AGI₂ sont gérés par l'agent gestionnaire extranet AGE qui communique avec l'agent gestionnaire des politiques de sécurité AGPS.

8.4 Conclusion

Dans ce chapitre nous avons présenté l'implémentation de notre SMA. Pour cela, nous avons commencé par implémenter un simulateur de messages. Puis, nous avons choisi un environnement de développement dont le choix s'est porté sur la plateforme DIMA. Ensuite, nous avons entamé l'implémentation des différentes classes d'agents.

En terme de résultats, malgré que nous n'ayons pas finalisé l'implémentation de notre SMA, les phases d'expérimentation réalisées ont permis de valider la majorité des concepts présentés dans la thèse.

Chapitre 9 Conclusion Générale

9.1 Introduction

Dans ce travail nous nous sommes focalisé sur le problème de la gestion de la sécurité des réseaux plus précisément la *détection d'intrusions*. Notre point de départ a été de faire une analyse des systèmes de détection d'intrusions existants, puis d'étudier les technologies agents pour voir comment elles pouvaient apporter une solution optimale au problème de détection d'intrusions.

Nous avons analysé les systèmes de détection d'intrusions existants afin :

- d'identifier les faiblesses des systèmes existants par rapport à l'évolution des besoins de sécurité ;
- de déterminer les caractéristiques nécessaires pour une détection d'intrusions efficace et plus flexible.

Les systèmes de détection d'intrusions existants ont été conçus pour des environnements connus et bien définis. Ils ne sont donc pas adaptés à des environnements dynamiques et c'est là leur principale faiblesse. Dans ce type d'environnement où les besoins en sécurité sont en perpétuelle augmentation, flexibilité et adaptativité deviennent des critères primordiaux.

Parmi les systèmes susceptibles de respecter ces critères, nous avons identifié les systèmes multi-agents. Nous avons donc lancé une étude destinée à s'assurer de la validité de cette hypothèse.

A l'issue de ces deux études, nous avons comparé les propriétés des systèmes multi-agents (SMA) et les caractéristiques nécessaires pour répondre à l'évolution des besoins de sécurité. Nous sommes arrivés à la conclusion que l'une des solutions pour améliorer la sécurité des réseaux est d'intégrer la technologie de SMA dans les systèmes de détection d'intrusions et de proposer ainsi une nouvelle génération de systèmes de détection d'intrusions par systèmes multi-agents.

Nous avons également étudié les différents modèles d'agents existants : les modèles réactifs, les modèles délibératifs et les modèles hybrides, combinaison des deux premiers.

Ces derniers, bien que peu nombreux, sont bien adaptés à notre application de détection d'intrusions pour raisonner sur les attaques complexes et détecter rapidement les attaques simples.

9.2 Contributions

Dans l'objectif de proposer une nouvelle approche, qui offre une solution plus flexible au problème de gestion de sécurité, nous avons conçu un modèle de détection d'intrusions multi-agents. Notre contribution se résume principalement aux points suivants :

Un modèle de gestion des événements de sécurité

L'analyse des attaques et des événements de sécurité, nous a amené à proposer un langage pour la représentation des schémas d'attaque et le traitement des événements. Dans ce langage, nous avons défini deux fonctions principales : la fonction de filtrage des événements de sécurité et la fonction de reconnaissance de schémas d'attaque.

Un modèle de gestion des politiques de sécurité

Dans le but de pouvoir instancier des schémas d'attaque, nous avons proposé un nouveau modèle hiérarchique de gestion de politiques à trois niveaux où nous avons défini la notion de politique schéma, qui est le niveau le plus bas de notre modèle. Nous avons enrichi le langage de spécification des politiques de gestion, défini par Sloman, avec le langage de représentation des schémas d'attaque que nous avons défini, dans le modèle événementiel.

Un modèle multi-agents organisationnel

Pour remplir les deux fonctions : spécification des politiques de sécurité et distribution de la détection des schémas d'attaque, nous avons proposé un modèle organisationnel où nous avons défini deux groupes d'agents : agents gestionnaires et agents surveillants locaux, structurés de manière hiérarchique et interagissant pour détecter aussi bien les attaques locales que globales.

Un modèle d'agents basé sur le BDI

Pour réaliser la détection des attaques, nous avons défini un modèle d'agent basé sur le modèle BDI. Ce modèle pilote notre modèle opérationnel (modèle événementiel). Les agents sont guidés par les buts de détection qui leurs spécifient les schémas d'attaque à détecter. Après analyse des événements filtrés par le modèle opérationnel, les agents s'échangent leurs croyances et leurs suspicions pour reconnaître les attaques. Pour implémenter ce modèle d'agent, nous avons utilisé la plate-forme DIMA. Nous l'avons choisie car elle offre les caractéristiques des agents de notre modèle de détection d'intrusions : agents hybrides et interactifs.

Un module délibératif BDI dans DIMA

Nous avons intégré dans DIMA, un module délibératif construit sur l'architecture BDI. Nous avons ainsi enrichi la librairie de modèles d'agents DIMA par un modèle BDI.

9.3 Perspectives

Il est intéressant de noter que le modèle proposé peut prévoir de nombreuses extensions, dont les principales sont les suivantes :

Apprentissage

Dans les systèmes de détection d'intrusions existants, la notion d'apprentissage est utilisée pour apprendre les comportements normaux des utilisateurs et du système à sécuriser et construire ainsi des profils normaux de comportements. Puis utiliser ces profils pour les comparer avec les profils courants. Dans notre cas, nous pensons qu'il serait plus intéressant d'utiliser cette notion pour apprendre les comportements anormaux qui correspondent à des attaques.

Nous avons introduit dans le Chapitre 3, l'importance de l'apprentissage. Il est donc nécessaire d'intégrer un mécanisme d'apprentissage de nouveaux schémas d'attaque dans notre modèle. Pour cela, nous pensons ajouter une fonction d'apprentissage dans l'architecture interne des différents agents.

Pour réaliser cette fonction, les agents utiliseraient les schémas d'attaque existants et les attaques qui se sont déjà produites, pour identifier des similitudes avec des suites d'événements qui ne correspondraient ni à une suite normale d'événements ni à un schéma d'attaque.

Lorsque des similitudes seraient identifiées, cela serait reporté à l'administrateur, qui pourrait confirmer ou non l'existence d'une attaque. A partir du moment où une attaque serait identifiée, les différents agents devraient intégrer le nouveau schéma d'attaque. Cela pourrait se faire de deux façons :

- soit c'est l'administrateur qui spécifie le nouveau schéma d'attaque à l'agent gestionnaire des politiques de sécurité ;
- soit c'est l'agent gestionnaire qui construit le nouveau schéma d'attaque à partir des événements observés par les différents agents locaux. Une fois le schéma construit, l'agent gestionnaire pourrait distribuer le nouveau schéma d'attaque aux différents agents.

Pour l'apprentissage, nous identifions les interactions d'agents suivantes :

- interactions entre agents surveillants locaux pour l'identification d'une suite suspecte d'événements ;
- interactions entre agents locaux et agents gestionnaires puis agents gestionnaires et administrateur, dans le sens ascendant, pour la confirmation de la détection d'une attaque ;
- interactions entre l'administrateur et l'agent gestionnaire des politiques ou entre les agents gestionnaires et les agents surveillants locaux, dans le sens descendant pour l'apprentissage de nouveaux schémas d'attaque.

Validation à grandeur réelle

L'implémentation de notre système de détection d'intrusions à base d'agents devra finaliser sous forme d'un premier prototype. Pour réaliser ce prototype, nous avons limité les validations à de simples scénarios. Une étape ultérieure consistera en des expérimentations à grandeur réelle. Enfin, des études de performances devront être menées pour identifier le nombre d'alarmes positives, d'alarmes négatives et de fausses alarmes.

Mécanismes de réponse

Dans cette thèse, nous ne nous sommes intéressés qu'aux méthodes de détection des attaques. Il serait également intéressant de définir un modèle capable de réagir aux attaques lorsqu'elles sont détectées. Nous avons introduit, dans le Chapitre 5, la notion de politiques de réponse aux schémas d'attaque qui permettent la mise en œuvre des mesures de réponse. Ces politiques, dérivant des politiques d'obligation, permettraient de définir les actions que l'administrateur et les agents auraient à entreprendre ou non en cas d'attaque.

Nous pensons notamment à des actions concernant le paramétrage d'outils tels que les firewalls. Cela impliquerait, par exemple de modifier les paramètres d'un firewall ou d'un routeur pour filtrer des paquets provenant d'une adresse suspecte ou décelée comme étant l'origine d'une attaque. Les agents pourraient réagir en fonction des outils existants dans l'entreprise, qui leur auront été spécifiés par l'administrateur.

Intégration de la mobilité

Pour intégrer des mécanismes de réponse et les améliorer, nous pensons intégrer la mobilité dans les agents locaux de notre modèle multi-agents. Cela serait fait dans le cadre d'un projet avec l'université de Genève.

Publications

1. Conferences

1. **N. Agoulmine, K. Boudaoud and B. Bhushan.** *A Mobile Agent –based Security Architecture for Intranets.* Proc. of the 2nd World Multi-conference on Systemics, Cybernetics and Informatics and, the 4th International Conference on Information Systems Analysis and Synthesis (SCI/ISAS'98), Orlando, Florida, USA, July 12-16, 1998.
2. **K. Boudaoud and H. Labiod.** *MA-NID: a Multi-Agent System for Network Intrusion Detection.* Proc. of the 8th International Conference on Intelligent Systems (ICIS'99), Denver, Colorado, USA, June 24 - 26, 1999.
3. **K. Boudaoud, N. Agoulmine and J. N De Souza.** *Distributed Network Security Management Using Intelligent Agents.* Proc. of the IEEE Latin American Network Operations and Management Symposium (LANOMS'99), Rio de Janeiro, Brazil, December 3-5, 1999.
4. **K. Boudaoud, H. Labiod, Z. Guessoum and R. Boutaba.** *Network Security Management with Intelligent Agents.* Proc. of the 2000 IEEE/IFIP Network Operations and Management Symposium, Honolulu, Hawaii, April 10-14, 2000.
5. **K. Boudaoud and Noria Foukia.** *An Intelligent Agent Approach for Security Management.* Proc. of the HP Openview University Association (HP-OVUA) 7th Plenary Workshop, Santorini, Greece, June 12 - 14, 2000.
6. **K. Boudaoud and Z. Guessoum.** *A Multi-agents System for Network Security Management.* Proc. of the Sixth IFIP Conference on Intelligence in Networks (SmartNet'2000), Vienna, Austria, September 18-22, 2000.
7. **K. Boudaoud.** *Un système multi-agents pour la détection d'intrusions.* Proc. des Journées Doctorales Informatique et Réseaux (JDIR'2000), Paris, France, Novembre 6-8, 2000.

2. Journaux

1. **H. Labiod, K. Boudaoud and J. Labetoulle.** *Towards a new approach for intrusion detection with intelligent agents.* Networking and Information Systems Journal, 2 (5-6): 701-739, 1999

3. Rapports internes

1. **K. Boudaoud.** *Etat de l'art sur la gestion de la sécurité informatique et la détection d'intrusion.* Octobre, 1998.
2. **K. Boudaoud.** *Analyse des attaques de sécurité par rapport aux propriétés des agents intelligents.* Novembre, 1998.

Références bibliographiques

- [Alagha and Labiod 1999] **K. Alagha and H. Labiod.** *MA-WATM:A new Approach Towards an Adaptive Wireless ATM Network.* ACM MONET Journal, Baltzer publishers, 4 (2), 1999.
- [Balasubramaniyan et al. 1998] **J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. H. Spafford and D. Zamboni.** *An Architecture for Intrusion Detection using Autonomous Agents.* Technical Report Coast-TR-98-05, Computer Sciences Department, Purdue University, 1998.
- [Bourron 1992] **T. Bourron.** *Structures de communication et d'organisation pour la coopération dans un univers multi – agents.* Thèse de l'université de Paris 6, Laforia, 1992.
- [Brazier et al. 1996] **F. Brazier, B. Dunin-Keplicz, J. Treur, R. Verbrugge.** *Beliefs, Intentions and DESIRE.* Proc. Of the 10th Banff Knowledge Acquisition for Knowledge-based Systems Workshop (KAW'96), Calgary: SRDG Publications, Department of Computer Science, University of Calgary, 1996.
- [Brooks 1991] **R. A. Brooks.** *Intelligence without Representation.* Artificial Intelligence, 47 : 139-159, 1991.
- [Castelfranchi 1995] **C. Castelfranchi.** *Commitments : From Individual Intentions to Groups and Organizations.* Proc. of the First International Conference on Multi – Agent Systems, San Francisco, pp. 41-48, June 1995.
- [Cert 1995] **CERT Advisory.** *IP Spoofing Attacks and Hijacked Terminal Connections.* CA-95.01, January 23, 1995.
- [Cert 1996] **CERT Advisory.** *TCP SYN Flooding and IP Spoofing Attacks.* CA-96.21, September 1996.
- [Cert 1998] **CERT Advisory.** *Smurf IP Denial-of-Service Attacks.* CA-98.01, January 1998.
- [Cheikhrouhou et al. 1998] **M.Cheikhrouhou, P. Conti, R. Texeira Oliveira and J. Labetoulle.** *Intelligent Agents in Network Management, a state of the art.* Networking and Information Systems, 1(1), 9-38, 1998.
- [Cheung et al. 1995] **S. Cheung, K. Levitt and C. Ko.** *Intrusion Detection for Network Infrastructures.* Proc. of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, 1995.

- [Cheung et al. 1997] **S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, S. Staniford-Chen, R. Yip, D. Zerkle.** *The Design of GrIDS: A Graph-Based Intrusion Detection System*. Technical Report, Department of Computer Sciences, UC Davis, 1997.
- [Cohen and Levesque 1988] **P. R. Cohen and H. J. Levesque.** *Intention In Choice with Commitment*. Artificial Intelligence, pp. 213-261, 1988.
- [Cohen and Levesque 1990] **P. R. Cohen and H. J. Levesque.** *Rational Interaction as the Basis for Communication*. Artificial Intelligence, pp. 225-253, 1990.
- [Conti 2000] **P. Conti.** *Agents Intelligents : Emergence d'une nouvelle technologie pour la Gestion de Réseaux*. Thèse de l'ENST de Paris, 2000.
- [Crosbie and Spafford 1995a] **M. Crosbie and E. H. Spafford.** *Active Defense of a Computer System using Autonomous Agents*. Technical Report CSD-TR-95-008, Purdue University, 1995.
- [Crosbie and Spafford 1995b] **M. Crosbie and E. H. Spafford.** *Defending a Computer System using Autonomous Agents*. Technical Report CSD-TR-95-022, Computer Sciences Department, Purdue University, 1995.
- [Crosbie and Spafford 1996] **M. Crosbie and E. H. Spafford.** *Applying Genetic Programming to Intrusion Detection*. Technical Report, Computer Sciences Department, Purdue University, 1996.
- [Crosbie et al. 1996] **M. Crosbie and E. H. Spafford.** *IDIOT-Users Guide*. Technical Report CSD-TR-96-050, Computer Sciences Department, Purdue University, 1996.
- [Damianou et al. 2000] **N. Damianou, N. Dulay, E. Lupu and M. Sloman.** *A Language for Specifying Security and Management Policies for Distributed Systems*. Imperial College Research Report Doc 2000/1, Department of Computing, Imperial College of Science Technology and Medicine, University of London, England, 2000.
- [Demazeau and Müller 1990] **Y. Demazeau, J. P. Müller.** *Decentralized Artificial Intelligence (1)*. Yves Demazeau and Jean-Pierre Müller (Eds), Elsevier Science Publisher B. V., pp. 3-16, 1990.
- [Demazeau and Müller 1991] **Y. Demazeau, J-P Müller.** *Decentralized Artificial Intelligence (2)*. Yves Demazeau and Jean-Pierre Müller (Eds.), Elsevier Science Publisher B.V. (North-Holland), pp. 3-10, 1991.
- [Denning 1987] **D. E. Denning.** *An Intrusion-Detection Model*. IEEE Transactions on Software Engineering, 13 (2), February 1987.
- [Erceau et Ferber 1991] **J. Erceau and J. Ferber.** *L'intelligence Artificielle Distribuée*. La Recherche, 23, 1991.

- [Escamilla 1998] **T. Escamilla.** *Intrusion Detection*. Published by John Wiley & Sons, Inc, 1998.
- [Fenzi 1998] **K. Fenzi.** *Linux Security HOWTO*. HOWTO Document, <http://sunsite.unc.edu/LDP/HOWTO/Security-HOWTO.html>, 1998.
- [Ferber 1995] **J. Ferber.** *Les Systèmes Multi-Agents, Vers une intelligence collective*. InterEditions, 1995.
- [Ferber 1996] **J. Ferber.** *Technologies Multi-Agent*. Actes du Forum France Télécom Recherche, Mémento Technique, 8, pp. 63-79, 1996.
- [Ferber and Ghallab 1998] **J. Ferber and M. Ghallab.** *Problématique des Univers Multi - Agents Intelligents*. Journées Nationales du PRC IA, Toulouse, pp. 295-320, 1998.
- [Ford 1994] **W. Ford.** *Computer Communications Security: Principles, Standard Protocols and Techniques*. Ed. Prentice Hall PTR, 1994.
- [Frank 1994] **J. Frank.** *Artificial Intelligence and Intrusion Detection: Current and Future*. Technical Report, Division of Computer Science, University of California, 1994.
- [Gallagher 1987] **P. R. Gallagher.** *A Guide to Understanding Audit in Trusted Systems*. Technical Guidelines NCSC-TG-001, 1987.
- [Guessoum 1996] **Z. Guessoum.** *Un environnement opérationnel de conception et de réalisation de systèmes multi – agents*. Thèse de l'université de paris 6, Laforia, 1996.
- [Gutknecht et Ferber 1998] **O. Gutknecht, J. Ferber.** *Un Méta - Modèle Organisationnel pour l'Analyse, la Conception et l'Exécution de Systèmes Multi-Agents*. Actes des JFIADSMA'98, Editions Hermès, 1998.
- [Gutknecht et Ferber 1999] **O. Gutknecht, J. Ferber.** *Vers une Méthodologie Organisationnelle de Conception de Systèmes Multi – Agents*. Rapport de recherche LIRMM 99073, Université de Montpellier, LIRM, Juin 1999.
- [Gutknecht, Ferber et Michel 2000] **O. Gutknecht, J. Ferber et F. Michel.** *MadKit: une plate-forme multi-agent générique*. Rapport de recherche LIRMM 00061, Université de Montpellier, LIRM, Mai 2000
- [Habra et al. 1992] **N. Habra, B. Le Charlier, A. Mounji and I. Mathieu.** *ASAX: Software Architecture and Rule-Based Language for Universal Audit Trail Analysis*. European Symposium on Research in Computer Security, Toulouse, France, Springer-Verlag 1992.
- [Habra et al. 1994] **N. Habra, B. Le Charlier, A. Mounji and I. Mathieu.** *Preliminary Report on Advanced Security Audit Trail Analysis on Unix (ASAX*

- also called SAT-X*). Technical Report, Institut d'Informatique, FUNDP-5000 Namur, Belgium, 1994.
- [Halpern and Moses 1992] **J. Y. Halpern and Y. Moses**. *A Guide to Completeness and Complexity for Modal Logics of Knowledge and Belief*. Artificial Intelligence, 54:319-379, 1992.
- [Heberlein 1996] **L. T. Heberlein and M. Bishop**. *Attack Class: Address Spoofing*. Proc of the 19th National Information Systems Security Conference, 1996.
- [Heilbronner 1997] **S. Heilbronner**. *Policies in the Management of Nomadic Computing Systems*. Proc. of the Fourth Workshop of the HP Openview University Association (HP-OVUA'97), Madrid, Spain, April 1997.
- [Heilbronner 1998] **S. Heilbronner**. *Policies in the Management of Nomadic Computing Systems*. Proc. of the Fifth Workshop of the HP Openview University Association (HP-OVUA'98), ENST Bretagne, Rennes, France, April 19-21, 1998.
- [Heilbronner 1999] **S. Heilbronner**. *Requirements for Policy-Based Management of Nomadic Computing Infrastructures*. Proc. of the Sixth Workshop of the HP Openview University Association (HP-OVUA'99), Bologna, Italy, June 1999.
- [Hoagland et al. 1995] **J. Hoagland, C. Wee and K. Levitt**. *Audit Log Analysis Using the Visual Audit Browser Toolkit*. Technical Report CSE-95-11, Department of Computer Sciences, UC Davis, 1995.
- [Hintikka 1962] **J. Hintikka**. *Knowledge and Belief*. Cornell University, Ithaca, New York, 1962.
- [Ilgun 1992] **K. Ilgun**. *USTAT: A Real-time Intrusion Detection System for UNIX*. Master thesis, University of California, 1992.
- [Kaplankiran et al. 2000] **Y. Kaplankiran, A. Keiblinger, H. Többen**. *Active Network Management via Agent Technology*. Proc. Of the Sixth International Conference on Intelligence in Networks (SmartNet 2000), Vienna, Austria, September 18-22, 2000.
- [Kemmerer 1997] **R. A. Kemmerer**. *The STAT Approach*. STAT Projects, <http://www.cs.ucsb.edu/~kemm/netstat.html/projects.html>, 1997.
- [Kinny et al.1996] **D. Kinny M. Georgeff and A. Rao**. *A Methodology and Modelling technique for systems of BDI agents*. In W. Van de Velde and J. W. Perram, editors, Agents Breaking Away, Proc. of the Seventh European Workshop on Modelling Autonomous Agents in a Multi – Agent World, (LNAI Volume 1038), pp. 56-71, springer-verlag, Berlin, Germany, 1997.
- [Kinny and Georgeff 1997] **D. Kinny and M. Georgeff**. *Modelling and Design of Multi – Agents Systemd*. In J. P. Müller, M. Wooldridge and N. R. Jennings, editors, Intelligent Agents III (LNAI Volume 1193), pp. 1-20, springer-verlag, Berlin, Germany, 1997.

- [Ko et al.1993] **C. Ko, D. A. Frincke, T. Goan, L. T. Heberlein, K. Levitt, B. Mukherjee and C. Wee.** *Analysis of an Algorithm for Distributed Recognition and Accountability*. Proc of the First ACM Conference on Computer and Communication Security, pp. 154-164, 1993.
- [Koch et al. 1996] **T. Koch, C. Krell and B. Krämer.** *Policy Definition Language for Automated Management of Distributed Systems*. Proc. of the Second International Workshop on Systems Management, IEEE Computer Society Press, 1996.
- [Kumar and Spafford 1994a] **S. Kumar and E. G. Spafford.** *An application of Pattern Matching in Intrusion Detection*. Technical Report CSD-TR-94-013, Computer Sciences Department, Purdue University, 1994.
- [Kumar and Spafford 1994b] **S. Kumar and E. G. Spafford.** *A Pattern Matching Model for Misuse Intrusion Detection*. Proc of the 17th National Computer Security Conference, Baltimore, pp. 11-21, 1994.
- [Kumar 1995] **S. Kumar.** *Classification and Detection of Computer Intrusions*. Technical Report Coast-TR-95-08, Coast Laboratory, Purdue University, 1995.
- [Kumar and Spafford 1995] **S. Kumar and E. G. Spafford.** *A Software Architecture to support Misuse Intrusion Detection*. Technical Report CSD-TR-95-009, Purdue University, 1995.
- [Labidi et Lejouad 1993] **S. Labidi and W. Lejouad.** *De l'Intelligence Artificielle Distribuée aux Systèmes Multi – Agents*. Technical report, INRIA Sophia – Antipolis, 1993.
- [Lane and Brodley 1997a] **T. Lane and C. E. Brodley.** *Detecting the Abnormal : Machine Learning in Computer Security*. Technical Report Coast-TR-97-02, Purdue University, 1997.
- [Lane and Brodley 1997b] **T. Lane and C. E. Brodley.** *An Application of Machine Learning to Anomaly Detection*. Technical Report Coast-TR-97-03, Purdue University, 1997.
- [Lane and Brodley 1997c] **T. Lane and C. E. Brodley.** *Sequence Matching and Learning in Anomaly Detection for Computer Security*. Technical Report Coast-TR-97-04, Purdue University, 1997.
- [Lunt and Anderson 1993] **T. F. Lunt and D. Anderson.** *Software Requirements Specification: Next-Generation Intrusion Detection Expert Systems*. Technical Report, SRI International, 1993.
- [Lupu et al 1995] **E. Lupu, D. A. Mariott, M. Sloman and N. Yialelis.** *A Policy Based Role Framework for Access Control*. First ACM/NIST Workshop on Role-Based Access Control, Gaithersburg, Maryland, USA, December 1995.

- [Lupu et al 1997] **E. Lupu, M. Sloman and N. Yialelis.** *Policy Based Roles for Distributed Systems Security*. Proc. of the Fourth Workshop of the HP Openview University Association (HP-OVUA'97), Madrid, Spain, April 1997.
- [Lupu and Sloman 1997a] **E. Lupu and M. Sloman.** *Conflict Analysis for Management Policies*. Proc. of the Fifth IFIP/IEEE International Symposium on Integrated Network Management (IM'97), San Diego, USA, May 1997.
- [Lupu and Sloman 1997b] **E. Lupu and M. Sloman.** *A Policy Based Role Object Model*. Proc. of the First IEEE International Enterprise Distributed Object Computing Workshop (EDOC'97), Gold Coast, Queensland, Australia, October 1997.
- [Lupu and Sloman 1997c] **E. Lupu and M. Sloman.** *Reconciling Role Based Management and Role Based Access Control*. Proc. of the Second Role Based Access Control Workshop (RBAC'97), George Mason University, Virginia USA, November 1997.
- [Lupu and Sloman 1997d] **E. Lupu and M. Sloman.** *Towards a Role-based Framework for Distributed Systems Management*. Journal of Network and Systems Management, 5(1), Plenum Press Publishing, pp 5-30, 1997.
- [Lupu et al 1999] **E. Lupu, Z. Milosevic and M. Sloman.** *Use of Roles and Policies for Specifying, and Managing a Virtual Enterprise*. Proc. of the Ninth IEEE International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE-VE'99), Sydney, Australia, March 23-24, 1999.
- [Magendaz 1995] **T. Magendaz.** *On the Impacts of Intelligent Agents Concepts on Future Telecommunication Environments*. Proc. of the Third Interbational Conference on Intelligence in Broadband Services and Networks, Crete, Greece, October 1995.
- [Mariott and Samani 1996] **D. A. Mariott and M. Mansouri-Samani.** *Specification of Management Policies*. Proc. of the Fifth IEEE/IFIP International Workshop on Distributed Systems Operations and Management (DSOM'94), Toulouse, France, October 10-12, 1994.
- [Mariott and Sloman 1996] **D. A. Mariott and M. Sloman.** *Implementation of a Management Agent for Interpreting Obligation Policy*. Proc. of the Seventh IEEE/IFIP International Workshop on Distributed Systems Operations and Management (DSOM'96), L'Aquila, Italy, October 28-30, 1996.
- [Mariott 1997] **D. A. Mariott.** *Policy Service for Distributed Systems*. PhD thesis, Department of Computing, Imperial College of Science Technology and Medicine, University of London, England, 1997.
- [Marshall 1991] **V. H. Marshall.** *Intrusion Detection in Computers*. Trusted Information Systems Report, 1991.

- [Mé et Alanou 1996] **L. Mé et V. Alanou.** *Détection d'intrusions dans un système informatique: méthodes et outils.* TSI Journal, 96(4), pp. 429-450, 1996.
- [Mé 1994] **L. Mé.** *Audit de sécurité par algorithmes génétiques.* Thèse de l'université de Rennes 1, France, 1994.
- [Mé 1995] **L. Mé.** *Un algorithme génétique pour détecter des intrusions dans un système informatique.* Valgo Journal, Volume 95, pp. 68-78, 1995.
- [Meyer et al. 1995] **K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt and Y. Yemini.** *Decentralizing control and intelligence in network management.* Proc of the Fourth International Symposium on Integrated Network Management, 4, 1995.
- [Meyer et al. 1996] **B. Meyer, F. Anstötz, and C. Popien.** *Towards Implementing Policy-based Systems Management.* IEE/IOP/BCS Distributed Systems Engineering Journal, 3(2), pp. 78-85, 1996.
- [Mitchell 1992]. **T. M. Mitchell.** *Learning agents.* Workshop on Approaches to Artificial Intelligence, Santa FE, New Mexico, November 1992.
- [Moffett 1993 and Sloman] **J. D. Moffett and M. Sloman.** *Policy Hierarchies for Distributed Systems Management.* IEEE Journal on Selected Areas in Communications, 11 (9), pp. 1404-1414, December 1993.
- [Mounji 1997] **A. Mounji.** *Rule-Based Distributed Intrusion Detection.* Thèse de l'université de Namur, Institut d'Informatique, Belgique 1997.
- [Mukherjee et al. 1994] **B. Mukherjee, L.T. Heberlein, and K.N.Levitt.** *Network Intrusion Detection.* IEEE Network Journal, pp. 26-41, May/June 1994.
- [Müller 1996] **J. P. Müller.** *The Design of Intelligent Agents – A layered Approach.* Lecture Notes in artificial Intelligence, springer 1996.
- [Müller 1998] **J. Müller.** *The right architecture to do the right thing.* Atal'99, Springer Verlag, pp. 1-18, 1998.
- [Oliveira 1998] **R. F. Teixeira de Oliveira.** *Gestion des Réseaux avec Connaissance des Besoins: Utilisation des Agents Logiciel.* Thèse de l'ENST de Paris, 1998.
- [Northcutt 1999] **S. Northcutt.** *Network Intrusion Detection – An Analyst's Handbook.* Published by New Riders, 1999.
- [Nwana 1996] **H. Nwana and M. Wooldridge.** *Software Agent: An overview.* Knowledge Engineering Review, 11(3) : 205-244, 1996.
- [Porras 1992] **P. A. Porras.** *A State Transition Analysis Tool for Intrusion Detection.* Master thesis, University of California, 1992.

- [Porras 1995] **P. A. Porras**. *State Transition Analysis : A Rule-Based Intrusion Detection Approach*. IEEE Transactions on Software Engineering, pp. 181-199, 1995.
- [Price 1998] **K. Price**. *Intrusion Detection Pages*. Purdue University, 1998. <http://www.cs.purdue.edu/coast/intrusion-detection/ids.html>.
- [Puketza et al. 1997] **N. Puketza, M. Chung, R. A. Olsson and B. Mukherjee**. *A Software Platform for Testing Intrusion Detection Systems*. IEEE Software Journal, pp. 43-51, 1997.
- [Rao and Georgeff 1991a] **A. S. Rao and M. P. Georgeff**. *Modelling Agents within a BDI – Architecture*. In R. Fikes and E. Sandewall, editors, Proc. of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91), pp. 473-484, April 1992.
- [Rao and Georgeff 1991b] **A. S. Rao and M. P. Georgeff**. *Modelling Rational Agents within a BDI – Architecture*. Technical report 14, Australian AI Institute, Carlton, Australia, 1991.
- [Rao and Georgeff 1991c] **A. S. Rao and M. P. Georgeff**. *Intelligent Real-Time Network Management*. Technical report 15, Australian AI Institute, Carlton, Australia, 1991.
- [Rao and Georgeff 1992] **A. S. Rao and M. P. Georgeff**. *An abstract architecture for rational agents*. Proc of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92), pp. 439-449, October 1992.
- [Rao et al 1993] **A. S. Rao, A. Lucas, D. Morley, M. Selvestrel and G. Murray**. *Agent – Oriented Architecture for Air Combat Simulation*. Technical report 42, Australian AI Institute, Carlton, Australia, Avril 1993.
- [Rao and Georgeff 1995] **A. S. Rao and M. P. Georgeff**. *BDI – agents: from theory to practice*. Proc. of the First International Conference on Multi – Agent Systems, San Francisco, 1995.
- [Samfat 1996] **D. Samfat**. *Architecture de Sécurité pour Réseaux Mobiles*. Thèse de l'ENST de Paris, 1996.
- [Shoham 1993] **Y. Shoham**. *Agent Oriented Programming*. Artificial Intelligence 60(1), pp. 51-92, 1993.
- [Shoham and Cousins 1994] **Y. Shoham and S. B. Cousins**. *Logics of Mental Attitudes in AI: a very preliminary survey*. In G. Lakemeyer and B. Nebel (eds.) Foundations of Knowledge Representation and Reasoning, Springer Verlag, pp. 296-309, 1994.
- [Shuba et al. 1997] **C. L. Shuba, I. V. Krsul, M. G. Kuhn, E. G. Spafford, A. Sundaram and D. Zamboni**. *Analysis of a Denial of Service Attack on TCP*. Technical Report Coast-TR-97-06, Purdue University, 1997.

- [Skarmeas 1998] **N. Skarmeas.** *Agents as Objects with Knowledge Base State.* Imperial College Press, 1998
- [Sloman 1993] **M. Sloman.** *Policy Driven Management for Distributed Systems.* Journal of Network and Systems Management, 2(4), Plenum Press Publishing, 1994.
- [Smith 1980] **R. G. Smith.** *Framework for Distributed Problem Solving.* UNI Research Press, 1980.
- [Smith and Davis 1981] **R. G. Smith and R. Davis.** *Framework for Cooperation in Distributed Problem Solving.* IEEE Transactions on Systems, Man and Cybernetics, SMC, 11(1), pp. 61-70, 1981.
- [Smith et al 1997] **J. M. Smith, D. J. Farber, C. A. Gunter, S.M. Nettles, M. E. Segal, W.D. Sincoskie, D.C. Feldmeier and D. S. Alexander.** *SwitchWare: towards a 21st century network infrastructure.* Technical Report, CIS Department, University of Pennsylvania, 1997.
- [Spafford 1988] **E. H. Spafford.** *The Internet Worm Program: An Analysis.* Technical Report CSD-TR-823, Purdue University, 1988.
- [Sri 1997a] **SRI/CSL.** *History of Intrusion Detection at SRI/CSL.* SRI International, Computer Science Laboratory, <http://www.CSL.sri.com/intrusion.html>, 1997.
- [Sri 1997b] **SRI/CSL.** *Next-Generation Intrusion Detection Expert System.* SRI International, Computer Science Laboratory, <http://www.CSL.sri.com/nides/index1.html>, 1997.
- [Stallings 1995] **W. Stallings.** *Network and Internetwork Security: principles and practice.* Ed. Prentice-Hall, 1995
- [Staniford-Chen et al. 1996] **S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, S. Templeton, K. Levitt, S. Walnum, C. Wee, and R. Yip.** *GrIDS-A Graph-Based Intrusion Detection System for Large Networks.* Proc of the 19th National Information Systems Security Conference, 1996.
- [Staniford-Chen 1997] **S. Staniford-Chen.** *GrIDS Outline Design Document.* GrIDS Project Home Page at UC Davis's Computer Science Department, <http://olympus.cs.ucdavis.edu/arpa/grids/design.html>, 1997.
- [Steenekamp and Roos 1996] **P. Steenekamp and J. Roos.** *Implementation of Distributed Systems Management Policies: A Framework for the Application of Intelligent Agent Technology.* Proc. of the Second International IEEE Workshop on Systems Management, Toronto, Ontario, Canada, June 19-21, 1996.
- [Sundaram 1996] **A. Sundaram.** *An Introduction to Intrusion Detection.* Technical Report, Purdue University, 1996.

- [Tennenhouse et al. 1997] **D. L. Tennenhouse, J. M. Smith, W.D. Sincoskie, D. J. Wetherall and G. J. Minden.** *A survey of active network research.* IEEE Communications Magazine, 35(1), 80-86, 1997.
- [Udod 1985] **U.S. Dep. Defense DOD.** *Dep. Defense trusted computer system evaluation criteria (the orange book).* Technical Report 5200.28-STD, DoD, 1985.
- [Walker and Cavanaugh 1998] **K. M. Walker and L. C. Cavanaugh.** *Computer Security Policies and SunScreen Firewalls.* Sun Microsystems Press, A Prentice Hall Title, 1998.
- [White et al. 1996] **B. White, E. A. Fisch, and U. W. Pooch.** *Cooperating Security Managers: A Peer-Based Intrusion Detection System.* IEEE Network Journal, pp. 20-23, January/February 1996.
- [Wies 1995a] **R. Wies.** *Policies in Integrated Network and Systems Management: Methodologies for the Definition, Transformation and Application of Management Policies.* PhD thesis, University of Munich, 1995.
- [Wies 1995b] **R. Wies.** *Using a Classification of Management Policies for Policy Specification and Policy Transformation.* Proc. of the Third IFIP/IEEE International Symposium on Integrated Network Management, Santa Barbara, CA, USA, May 1995.
- [Wies et al. 1997] **R. Wies, M. Mountzia and P. Steenekamp.** *A Practical Approach Towards a Distributed and Flexible Realization of Policies using Intelligent Agents.* Proc. of the Eighth IFIP/IEEE International Workshop on Distributed Systems: Operations & Management, October 1997.
- [Wooldridge 1992] **M. J. Wooldridge.** *On the Logical Modelling of Computational Multi-Agent Systems.* Thèse de l'université de Manchester, UMIST, Department of Computation, Manchester, UK, 1992.
- [Wooldridge and Jennings 1995] **M. Wooldridge and N. R. Jennings.** *Intelligent Agents: Theory and Practice.* Knowledge Engineering Review, 10(2) : 115-152, 1995.
- [Wooldridge et al. 1999] **M. Wooldridge, N. R. Jennings and D. Kinny.** *A Methodology for Agent Oriented Analysis and Design.* Proc. of the Third International Conference on Autonomous Agents (Agents 99), Seattle, WA, pp. 69-76, May 1999.
- [Yialelis and Sloman 1995a] **N. Yialelis and M. Sloman.** *An Authentication Service Supporting Domain Based Access Control Policies.* Imperial College Research Report Doc 1995/13, Department of Computing, Imperial College of Science Technology and Medicine, University of London, England, 1995.
- [Yialelis and Sloman 1995b] **N. Yialelis and M. Sloman.** *A Security Framework Supporting Domain Based Access Control in Distributed Systems.* Imperial

College Research Report Doc 1995/14, Department of Computing, Imperial College of Science Technology and Medicine, University of London, England, 1995.

[Yialelis et al. 1996] **N. Yialelis, E. Lupu and M. Sloman.** *Role-Based Security for Distributed Object Systems*. Proc. of the Fifth IEEE Workshop on Enabling Technology: Infrastructure for Collaborative Enterprises (WET-ICE'96), Standford, California, USA, June 19-21, 1996.

[Yialelis 1996] **N. Yialelis.** *Domain-Based for Distributed Object Systems*. PhD thesis, Department of Computing, Imperial College of Science, Technology and Medicine, University of London, England, 1996.

Annexe A

CATEGORIE	NOM	SIGNIFICATION	R.A	R	P.R	: CONTENT
Discours	ask-if	E veut savoir si le : content existe dans la base de connaissances de D.	X			<expression>
	ask-all	E veut connaître toutes les instanciations de : content qui sont vraies pour D.	X			<expression>
	ask-one	E veut connaître une seule instanciation de : content qui est vraie pour D.	X			<expression>
	stream-all	Idem que ask-all – les instanciations vraies pour D sont envoyées chacunes dans un message.	X			<expression>
	eos	<i>end-of-stream</i> marque la fin d'une réponse multiple engendrée par un stream-all .		X		vide
	tell	le : content est dans la base de connaissances de E.		X		<expression>
	untell	le : content n'est pas dans le base de connaissances de E.		X		<expression>
	deny	la négation du : content est dans la base de connaissances de E.		X		<expression>
	insert	E demande à D d'ajouter : content dans sa base de connaissances.			X	<expression>
	uninsert	E vient d'envoyer un insert et revient sur sa décision.			X	<expression>
	delete-one	E veut que D retire un type d'instanciation de sa base de connaissance.			X	<expression>
	delete-all	E veut que D retire toutes les instanciations d'un type donné de sa base de connaissance.			X	<expression>
undelete	E vient d'envoyer un delete-x et revient sur sa décision.			X	<expression>	

	achieve	E veut que D teste la véracité de : content dans sa base de connaissance.			X	<expression>
	unachieve	E vient d'envoyer un achieve et revient sur sa décision.			X	<expression>
	advertise	E veut savoir si D peut et va envoyer un message de type : content .			X	<expression>
	unadvertise	E veut savoir si D annule un précédent advertise et ne va plus générer de message de type : content .			X	<expression>
	subscribe	E veut mettre à jour la réponse à D par un message (peut être utilisé à tout moment).	X			<expression>
Inter-ventions sur la mécanique de la conversation	error	E considère le dernier message de D comme n'étant pas conforme au protocole.		X		vide
	sorry	E comprend le message de D mais ne peut pas générer une réponse plus instructive.		X		vide
	standby	E veut prévenir D qu'il ne peut pas envoyer une réponse immédiate au message de D.	-	-	-	<message>
	ready	E veut prévenir qu'il est maintenant prêt à répondre à un précédent message de D.	-	-	-	vide
	next	E veut connaître la prochaine réponse de D à un précédent message envoyé par E.	-	-	-	vide
	rest	E veut connaître la suite de la réponse de D à un message précédemment envoyé par E.	-	-	-	vide
	discard	E ne veut pas connaître la suite de la réponse (multiple réponse) de D à un message précédemment envoyé par E.	-	-	-	vide

Gestion du réseau et del'équipement	register	E annonce à D qu'il est présent et lui envoie son nom symbolique associé à son adresse physique.			X	<expression>
	unregister	E revient sur un précédent register .			X	vide
	forward	E veut communiquer (: content) avec un agent en communication avec D, attendu que D ne peut pas communiquer directement avec l'agent en communication avec E. (Utilisation de la transitivité des communications inter agents)			C	<message>
	broadcast	E veut envoyer un message à tous les agents en communication avec S.			C	<message>
	transport-adress	E prévient que son nom symbolique correspond dorénavant à une nouvelle adresse.			X	<expression>
	broker-one	E veut trouver une réponse à un message (un agent autre que D va répondre à la question).			C	<message>
	broker-all	E veut trouver toutes les réponses à un message (un agent autre que D va répondre à la question).			C	<message>
	recommend-one	E veut interroger un agent pour savoir qui peut répondre à sa demande.	X			<message>
	recommend-all	E veut savoir de tous les agents, qui peut répondre à sa demande.	X			<message>
	recruit-one	E veut que l'agent D désigne un agent capable de répondre à sa question.			C	<message>
	recruit-all	E veut que l'agent D désigne tous les agents capables de répondre à sa question.			C	<message>

Figure A13 : Tableau récapitulatif des types de messages KQLM.

